Africa *et al. BMC Bioinformatics* (2023) 24:143
https://doi.org/10.1186/s12859-023-05260-w

SOFTWARE

Open Access

lifex-fiber: an open tool for myofibers generation in cardiac computational models

Pasquale Claudio Africa[1*], Roberto Piersanti[1], Marco Fedele[1], Luca Dede'[1] and Alfio Quarteroni[1,2]

*Correspondence: pasqualeclaudio.africa@polimi.it

[1] MOX, Department of Mathematics, Politecnico di Milano, Milano, Italy
[2] Institute of Mathematics, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

Abstract

Background: Modeling the whole cardiac function involves the solution of several complex multi-physics and multi-scale models that are highly computationally demanding, which call for simpler yet accurate, high-performance computational tools. Despite the efforts made by several research groups, no software for whole-heart fully-coupled cardiac simulations in the scientific community has reached full maturity yet.

Results: In this work we present `life`$^{\mathrm{x}}$-fiber, an innovative tool for the generation of myocardial fibers based on Laplace-Dirichlet Rule-Based Methods, which are the essential building blocks for modeling the electrophysiological, mechanical and electromechanical cardiac function, from single-chamber to whole-heart simulations. `life`$^{\mathrm{x}}$-fiber is the first publicly released module for cardiac simulations based on `life`$^{\mathrm{x}}$, an open-source, high-performance Finite Element solver for multi-physics, multi-scale and multi-domain problems developed in the framework of the iHEART project, which aims at making *in silico* experiments easily reproducible and accessible to a wide community of users, including those with a background in medicine or bio-engineering.

Conclusions: The tool presented in this document is intended to provide the scientific community with a computational tool that incorporates general state of the art models and solvers for simulating the cardiac function within a high-performance framework that exposes a user- and developer-friendly interface. This report comes with an extensive technical and mathematical documentation to welcome new users to the core structure of `life`$^{\mathrm{x}}$-fiber and to provide them with a possible approach to include the generated cardiac fibers into more sophisticated computational pipelines. In the near future, more modules will be successively published either as pre-compiled binaries for `x86-64 Linux` systems or as open source software.

Keywords: Computational cardiology, High-performance computing, Cardiac fibers, Mathematical modeling, Finite element methods

Mathematical Subject Classification: Primary 68-04, 68N30, Secondary 35-04, 65-04, 65M60, 65N30, 65Y05, 65Y20, 92-04, 92C50

Background

The human heart function is a complex system involving interacting processes at the molecular, cellular, tissue, and organ levels with widely varying time scales. For this reason, it is still among the most arduous modeling and computational challenges in a field where *in silico* models and experiments are essential to reproduce both physiological and pathological behaviors [1].

A satisfactorily accurate model for the whole cardiac function must be able to describe a wide range of different processes, such as: the propagation of the trans-membrane potential and the flow of ionic species in the myocardium, the deformation caused by the muscle contraction, the dynamics of the blood flow through the heart chambers and cardiac valves [2]. In particular, the dynamics of ionic species needs for accurate models specifically designed to reproduce physiological [3] and pathological scenarios [4, 5] (such as the *ten Tusscher–Panfilov* [6] and the *Courtemanche-Ramirez-Nattel* [7] ionic models for ventricular/atrial cells, respectively).

These demanding aspects make whole-heart fully-coupled simulations computationally intensive and call for simpler yet accurate, high-performance computational tools.

In this work we introduce `life`^{x}-fiber, an innovative tool for the generation of myocardial fibers based on `life`^{x} [8], an open-source, high-performance Finite Element (FE) numerical solver for multi-physics, multi-scale and multi-domain differential problems. It is written in `C++` using the most modern programming techniques available in the `C++17` standard and is built upon the `deal.II`[1] [9] FE core. The code is natively parallel and designed to run on diverse architectures, ranging from laptop computers to High Performance Computing (HPC) facilities and cloud platforms. We tested our software on a cluster node endowed with 192 cores based on Intel Xeon Gold 6238R, 2.20 GHz, available at MOX, Dipartimento di Matematica, Politecnico di Milano, and on the `GALILEO100` supercomputer available at `CINECA` (Intel CascadeLake 8260, 2.40GHz, see https://wiki.u-gov.it/confluence/display/SCAIUS/UG3.3%3A+GALILEO100+UserGuide for more technical specifications).

Despite being conceived as an academic research library in the framework of the iHEART project (see Section "Funding"), `life`^{x} is intended to provide the scientific community with a Finite Element solver for real world applications that boosts the user and developer experience without sacrificing its computational efficiency and generality.

Since its initial development, `life`^{x} has served as the core block to build several modules for the simulation of the cardiac function, such as electrophysiology, mechanics, electromechanics, blood fluid dynamics, and myocardial perfusion [8], as displayed in Fig. 1. Such models have been recently exploited for a variety of standalone or coupled simulations both under physiological and pathological conditions (see, e.g., [4, 5, 10–17]).

As the present release focuses on modeling the cardiac fibers, in the next two paragraphs we will briefly review the physiology of myofibers and the most used mathematical methods to model them, describing in detail their implementation which is included within `life`^{x}-fiber.

Cardiac fibers: physiology and modeling

The heart is a four chambers muscular organ whose function is to pump the blood throughout the whole circulatory system. The upper chambers, the right and left atria,

[1] https://www.dealii.org/.

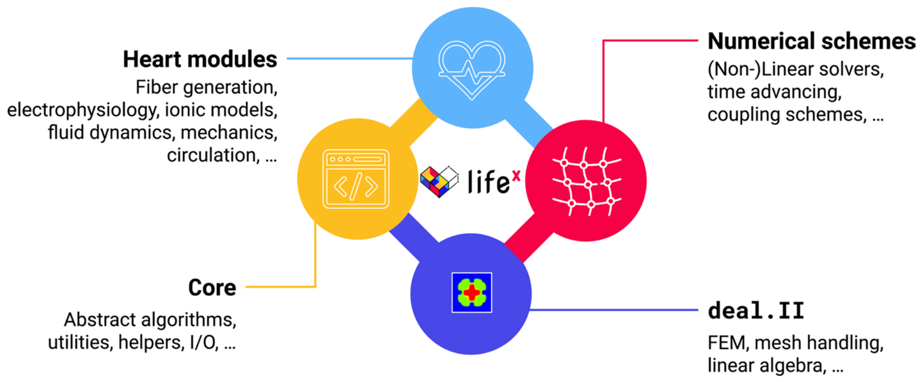
Heart modules
Fiber generation, electrophysiology, ionic models, fluid dynamics, mechanics, circulation, ...
Numerical schemes
(Non-)Linear solvers, time advancing, coupling schemes, ...
life^x
Core
Abstract algorithms, utilities, helpers, I/O, ...
deal.II
FEM, mesh handling, linear algebra, ...

Fig. 1 `life`^{x} modules: the library provides core functionalities and a framework for the numerical solution of the Finite Element problems described in Section "Laplace-Dirichlet rule-based methods". `life`^{x}-fiber is the first publicly released *heart module* based on `life`^{x}

receive incoming blood. The lower chambers, the right and left ventricles, pump blood out of the heart and are more muscular than atria. The left heart (i.e. left atrium and left ventricle) pumps the oxygenated blood through the systemic circulation, meanwhile the right heart (i.e. right atrium and right ventricle) recycles the deoxygenated blood through the pulmonary circulation [1, 2]. The atria and the ventricles are separated by the atrioventricular valves (mitral and tricuspid valves) that regulate the blood transfer from the upper to lower cavities. The four chambers are connected to the circulatory system: the ventricles with the aorta through the aortic valve and pulmonary artery via the pulmonary valve; the left atrium with the left and right pulmonary veins, whereas the right atrium with superior and inferior caval veins [1, 2].

The heart wall is made up of three layers: the internal thin *endocardium*, the external thin *epicardium* and the thick muscular cardiac tissue, the *myocardium*. Most of the myocardium is occupied by *cardiomyocytes*, striated excitable muscle cells that are joined together in linear arrays. The result of cluster cardiomyocytes, locally organized as composite laminar sheets, defines the orientation of muscular *fibers* (also called *myofibers*). Aggregations of myofibers give rise to the fiber-reinforced heart structure defining the cardiac muscular architecture [18, 19].

A schematic representation of the multiscale myocardial fiber-structure is shown in Fig. 2. Ventricular muscular fibers are well-organized as two intertwined spirals wrapping the heart around, defining the characteristic myocardial helical structure [20, 21]. Local orientation of myofibers is identified by their angle on the tangent plane and on the normal plane of the heart, called the *helical* and the *sheet* angles, respectively [18, 22]. The transition inside the myocardial wall is characterized by a continuous, almost linear change in helical angle from about 60° at the epicardium to nearly −60° at the endocardium [18, 21].

Atrial fibers architecture is very different from that of the ventricles, where myofibers are aligned in a regular pattern [18]. Indeed, myofibers in the atria are arranged in individual bundles running along different directions throughout the wall chambers [23, 24]. Preferred orientation of myofibers in the human atria is characterized by multiple overlapping structures, which promote the formation of separate attached bundles [25], as shown in Fig. 2.

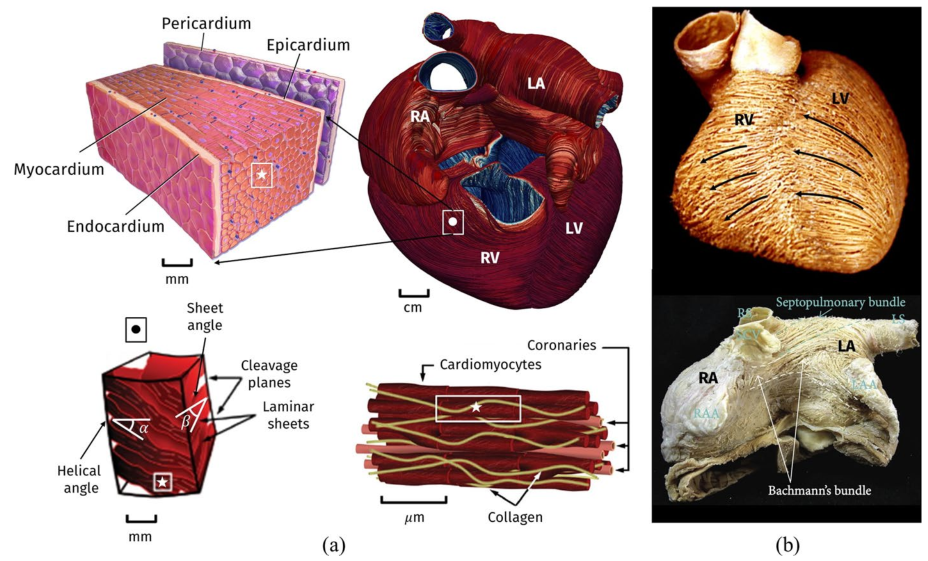

Fig. 2 **a** Representation of the multiscale cardiac muscle. **b** Anatomical dissection of myocardial fibers in ventricles (top) and atria (bottom). Images taken and readapted from [54–57]. Images were available either freely under a Creative Commons Attribution license or have been granted reuse permission by the copyright holder

The cardiac muscular fiber architecture is the backbone of a proper pumping function and has a strong influence on the electric signal propagation throughout the myocardium and also on the mechanical contraction of the muscle [26–29]. This motivates the need to accurately include fibers orientation in cardiac computational models in order to obtain physically sound results [3, 17, 30].

Due to the difficulty of reconstructing cardiac fibers from medical imaging, different methodologies have been proposed to provide a realistic surrogate of myofibers orientation [21, 31–39]. Among these, atlas-based methods map and project a detailed fiber field, previously reconstructed on an atlas, on the geometry of interest, exploiting imaging or histological data [21, 34, 39]. However, these methods require complex registration algorithms and their results depend on the original atlas data upon which they have been built.

Alternative strategies for generating myofiber orientations are the Rule-Based Methods (RBMs) [3, 32, 35–37, 40–42]. RBMs describe fiber orientations with mathematically sound rules based on histological and imaging observations and only require information about the myocardial geometry [18]. These methods parametrize the transmural and apico-basal directions in the entire myocardium in order to assign orthotropic (longitudinal, transversal and normal) myofibers [3].

A particular class of RBMs, which relies on the solution of Laplace boundary-value problems, is known as Laplace-Dirichlet Rule-Based Methods (LDRBMs), addressed in [31–33] and recently analyzed under a unified mathematical formulation [3]. LDRBMs define the transmural and apico-basal directions by taking the gradient of harmonic functions (the potentials) corresponding to suitable Dirichlet boundary conditions. These directions are then properly rotated to match histological observations [18, 20, 21, 23].

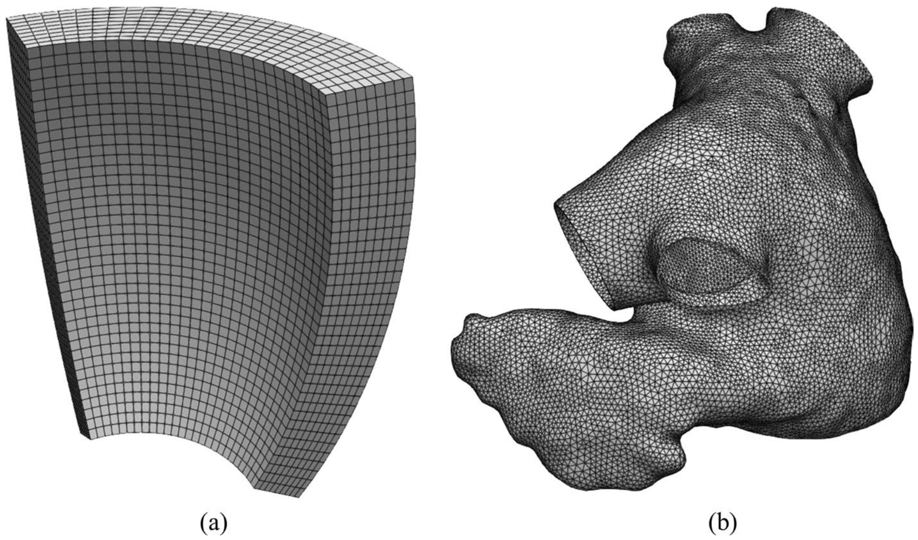

Fig. 3 **a** Hexahedral mesh of a ventricular slab. **b** Tetrahedral mesh of a realistic left atrium [38]

This initial release includes a generator for myocardial fibers based on LDRBMs [3], with application to a number of different prototypical and realistic geometries (slab models, left ventricles and left atria).

Laplace-Dirichlet rule-based methods

In this section, we briefly recall the LDRBMs that stand behind the myocardial fiber generation. In Section "Results and discussion" we will present several examples where we elaborate on how to reproduce and run the algorithms presented hereafter.

This *getting started* guide presents LDRBMs for (ventricular and spherical) slabs, (based and complete) left ventricular and left atrial geometries. For further details about the LDRBMs presented here see also [3].

The following common steps are the building blocks of all LDRBMs.

1. Labeled mesh: A labeled volumetric mesh of the domain Ω must be provided to define specific partition of the boundary $\partial\Omega$ as

$$\partial\Omega = \Gamma_{\text{epi}} \cup \Gamma_{\text{endo}} \cup \Gamma_{\text{base}} \cup \Gamma_{\text{apex}},$$

where Γ_{endo} is the endocardium, Γ_{epi} is the epicardium, Γ_{base} is the basal plane and Γ_{apex} is the apex, which are demarcated through proper surface labels included in the input mesh. `life`$^{\text{x}}$ is designed to support both hexahedral and tetrahedral labeled meshes in the widely used `*.msh` format [8], see Fig. 3. This type of mesh can be generated by a variety of mesh generation software (e.g. `gmsh`,[2] `netgen`,[3] `vmtk`[4] and `meshtools`[5]).

[2] http://gmsh.info.

[3] https://ngsolve.org/.

[4] http://www.vmtk.org.

[5] https://bitbucket.org/aneic/meshtool/src/master/.

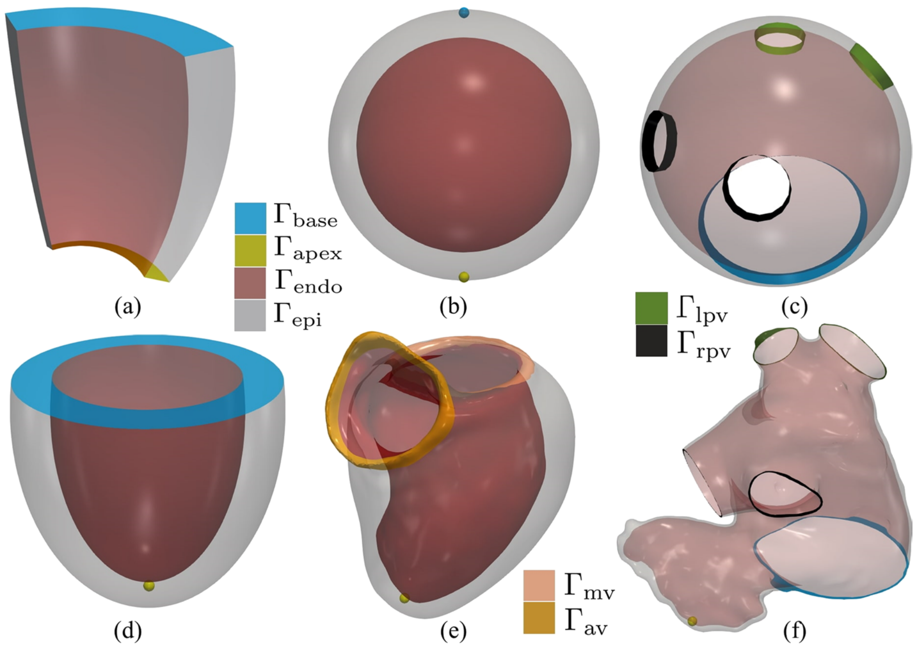

Fig. 4 Labeled meshes. **a** Ventricular slab. **b** Spherical slab. **c** Idealized left atrium. **d** Idealized based left ventricle. **e** Realistic complete left ventricle. **f** Realistic left atrium. Γ_{base} denotes the basal plane, Γ_{apex} the apex, Γ_{endo} the endocardium, Γ_{epi} the epicardium, Γ_{lpv}, Γ_{rpv} the left (right) pulmonary veins, respectively

Otherwise, other mesh-format types can be converted in `*.msh` using for example the open-source library `meshio`.[6]

Epicardium and endocardium:	for the ventricular slab geometry Γ_{endo} and Γ_{epi} are the lateral walls of the slab, see Fig. 4a; for the spherical slab, the ventricular and atrial geometries Γ_{endo} and Γ_{epi} are the internal and external surfaces, see Fig. 4b–f.
Basal plane and apex:	for the ventricular slab geometry Γ_{base} and Γ_{apex} are the top and bottom surfaces, respectively (see Fig. 4a); for the spherical slab, Γ_{base} and Γ_{apex} are selected as the north and south pole points of the epicardial sphere (see Fig. 4b); for the based ventricular geometry Γ_{base} is an artificial basal plane located well below the cardiac valves (see Fig. 4d), whereas for the complete ventricular geometry Γ_{base} is split into Γ_{mv} and Γ_{av}, representing the mitral and aortic valve rings, respectively (see Fig. 4e); for the ventricular geometries Γ_{apex} is selected as the epicardial point furthest from the ventricular base (see Fig. 4d, e); for the atrial geometry Γ_{base} is the mitral valve ring and Γ_{apex} represents the apex of the

[6] https://pypi.org/project/meshio/.

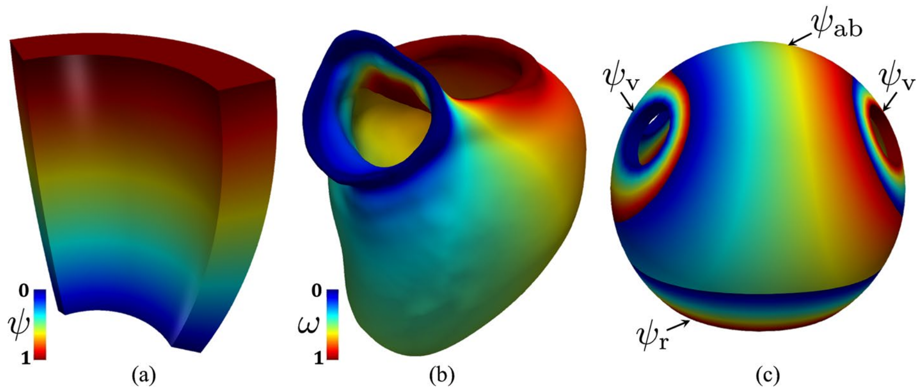

Fig. 5 Different types of normal distances: **a** Bayer-Trayanova et al. approach [32]. **b** Doste et al. approach [33]. (c) Piersanti et al. approach [3]

left atrial appendage (see Fig. 4c–f);

Atrial pulmonary rings: the atrial geometry type also requires the definition of the boundary labels for the left Γ_{lpv} and right Γ_{rpv} pulmonary vein rings, see Fig. 4c–f.

2. Transmural direction: A transmural distance ϕ is defined to compute the distance of the epicardium from the endocardium, by means of the following Laplace-Dirichlet (LD) problem:

$$\begin{cases} -\Delta\phi = 0, & \text{in } \Omega, \\ \phi = 1, & \text{on } \Gamma_{\mathrm{epi}}, \\ \phi = 0, & \text{on } \Gamma_{\mathrm{endo}}, \\ \nabla\phi \cdot \mathbf{n} = 0, & \text{on } \partial\Omega \setminus (\Gamma_{\mathrm{endo}} \cup \Gamma_{\mathrm{epi}}). \end{cases} \tag{1}$$

Then, the transmural distance gradient $\nabla\phi$ is used to build the unit transmural direction:

$$\widehat{\boldsymbol{e}}_t = \frac{\nabla\phi}{\|\nabla\phi\|}.$$

3. Normal direction: A normal (or apico-basal) direction $\boldsymbol{k}$ (which is directed from the apex towards the base) is introduced and used to build the unit normal direction $\widehat{\boldsymbol{e}}_n$:

$$\widehat{\boldsymbol{e}}_n = \frac{\boldsymbol{k} - (\boldsymbol{k} \cdot \widehat{\boldsymbol{e}}_t)\widehat{\boldsymbol{e}}_t}{\|\boldsymbol{k} - (\boldsymbol{k} \cdot \widehat{\boldsymbol{e}}_t)\widehat{\boldsymbol{e}}_t\|}.$$

The normal direction $\boldsymbol{k}$ can be computed following one of these approaches (see also Fig. 5):

Rossi-Lassila (RL) et al. approach [31]: $\boldsymbol{k}$ is defined as the vector $\mathbf{n}_{\mathrm{base}}$, i.e. the outward normal to the basal plane, that is $\boldsymbol{k} = \mathbf{n}_{\mathrm{base}}$.

Bayer-Trayanova (BT) et al. approach [32]: $\boldsymbol{k}$ is the gradient of the solution ψ ($\boldsymbol{k} = \nabla\psi$), which can be obtained by solving the following LD problem:

$$\begin{cases} -\Delta\psi = 0, & \text{in } \Omega, \\ \psi = 1, & \text{on } \Gamma_{\text{base}}, \\ \psi = 0, & \text{on } \Gamma_{\text{apex}}, \\ \nabla\psi \cdot \mathbf{n} = 0, & \text{on } \partial\Omega \setminus (\Gamma_{\text{base}} \cup \Gamma_{\text{apex}}). \end{cases} \tag{2}$$

Doste et al. approach [33]: $\boldsymbol{k}$ is a weighted sum of the apico-basal ($\nabla\psi_{\text{ab}}$) and apico-outflow-tract ($\nabla\psi_{\text{ot}}$) directions, obtained using an interpolation function w:

$$\boldsymbol{k} = w\nabla\psi_{\text{ab}} + (1 - w)\nabla\psi_{\text{ot}},$$

where ψ_{ab} and ψ_{ot} are obtained by solving LD problems in the form of (2) where $\Gamma_{\text{base}} = \Gamma_{\text{mv}}$ (for ψ_{ab}) and $\Gamma_{\text{base}} = \Gamma_{\text{av}}$ (for ψ_{ot}), respectively. Moreover, the interpolation function w is obtained by solving:

$$\begin{cases} -\Delta w = 0, & \text{in } \Omega, \\ w = 1, & \text{on } \Gamma_{\text{mv}} \cup \Gamma_{\text{apex}}, \\ w = 0, & \text{on } \Gamma_{\text{av}}, \\ \nabla w \cdot \mathbf{n} = 0, & \text{on } \partial\Omega \setminus (\Gamma_{\text{av}} \cup \Gamma_{\text{mv}} \cup \Gamma_{\text{apex}}). \end{cases}$$

Piersanti et al. approach [3]: for each point in Ω, a unique normal direction $\boldsymbol{k}$ is selected among the gradient of several normal directions $\boldsymbol{k} = \nabla\psi_{\text{i}}$ ($\text{i} = \text{ab}, \text{v}, \text{r}$), where ψ_{i} are obtained by solving the following LD problem

$$\begin{cases} -\Delta\psi_{\text{i}} = 0, & \text{in } \Omega, \\ \psi_{\text{i}} = \chi_{\text{a}}, & \text{on } \Gamma_{\text{a}}, \\ \psi_{\text{i}} = \chi_{\text{b}}, & \text{on } \Gamma_{\text{b}}, \\ \nabla\psi_i \cdot \mathbf{n} = 0, & \text{on } \partial\Omega \setminus (\Gamma_{\text{a}} \cup \Gamma_{\text{b}}). \end{cases} \tag{3}$$

Please refer to [3] for further details about the selection procedure for $\boldsymbol{k}$ and the specific choices of χ_{a}, χ_{b}, Γ_{a} and Γ_{b} in problem (2) made for ψ_{i} ($i = \text{ab}, \text{v}, \text{r}$).

The BT approach is used in the ventricular and spherical slab geometry types, see also Fig. 5a. The BT and RL approaches can be adopted in the based ventricular geometry (by setting either `Algorithm type` equal to `BT` or `RL` in the parameter file, respectively), whereas the Doste approach is used in the complete ventricular geometry, see also Fig. 5b. Finally, the Piersanti approach is employed for the atrial geometry, see also Fig. 5c.

4. Local coordinate system: For each point of the domain an orthonormal local coordinate axial system is defined by $\widehat{\boldsymbol{e}}_t$, $\widehat{\boldsymbol{e}}_n$ and the unit longitudinal direction $\widehat{\boldsymbol{e}}_l$ (orthogonal to the previous ones), as shown in Fig. 6:

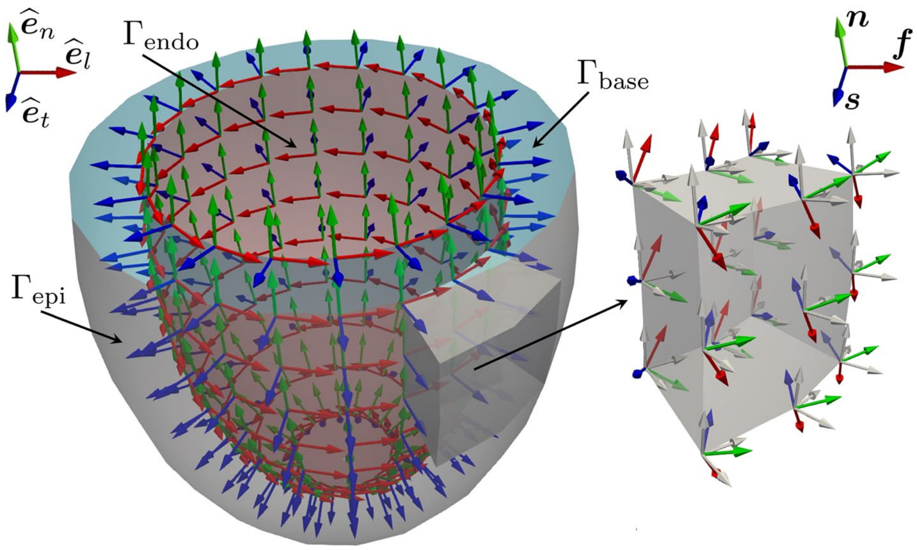

Fig. 6 Representation of the local coordinate system employed by a LDRBM for an idealized ventricular domain. Only directions on the endocardium Γ_{endo} are represented. In blue: unit transmural direction, $\widehat{\boldsymbol{e}}_t$. In green: unit normal direction, $\widehat{\boldsymbol{e}}_n$. In red: unit longitudinal direction, $\widehat{\boldsymbol{e}}_l$. Right: zoom on a slab of the left ventricular myocardium showing the three final myofibers orientations $\boldsymbol{f}$, $\boldsymbol{s}$ and $\boldsymbol{n}$

$$Q = \left[\widehat{\boldsymbol{e}}_l, \widehat{\boldsymbol{e}}_n, \widehat{\boldsymbol{e}}_t\right] = \begin{cases} \widehat{\boldsymbol{e}}_t = \dfrac{\nabla\phi}{\|\nabla\phi\|}, \\ \widehat{\boldsymbol{e}}_n = \dfrac{\boldsymbol{k} - (\boldsymbol{k}\cdot\widehat{\boldsymbol{e}}_t)\widehat{\boldsymbol{e}}_t}{\|\boldsymbol{k} - (\boldsymbol{k}\cdot\widehat{\boldsymbol{e}}_t)\widehat{\boldsymbol{e}}_t\|}, \\ \widehat{\boldsymbol{e}}_l = \widehat{\boldsymbol{e}}_n \times \widehat{\boldsymbol{e}}_t. \end{cases} \tag{4}$$

5. Axis rotation: The reference frame is rotated with the purpose of defining the myofibers orientation: $\boldsymbol{f}$ the fiber direction, $\boldsymbol{n}$ the sheet-normal direction and $\boldsymbol{s}$ the sheet direction. Specifically, $\widehat{\boldsymbol{e}}_l$ rotates counter-clockwise around $\widehat{\boldsymbol{e}}_t$ by the helical angle α, whereas the transmural direction $\widehat{\boldsymbol{e}}_t$ is rotated counter-clockwise around $\widehat{\boldsymbol{e}}_l$ by the sheetlet angle β, see Fig. 6:

$$\left[\widehat{\boldsymbol{e}}_l, \widehat{\boldsymbol{e}}_n, \widehat{\boldsymbol{e}}_t\right] \longrightarrow [\boldsymbol{f}, \boldsymbol{n}, \boldsymbol{s}],$$

The rotation angles follow the linear relationships:

$$\alpha(\phi) = \alpha_{\text{endo}}(1-\phi) + \alpha_{\text{epi}}\phi, \qquad \beta(\phi) = \beta_{\text{endo}}(1-\phi) + \beta_{\text{epi}}\phi,$$

where α_{endo}, α_{epi}, β_{endo}, β_{epi} are suitable helical and sheetlet rotation angles on the epicardium and endocardium (specifying in the parameter file `alpha epi`, `alpha endo`, `beta epi`, `beta endo`). Moreover, for the complete ventricular geometry it is possible to set specific fiber and sheet angle rotations in the outflow tract (OT) region (i.e. around the aortic valve ring) by specifying `alpha epi OT`, `alpha endo OT`, `beta epi OT`, `beta endo OT`). Finally, for the atrial geometry type, no transmural variation in the myofibers direction is prescribed and the three unit directions correspond to the final myofibers directions $[\widehat{\boldsymbol{e}}_l, \widehat{\boldsymbol{e}}_n, \widehat{\boldsymbol{e}}_t] = [\boldsymbol{f}, \boldsymbol{n}, \boldsymbol{s}]$.

In order to represent the fiber architecture, LDRBMs use the gradient of specific intra-chamber distances, by means of harmonic problems, combined with a precise definition of boundary sections where boundary conditions are prescribed. This strategy makes the fibers less open to subjective variability. On the other hand, the myofiber orientations could be adapted to a patient-specific setting by simply changing the parameters involved in LDRBMs (e.g. the helical and sheetlet angles α and β). Therefore, unlike other RBMs requiring manual or semi-automatic interventions, LDRBMs can be easily applied to any arbitrary patient-specific geometry [3].

Comparison to existing software

Several packages have been developed and are available in the framework of cardiac fibers generation.

`Meshtools`[7] [43] is a comand-line tool designed to automate image-based mesh generation and manipulate tasks in cardiac modeling workflows, such as operations on label fields and/or geometric features; it integrates seamlessly with the `openCARP`[8] ecosystem [44]; the algorithms supported are only for left ventricular geometries and of BT type. `KIT-IBT-LDRB_Fibers`[9] is a `MATLAB` tool for generating left and bi-ventricular fibers; the original BT algorithm was adapted to eliminate a discontinuity in the fiber field in correspondence of the free walls and to yield a fiber rotation that is directly proportional to the transmural Laplace solution (approximately linear across the wall) [45]. `CARDIO SUITE for GIMIAS`[10] includes tools for patient-specific modelling that allow to generate the FE meshes required for the simulations and to build additional structures such as fiber orientation [46]; at the time of writing and to the best of our knowledge, the latest version was released in 2016. `SimCardio`[11] is advertised as the only fully open-source software package providing a complete pipeline from medical image data segmentation to patient specific blood flow simulation and analysis; its module `svFSI` supports specifying distributed fiber and sheet direction generated by BT-like rule-based algorithms.

Poisson interpolation algorithms [35] have inspired the implementation of fiber generation packages, despite being generally more computationaly demanding than, e.g., RL or BT algorithms [31]. This class of methods has been implemented in `Cardiac Chaste`,[12] which supports automatic generation of mathematical model for fiber orientation associated with both idealized and anatomically-based geometry meshes [47], and in `BeatIt`,[13] which is to our knowledge the only publicly available software which natively supports the generation of fiber architectures also for atrial geometries [48].

Compared to the software described above, the strengths of `life`$^{\texttt{x}}$-fiber reside in its user-friendly interface and in its generality, by supporting either idealized and realistic, (left) ventricular and atrial geometries and for each of them the user can selected

[7] https://bitbucket.org/aneic/meshtool/src/master.

[8] https://opencarp.org/.

[9] https://github.com/KIT-IBT/LDRB_Fibers.

[10] http://www.gimias.org/index.php/clinical-prototypes/cardiosuite.

[11] http://simvascular.github.io/index.html.

[12] https://www.cs.ox.ac.uk/chaste/cardiac_index.html.

[13] https://github.com/rossisimone/beatit.

one of the different state-of-the-art algorithms described in Section "Laplace-Dirichlet rule-based methods". Finally, `life`^{x}-fiber offers a seamless integration with many other cardiac core models based on `life`^{x} (such as electrophysiology, mechanics, electromechanics, and blood fluid dynamics) for modeling the cardiac function from single-chamber to whole-heart simulations which will be targeted by future releases.

Implementation

In this section we introduce the technical specifications of `life`^{x}-fiber as well as a thorough documentation of the user interface exposed. The users will be guided from downloading it to running a full simulation of the algorithms presented in Section "Laplace-Dirichlet rule-based methods"

Technical specifications

Here we specify the package content, copyright and licensing information and software and hardware specifications required by `life`^{x}-fiber.

Package content

`life`^{x}-fiber is shipped in binary form as an `AppImage`[14] executable.

This provides a universal package for `x86-64 Linux` operating systems, without the need to deliver different distribution-specific versions. From the user's perspective, this implies an effortless *download-then-run* process, without having to manually take care of installing the proper system dependencies required.

Once the source code will be made publicly accessible, a standard *build-from-source* procedure with automatic installers will be available to make the dependencies setup tailored to the specific hardware of HPC facilities or cloud platforms.

License and third-party software

This work is copyrighted by the `life`^{x}-fiber authors and licensed under the Creative Commons Attribution Non-Commercial No-Derivatives 4.0 International License.[15]

`life`^{x}-fiber makes use of third-party libraries. Please note that such libraries are copyrighted by their respective authors (independent of `life`^{x} and `life`^{x}-fiber authors) and are covered by various permissive licenses.

The third-party software bundled with (in binary form), required by, copied, modified or explicitly used in `life`^{x}-fiber include the following packages [8]:

`life`^{x}[16]: the open-source, high-performance framework providing the core functionalities for the numerical solution of the Finite Element problems described in Section "Laplace-Dirichlet rule-based methods";

`Boost`[17]: its modules `Filesystem` and `Math` are used for manipulating files/directories and for advanced mathematical functions and interpolators, respectively;

[14] https://appimage.org/.

[15] http://creativecommons.org/licenses/by-nc-nd/4.0/.

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

[17] https://www.boost.org/.

`deal.II`[18]: it provides support to mesh handling, assembling and solving Finite Element problems (with a main support to third-party libraries as `PETSc`[19] and `Trilinos`[20] for linear algebra data structures and solvers) and to input/output functionalities;

`VTK`[21]: it is used for importing external surface or volume input data and coefficients appearing in the mathematical formulation.

Some of the packages listed above, as stated by their respective authors, rely on additional third-party dependencies that may also be bundled (in binary form) with `life`$^{\mathtt{x}}$-fiber, although not used directly. These dependencies include: `ADOL-C`,[22] `ARPACK-NG`,[23] `BLACS`,[24] `Eigen`,[25] `FFTW`,[26] `GLPK`,[27] `HDF5`,[28] `HYPRE`,[29] `METIS`,[30] `MUMPS`,[31] `NetCDF`,[32] `OpenBLAS`,[33] `ParMETIS`,[34] `ScaLAPACK`,[35] `Scotch`,[36] `SuiteSparse`,[37] `SuperLU`,[38] `oneTBB`,[39] `p4est`.[40]

The libraries listed above are all free software and, as such, they place few restrictions on their use. However, different terms may hold. Please refer to the content of the folder `doc/licenses/` for more information on license and copyright statements for these packages.

Finally, an `MPI` installation (such as `OpenMPI`[41] or `MPICH`[42]) may also be required to successfully run `life`$^{\mathtt{x}}$ executables in parallel.

[18] https://www.dealii.org/.
[19] https://www.mcs.anl.gov/petsc/.
[20] https://trilinos.github.io/.
[21] https://vtk.org/.
[22] https://github.com/coin-or/ADOL-C.
[23] https://github.com/opencollab/arpack-ng.
[24] https://www.netlib.org/blacs/.
[25] https://eigen.tuxfamily.org/.
[26] https://www.fftw.org/.
[27] https://www.gnu.org/software/glpk/.
[28] https://www.hdfgroup.org/solutions/hdf5/.
[29] https://www.llnl.gov/casc/hypre/.
[30] http://glaros.dtc.umn.edu/gkhome/metis/metis/overview.
[31] http://mumps.enseeiht.fr/index.php?page=home.
[32] https://www.unidata.ucar.edu/software/netcdf/.
[33] https://www.openblas.net/.
[34] http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview.
[35] https://www.netlib.org/scalapack/.
[36] https://gitlab.inria.fr/scotch/scotch.
[37] https://people.engr.tamu.edu/davis/suitesparse.html.
[38] https://portal.nersc.gov/project/sparse/superlu/.
[39] https://oneapi-src.github.io/oneTBB/.
[40] https://www.p4est.org/.
[41] https://www.open-mpi.org/.
[42] https://www.mpich.org/.

Software and hardware requirements

As an `AppImage`, `life`^{x}-fiber has been built on `Debian Buster` (the current `oldstable` version)[43] following the *"Build on old systems, run on newer systems"* paradigm.[44]

Therefore, it is expected to run on (virtually) any *recent enough* `x86-64 Linux` distribution, assuming that `glibc`[45] version `2.28` or higher is installed.

Quick start guide: running the `life`^{x}-fiber executable

The following steps are required in order to run the `life`^{x}-fiber executable.

Download and installation

The `life`^{x}-fiber release archive can be downloaded from https://doi.org/10.5281/zenodo.5810268. After extracting the archive, the `AppImage` file should be made executable by typing the following command in a terminal:

```
    chmod +x lifex_fiber_generation-1.4.0-x86_64.AppImage
```

Finally, `lifex_fiber_generation-1.4.0-x86_64.AppImage` can be executed with:

```
./lifex_fiber_generation-1.4.0-x86_64.AppImage [ARGS]...
```

Root permissions are **not** required. Please note that, in order for the above procedure to succeed, `AppImage` relies upon the userspace filesystem framework `FUSE`[46] which is assumed to be installed on your system. In case of errors, the following commands might be decisive:

```
./lifex_fiber_generation-1.4.0-x86_64.AppImage \
  --appimage-extract

squashfs-root/usr/bin/lifex_fiber_generation [ARGS]...
```

We also refer the reader to the `AppImage` troubleshooting guide.[47]

Step 0: parameter file configuration

Each `life`^{x} application or example, including the `life`^{x}-fiber executable described in Section "Download and installation", defines a set of parameters that are required in order to be run [8]. They involve problem-specific parameters (such as constitutive

[43] https://www.debian.org/releases/.

[44] https://docs.appimage.org/introduction/concepts.html.

[45] https://www.gnu.org/software/libc/.

[46] https://www.kernel.org/doc/html/latest/filesystems/fuse.html.

[47] https://docs.appimage.org/user-guide/troubleshooting/fuse.html.

relations, geometry, time interval, boundary conditions) as well as numerical parameters (types of linear/non-linear solvers, tolerances, maximum number of iterations) or output-related options.

In case an application has sub-dependencies (such as a linear solver), also the related parameters are included (typically in a proper subsection).

Every application comes with a set of command line options, which can be printed using the `-h` (or `-help`) flag:

```
./executable_name -h
```

The first step before running an executable is to generate the parameter file(s) containing all the default parameter values. This is done via the `-g` (or `-generate-params`) flag:

```
./executable_name -g
```

that by default generates a parameter file named after the executable, in `.prm` format.

By default, only parameters considered *standard* are printed. The parameter file verbosity can be decreased or increased by passing an optional flag `minimal` or `full` to the `-g` flag, respectively:

```
./executable_name -g minimal
```

The parameter basename to generate can be customized with the `-f` (or `-params-filename`) option:

```
./executable_name -g -f custom_param_file.ext
```

Absolute or relative paths can be specified.

At user's option, in order to guarantee a flexible interface to external file processing tools, the parameter file extension `ext` can be chosen among three different interchangeable file formats `prm`, `json` or `xml`, from the most human-readable to the most machine-readable.

As an example, the three parameter files displayed in Listings 1, 2 and 3 are semantically equivalent.

We highlight that, following with the design of the `ParameterHandler` class of `deal.II`,[48] each parameter is provided with:

[48] https://www.dealii.org/current/doxygen/deal.II/classParameterHandler.html.

- a given pattern, specifying the parameter type (e.g. boolean, integer, floating-point number, string, list, ...) and, whenever relevant, a range of admissible values (*pattern description*);
- a default value, printed in the parameter file upon generation and implicitly assumed if the user omits a custom value;
- the *actual* value, possibly overriding the default one;
- a documentation string;
- a global index.

All of these features, combined to a runtime check for the correctness of each parameter, make the code syntactically and semantically robust with respect to possible errors or typos introduced unintentionally.

```
# Listing of Parameters
# ---------------------
subsection Fiber generation
  subsection Mesh and space discretization

    # Specify whether the input mesh has hexahedral or
    # tetrahedral elements. Available options are: Hex | Tet.
    set Element type           = Hex

    # Number of global mesh refinement steps applied to the initial
    grid (Hex only).
    set Number of refinements = 0

    subsection File
      # Mesh file.
      set Filename       =

      # Mesh scaling factor: 1e-3 => from [mm] to [m].
      set Scaling factor = 1e-3
    end
  end

  subsection Output
    # Enable/disable output.
    set Enable output = true

    # Output file.
    set Filename       = fibers
  end
end
```

```
{
  "Fiber_20generation": {
    "Mesh_20and_20space_20discretization": {
      "File": {
        "Filename": {
          "value": "",
          "default_value": "",
          "documentation": "Mesh file.",
          "pattern": "0",
          "pattern_description": "[FileName (Type: input)]"
        },
        "Scaling_20factor": {
          "value": "1",
          "default_value": "1e-3",
          "documentation": "Mesh scaling factor: 1e-3 => from [mm] to
    [m].",
          "pattern": "1",
          "pattern_description": "[Double 0...MAX_DOUBLE (inclusive)]"
        }
      },
      "Element_20type": {
        "value": "Hex",
        "default_value": "Hex",
        "documentation": "Specify whether the input mesh has
    hexahedral or tetrahedral elements. Available options are: Hex |
    Tet.",
        "pattern": "2",
        "pattern_description": "[Selection Hex|Tet ]"
      },
      "Number_20of_20refinements": {
        "value": "0",
        "default_value": "0",
        "documentation": "Number of global mesh refinement steps
    applied to the initial grid (Hex only).",
        "pattern": "3",
        "pattern_description": "[Integer range 0...2147483647 (
    inclusive)]"
      }
    },
    "Output": {
      "Enable_20output": {
        "value": "true",
        "default_value": "true",
        "documentation": "Enable\/disable output.",
        "pattern": "4",
        "pattern_description": "[Bool]"
      },
      "Filename": {
        "value": "fibers",
        "default_value": "fibers",
        "documentation": "Output file.",
        "pattern": "5",
        "pattern_description": "[FileName (Type: output)]"
      }
    }
  }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<ParameterHandler>
  <Fiber_20generation>
    <Mesh_20and_20space_20discretization>
      <File>
        <Filename>
          <value/>
          <default_value/>
          <documentation>Mesh file.</documentation>
          <pattern>0</pattern>
          <pattern_description>[FileName (Type: input)]</
    pattern_description>
        </Filename>
        <Scaling_20factor>
          <value>1</value>
          <default_value>1</default_value>
          <documentation>Mesh scaling factor: 1e-3 =&gt; from [mm] to
    [m].</documentation>
          <pattern>1</pattern>
          <pattern_description>[Double 0...MAX_DOUBLE (inclusive)]</
    pattern_description>
        </Scaling_20factor>
      </File>
      <Element_20type>
        <value>Hex</value>
        <default_value>Hex</default_value>
        <documentation>Specify whether the input mesh has hexahedral
    or tetrahedral elements. Available options are: Hex | Tet.</
    documentation>
        <pattern>2</pattern>
        <pattern_description>[Selection Hex|Tet ]</pattern_description
    >
      </Element_20type>
      <Number_20of_20refinements>
        <value>0</value>
        <default_value>0</default_value>
        <documentation>Number of global mesh refinement steps applied
    to the initial grid (Hex only).</documentation>
        <pattern>3</pattern>
        <pattern_description>[Integer range 0...2147483647 (inclusive)
    ]</pattern_description>
      </Number_20of_20refinements>
    </Mesh_20and_20space_20discretization>
    <Output>
      <Enable_20output>
        <value>true</value>
        <default_value>true</default_value>
        <documentation>Enable/disable output.</documentation>
        <pattern>4</pattern>
        <pattern_description>[Bool]</pattern_description>
      </Enable_20output>
      <Filename>
        <value>fibers</value>
        <default_value>fibers</default_value>
        <documentation>Output file.</documentation>
        <pattern>5</pattern>
        <pattern_description>[FileName (Type: output)]</
    pattern_description>
      </Filename>
    </Output>
  </Fiber_20generation>
</ParameterHandler>
```

Once generated, the user can modify, copy, move or rename the parameter file depending on their needs.

Step 1: run

The executable can be run by simply omitting the `-g`, whereas the `-f` option is used to specify the parameter file to be *read* (as opposed to *written*, in generation mode), e.g.:

```
./executable_name -f custom_param_file.ext [option...]
```

If no `-f` flag is provided, a file named `executable_name.prm` is assumed to be available in the directory where the executable is run from.

The path to the directory where all the app output files will be saved to can be selected via the `-o` (or `-output-directory`) flag:

```
./executable_name -o ./results/
```

If the specified directory does not already exist, it will be created. By default, the current working directory is used.

Absolute or relative paths can be specified for both the input parameter file and the output directory.

Parallel run In order to run an app in parallel, the `mpirun` or `mpiexec` wrapper commands (which may vary depending on the MPI implementation available on your machine) should be prepended, e.g.:

```
mpirun -n <N_PROCS> ./executable_name [option...]
```

where `<N_PROCS>` is the desired number of parallel processes to run on.

As a rule of thumb, 10,000 to 100,000 degrees of freedom per process should lead to the best performance.

Dry run and parameter file conversion Upon running, a parameter log file is automatically generated in the output directory, that can be used later to retrieve which parameters had been used for a specific run.

By default, `log_params.ext` will be used as its filename. This can be changed via the `-l` (or `-log-file`) flag, e.g.:

```
./executable_name -l my_log_file.ext [option...]
```

The file extension is not mandatory: if unspecified, the same extension as the input parameter file will be used.

If the **dry run** option is enabled via the `-d` (or `-dry-run`) flag, the execution terminates right after the parameter log file generation. This has a two-fold purpose:

1 checking the correctness of the parameters being declared and parsed *before* running the actual simulation (if any of the parameters does not match the specified pattern

Table 1 Mesh information regarding the ready-to-use meshes presented in this work. In particular the mesh quality is computed selecting the edge ratio option in the `ParaView` mesh quality filter

Geometry	Type	h_{min} [mm]	h_{avg} [mm]	h_{max} [mm]	#elements	#vertices	Quality
Ventricular slab	Hex	4.74	5.95	6.73	988	1400	1.69
Ventricular slab	Tet	3.10	6.76	8.71	2439	762	2.91
Spherical slab	Tet	1.96	3.46	4.70	18455	4924	2.46
Idealized left atrium	Tet	2.78	3.43	4.15	10039	3387	2.36
Idealized left ventricle	Tet	1.71	3.53	4.93	74488	16567	3.27
Realistic left ventricle	Tet	0.51	1.17	1.67	1885227	330703	2.80
Realistic left atrium	Tet	0.15	1.00	1.76	112994	33297	4.23

or has a wrong name or has not been declared in a given subsection then a runtime exception is thrown);

2 **converting** a parameter file between two different formats/extensions. For example, the following command converts `input.xml` to `output.json`:

```
./executable_name -f input.xml -d -l output.json [option...]
```

Results and discussion

The LDRBMs described in Section "Laplace-Dirichlet rule-based methods" have been applied to a set of idealized or realistic test cases, namely ventricular and spherical slabs, based and complete left ventricular and left atrial geometries.

This section presents the results obtained, as well as a possible pipeline for reproducing such test cases, consisting of the following steps:

1 setting up input data (e.g. generating or importing computational meshes);
2 setting up the parameter files associated with a given simulation scenario and running the corresponding simulation;
3 post-processing the solution and visualizing the output.

A mesh sensitivity analysis has been performed and reported in Section "Mesh sensitivity and validation".

Input data

Additional input data (scripts, meshes and parameter files) associated with the guided examples described below can be downloaded from the release archive https://doi.org/10.5281/zenodo.5810268.

In this *getting started* guide, we provide different ready-to-use meshes, namely

- a set of four idealized geometries consisting of a ventricular slab, a spherical slab, an idealized based left ventricle and an idealized left atrium, see Fig. 4a–d;
- two realistic geometries composed by a left ventricle and a left atrium, see Fig. 4e, f.

The idealized meshes have been generated using the built-in CAD engine of `gmsh`, an open-source 3D FE mesh generator, starting from the corresponding `gmsh` geometrical models (represented by `*.geo` files, also provided) defined using their boundary representation, where a volume is bounded by a set of surfaces. For details about the geometrical definition of a model geometry we refer to the online documentation of `gmsh`.[49]

In order to perform the mesh generation, starting from the geometrical files provided in this tutorial, the following command can be run in a terminal:

```
gmsh geometry.geo -clscale s -o mesh.msh -save
```

where `geometry.geo` is the geometrical file model, `mesh.msh` is the output mesh file, which will be provided as an input to the `life`^{x}-fiber app, and `s` $\in (0, 1]$ is the mesh element size factor. To produce a coarser (finer) mesh the `clscale` factor can be reduced (increased).

The realistic left ventricle and left atrium have been produced starting from the open-source meshes adopted in [38] (for the left atrium[50]) and in [49] (for the left ventricle[51]) and using the Vascular Modelling Toolkit (`vmtk`) software [50] along with the semi-automatic meshing tools[52] recently proposed in [51].

All the characteristic informations of the ready-to-use meshes, above described, are reported in Table 1.

Generating fibers

Parameter files for fiber generation are characterized by a common section named `Mesh and space discretization`. Select `Element type = Tet` for tetrahedral meshes or `Element type = Hex` for hexahedral meshes. Finally, specify in `FE space degree` the degree of the (piecewise continuous) polynomial FE space used to solve the LD problems described above. Finally, we remark that `life`^{x}-fiber internally treats all physical quantities as if they are provided in International System of Units (SI): therefore, a `Scaling factor` can be set in order to convert the input mesh from a given unit of measurement (e.g. if the mesh coordinates are provided in millimeters then `Scaling factor` must be set equal to `1e-3`).

The `Geometry type` parameters enables to specify the kind of geometry provided in input, in order to apply the proper LDRBM algorithm among the ones described above. Once specified, parameters related to the specific algorithm and geometry will be parsed from a subsection named after the value of `Geometry type`.

All parameters missing from the parameter file will take their default value, which is hard-coded.

[49] https://gmsh.info/doc/texinfo/gmsh.html.

[50] https://doi.org/10.18742/RDM01-289.

[51] https://doi.org/10.5281/zenodo.3890034.

[52] https://github.com/marco-fedele/vmtk.

```
subsection Mesh and space discretization
  set Element type     = Tet
  set FE space degree = 1
  set Geometry type    = Slab

  subsection File
    set Filename        = /path/to/mesh/slab.msh
    set Scaling factor = 1e-3
  end
end
```

Slab fibers

The parameter `Geometry type = Slab` must be set to prescribe fibers in slab geometries and the path of the input mesh file in `Filename`.

Ventricular slab For a ventricular slab geometry the user should set `Sphere slab = false`. Labels for the top (`Tags base up`) and bottom (`Tags base down`) surfaces of the slab, as well as for the epicardium (`Tags epi`) and endocardium (`Tags endo`) must be prescribed. Finally, the helical and sheetlet rotation angles at the epicardium and endocardium must be provided in the corresponding `alpha epi`, `alpha endo`, `beta epi`, `beta endo` parameters.

```
subsection Slab
  set Sphere slab = false

  set Tags base up   =  50
  set Tags base down =  60
  set Tags epi       =  10
  set Tags endo      =  20
  set alpha epi      = -60
  set alpha endo     =  60
  set beta epi       =  45
  set beta endo      = -45
end
```

Spherical slab For a spherical slab geometry the user should set `Sphere slab = true`. The epicardial coordinates (x, y, z) of the north (`North pole`) and south (`South pole`) poles of the sphere of the slab, and the labels of the endocardium (`Tags endo`) and epicardium (`Tags epi`) must be prescribed. Finally, the helical and sheetlet rotation angles at the epicardium and endocardium must be provided in the `alpha epi`, `alpha endo`, `beta epi`, `beta endo` parameters. A fiber architecture for the sphere slab with radial fiber $\boldsymbol{f}$ can be prescribed by setting in the parameter file `Sphere with radial fibers = true`. This consists of exchanging the sheet direction $\boldsymbol{s}$ with the fiber direction $\boldsymbol{f}$. Instead, a tangential (to the epicardial and endocardial surfaces) fiber field $\boldsymbol{f}$ is assigned when `Sphere with radial fibers = false`.

```
subsection Slab
  set Sphere slab                 = true
  set Sphere with radial fibers = true

  set North pole = 0 0 0.025
  set South pole = 0 0 -0.025

  set Tags epi  = 10
  set Tags endo = 20

  set alpha epi  = 0
  set alpha endo = 0
  set beta epi   = 0
  set beta endo  = 0
end
```

Left ventricular fibers

The parameter `Geometry type = Left ventricle` (`Left ventricle complete`) prescribes fibers in a based (complete) left ventricular geometry. Other mesh-related parameters have the same meaning as described above.

```
subsection Mesh and space discretization
  set Element type    = Tet
  set FE space degree = 1
  set Geometry type   = Left ventricle complete

  subsection File
    set Filename       = /path/to/mesh/ventricle.msh
    set Scaling factor = 1e-3
  end
end
```

Based left ventricle The parameters needed are the labels for the basal plane (`Tags base`), the epicardium (`Tags epi`) and endocardium (`Tags endo`) of the ventricle. The `RL` or `BT` approach can be toggled via `Algorithm type`. The helical and sheetlet rotation angles at the epicardium and endocardium must be prescribed in `alpha epi`, `alpha endo`, `beta epi`, `beta endo`. Finally, for the `RL` approach the outward normal vector to the basal plane must be specified in `Normal to base`, whereas for the `BT` approach the apex epicardial coordinates (x, y, z) (`Apex`) of the ventricle is needed.

```
subsection Left ventricle
  set Tags base = 50
  set Tags epi  = 10
  set Tags endo = 20

  set Algorithm type = BT

  set alpha epi  = -60
  set alpha endo =  60
  set beta epi   =  20
  set beta endo  = -20

  subsection RL
    set Normal to base = 0 0 -1
  end

  subsection BT
    set Apex = 0 0 0.0601846
  end
end
```

Selecting the `RL` approach as `Algorithm type`, this setup can be also exploited in bi-ventricular geometries (not included in the example meshes). In this case, the mesh must have two additional surface labels in the right ventricular endocardium: one delimiting the part facing to the septum (e.g. 15) and the other for the remaining part (e.g. 25). In this way, it is sufficient to set `Tags endo` as the two endocardial labels (excluding the right septum) and `Tags epi` as the epicardial label and the right endocardial septum label. Further detail on this approach can be found in [3].

```
subsection Left ventricle
  ...
  set Tags epi  = 10, 15
  set Tags endo = 20, 25

  set Algorithm type = RL
  ...
end
```

Complete left ventricle The labels for the mitral (`Tags MV`) and aortic (`Tags AV`) valve rings, the epicardium (`Tags epi`) and endocardium (`Tags endo`) of the ventricle are required. The apex epicardial coordinates (x, y, z) must be set in `Apex`. Finally, the helical and sheetlet rotation angles at the epicardium and endocardium must be prescribed in `alpha epi`, `alpha endo`, `beta epi`, `beta endo`. A specific helical and sheetlet rotation angles around the outflow tract of the left ventricle (i.e. the mitral valve ring) can be specified by setting `alpha epi OT`, `alpha endo OT`, `beta epi OT`, `beta endo OT`.

```
subsection Left ventricle complete
  set Tags MV   = 50
  set Tags AV   = 60
  set Tags epi  = 10
  set Tags endo = 20

  set Apex = 0.0692 0.0710 0.3522

  set alpha epi  = -60
  set alpha endo =  60
  set beta epi   =  20
  set beta endo  = -20

  set alpha epi OT  = 0
  set alpha endo OT = 90
  set beta epi OT   = 0
  set beta endo OT  = 0
end
```

Left atrial fibers

Fibers in a left atrial geometry can be generated by setting `Geometry type = Left atrium`. Other mesh-related parameters have the same meaning as described above.

```
subsection Mesh and space discretization
  set Element type    = Tet
  set FE space degree = 1
  set Geometry type   = Left atrium

  subsection File
    set Filename       = /path/to/mesh/atrium.msh
    set Scaling factor = 1e-3
  end
end
```

Idealized left atrium The parameter `Appendage = false` prescribes fibers in the hollow sphere geometry. The labels for the mitral valve ring (`Tags MV`), the right (`Tags RPV`) and left (`Tags LPV`) pulmonary veins rings, the epicardium (`Tags epi`) and endocardium (`Tags endo`) of the idealized atrium must be provided. Finally, the dimension of each atrial bundle is needed: `Tau bundle MV` for the mitral valve bundle; `Tau bundle LPV` and `Tau bundle RPV` for the left and right pulmonary valves ring bundles.

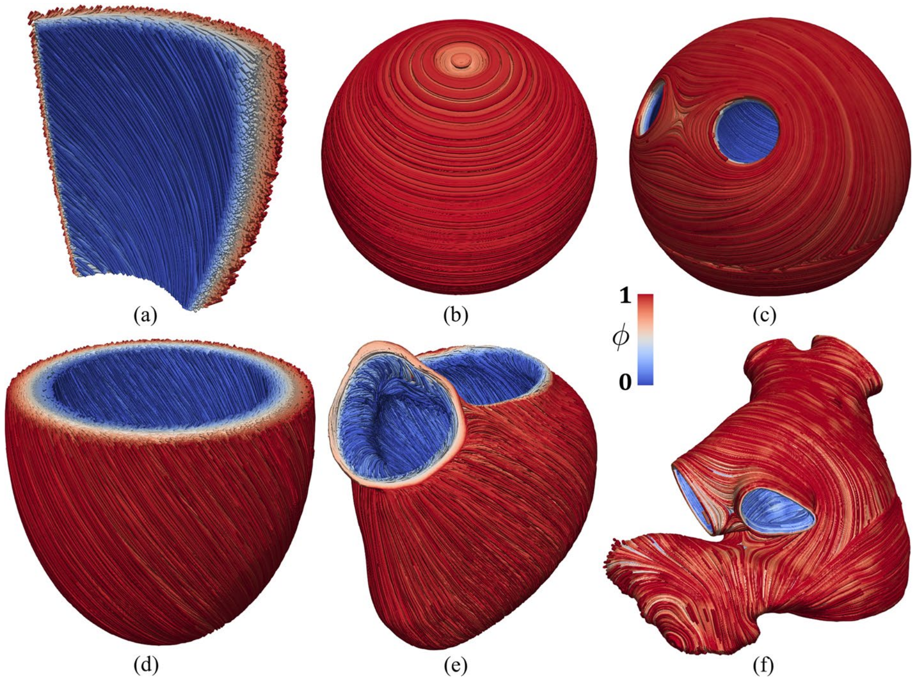

Fig. 7 Fiber field ***f*** visualized as streamlines. **a** Ventricular slab. **b** Spherical slab with circumferential fibers. **c** Idealized left atrium. **d** Idealized based left ventricle. **e** Realistic complete left ventricle. **f** Realistic left atrium

```
subsection Left atrium
  set Appendage = false

  set Tags epi  = 30
  set Tags endo = 10
  set Tags RPV  = 20
  set Tags LPV  = 50
  set Tags MV   = 40

  set Tau bundle MV  = 0.65
  set Tau bundle LPV = 0.85
  set Tau bundle RPV = 0.15
end
```

Realistic left atrium The parameter `Appendage = true` prescribes fibers in a realistic left atrial geometry. The user should provide labels for the mitral valve ring (`Tags MV`), the right (`Tags RPV`) and left (`Tags LPV`) pulmonary veins rings, the epicardium (`Tags epi`) and endocardium (`Tags endo`) of the idealized atrium. The epicardial coordinates (x, y, z) for the apex of the atrial appendage must be provided in `Apex`. Finally, the dimension of each atrial bundle is needed: for the mitral valve bundle `Tau bundle MV`; for the left and right pulmonary valves rings bundle `Tau bundle LPV` and `Tau bundle RPV`, respectively.

Table 2 Mesh sensitivity analysis for hexahedral elements

i	*h* [mm]	#dofs	$\underset{\Omega}{\text{avg}}\ \Delta_\theta^i$ [°]	$\underset{\Omega}{\max}\ \Delta_\theta^i$ [°]
1	5.95	1400	0.62	6.33
2	3.00	9477	0.24	4.54
3	1.50	69377	0.08	2.40
4	0.75	530145	–	–

Table 3 Mesh sensitivity analysis for tetrahedral elements

i	*h* [mm]	#dofs	$\underset{\Omega}{\text{avg}}\ \Delta_\theta^i$ [°]	$\underset{\Omega}{\max}\ \Delta_\theta^i$ [°]
1	6.76	762	1.35	7.85
2	3.53	3989	0.50	5.57
3	1.76	25661	0.19	4.28
4	0.88	179338	–	–

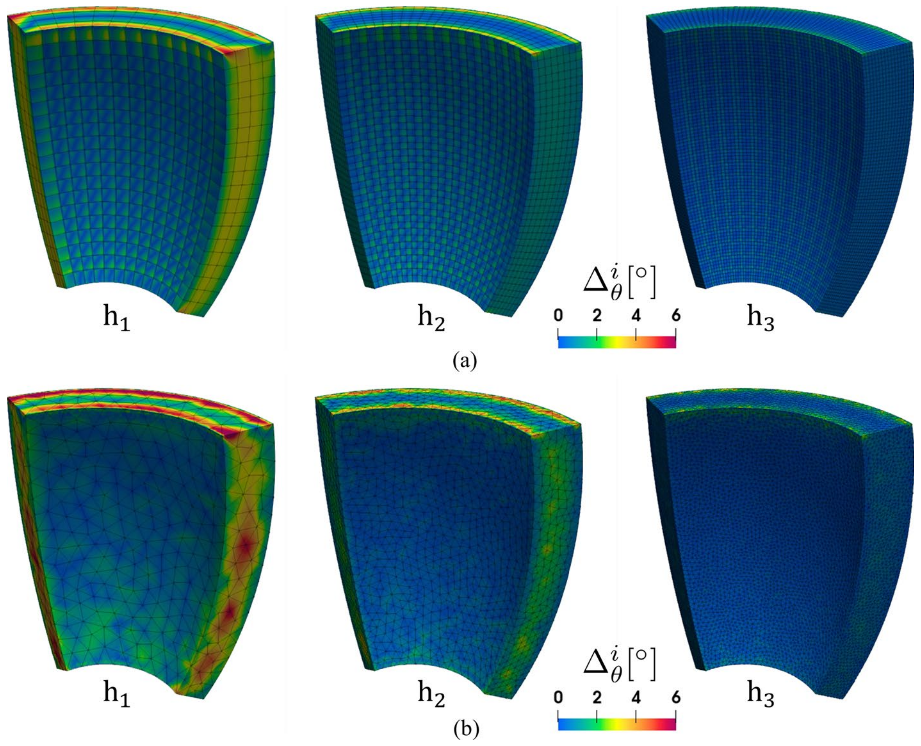

Fig. 8 Mesh sensitivity analysis performed on the ventricular slab geometry and three different mesh sizes h_1, h_2, h_3. **a** Hexahedral meshes. **b** Tetrahedral meshes

```
subsection Left atrium
  set Appendage = true

  set Apex = 83.868 16.369 45.989

  set Tags epi  = 30
  set Tags endo = 10
  set Tags RPV  = 20
  set Tags LPV  = 50
  set Tags MV   = 40

  set Tau bundle MV  = 0.60
  set Tau bundle LPV = 0.90
  set Tau bundle RPV = 0.10
end
```

Output and visualization

To enable the output the user should set `Enable output = true` and specify the corresponding output filename in `Filename`. This will produce a `XDMF` schema file named `output_filename.xdmf` (wrapped around a same-named `HDF5` output file `output_filename.h5`) that can be visualized in `ParaView`,[53] an open-source multi-platform data analysis and visualization application. Specifically, the `streamtracer` and the `tube` filters of `ParaView` can be applied in sequence to visualize the fiber fields, such as the one shown in Fig. 7.

```
subsection Output
  set Enable output = true
  set Filename      = fibers
end
```

Moreover, the `HDF5` file format guarantees that the output can easily be further post-processed, not only for visualization purposes but rather to be fed as an input to more sophisticated computational pipelines.

Mesh sensitivity and validation

The robustness of the algorithms presented above is confirmed by performing a mesh sensitivity analysis. Specifically, we consider the ventricular slab geometry shown in Fig. 4a and two related sets of discretization with hexahedral and with tetrahedral elements, respectively. For each of the two element types, we run the fiber generation algorithm (see Section "Ventricular slab") for four decreasing mesh sizes $h_1 > h_2 > h_3 > h_4$.

As the fibers field is orthonormalized, the difference between two numerical solutions is only due to the orientation angle. This allows us to define an error estimate as:

$$\Delta_\theta^i = \left|\arccos\left(\mathbf{f}_0^i \cdot \mathbf{f}_0^4\right)\right|, \; i = 1, 2, 3,$$

[53] https://www.paraview.org.

where $\mathbf{f}_0^i$ is the fiber field computed on the mesh with size h_i and the results on the finest mesh with size h_4 are considered as a reference solution.

We show the error distribution in the whole domain in Fig. 8, whereas Tables 2 and 3 report the average and maximum errors for the hexahedral and the tetrahedral meshes, respectively. We determine both an average and a maximum error smaller than 8° even for the coarsest mesh. These values are very small and, in particular, much smaller than the physiological fibers dispersion angle [52].

The interest of accurately reconstructing a fiber orientation field consists of providing it as an input to more sophisticated computational models, such as for cardiac electrophysiology and electromechanics. As the dynamics of such physical models demands for a high resolution in both time and space [3, 17, 53], the LDRBM algorithms are typically run on fine meshes. Therefore, the impact of possible numerical errors due to the mesh size is to be considered negligible, as confirmed by the small errors presented above.

The proposed LDRBMs have been validated in [3], where the fiber orientation field computed numerically provided a satisfactory match to histological data.

Furthermore, the anisotropic nature of the fiber orientation strongly influences the electrophysiological, mechanical, and electromechanical cardiac function. Therefore, an indirect way to validate the fiber field provided by LDRBMs is to compute quantitative indices or biomarkers in such kind of simulations.

Matching quantitative indices in a physiological range is only possible when the fiber reconstruction algorithm is accurate enough: this is shown in [3] for the whole-heart electrophysiology, in [17] for the ventricular electromechanics, and in [53] for the whole-heart electromechanics.

Future developments

As anticipated in Section ”Background”, `life`$^\mathrm{x}$ has served as the core framework for the development of several *heart modules* for the simulation of cardiac electrophysiology, mechanics, electromechanics, and blood fluid dynamics models.

In the near future, the deployment of `life`$^\mathrm{x}$ modules will follow two lines:

- more modules will be successively published in binary form, starting from an advanced solver for cardiac electrophysiology, blood fluid dynamics and other solvers for the cardiac function;
- in the meantime, the source code associated with previous binary releases will be gradually made publicly available under an open-source license.

In the long run, also modules unrelated to the cardiac function are expected to be included within `life`$^\mathrm{x}$.

Conclusions

`life`$^{\mathrm{x}}$ is intended to provide the scientific community with an integrated FE framework for exploring many physiological and pathological scenarios using *in silico* experiments for the whole-heart cardiac function, boosting both the user and developer experience without sacrificing its computational efficiency and universality.

We believe that the release of `life`$^{\mathrm{x}}$-fiber provides the scientific community with an invaluable tool for *in silico* scenario analyses of myofibers orientation; such a tool supports either idealized and realistic, (left) ventricular and atrial geometries. It also offers a seamless integration of LDRBMs into more sophisticated computational pipelines involving other core models – such as electrophysiology, mechanics and electromechanics – for the cardiac function, in a wide range of settings covering from single-chamber to whole-heart simulations.

The content of this initial release is published on the official website https://lifex.gitlab.io/heart.html: we encourage users to interact with the `life`$^{\mathrm{x}}$ development community via the `issue tracker`[54] of our public website repository. Any curiosity, question, bug report or suggestion is welcome.

News and announcements about `life`$^{\mathrm{x}}$ will be posted to the official website https://lifex.gitlab.io/.

Availability and requirements

Project name: `life`$^{\mathrm{x}}$-fiber
Project home page: https://lifex.gitlab.io/heart.html
Operating system(s): `Linux (x86-64)`
Programming language: `C++`
Other requirements: `glibc` version `2.28` or higher
License: CC BY-NC-ND 4.0
Any restrictions to use by non-academics: no additional restriction.

Abbreviations

FE	Finite element
BT	Bayer-Trayanova
RL	Rossi-Lassila
HPC	High performance computing
LD	Laplace-Dirichlet
RBM	Rule-based method
LDRBM	Laplace-Dirichlet rule-based method
SI	International system of units

Acknowledgements
Not applicable.

Author contributions
The initial design of `life`$^{\mathrm{x}}$ was conceived by PCA with precious hints and contributions by MF. The mathematical models and the Laplace-Dirichlet Rule-Based Methods were developed by RP. The software development and testing were carried out by PCA, MF and RP. PCA and RP wrote the paper. LD and AQ supervised the model development and the interpretation of results. All authors read and approved the final manuscript.

[54] https://gitlab.com/lifex/lifex.gitlab.io/-/issues.

Funding
This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 740132, iHEART - An Integrated Heart Model for the simulation of the cardiac function, P.I. Prof. A. Quarteroni).

Availability of data and materials
All input data, meshes and the binary executable of `life`^{x} can be found at https://doi.org/10.5281/zenodo.5810268.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interest
The authors declare that they have no competing interests.

Received: 3 March 2022 Accepted: 27 March 2023
Published online: 12 April 2023

References
1. Quarteroni A, Dede' L, Manzoni A, Vergara C. Mathematical modelling of the human cardiovascular system: data, numerical approximation, clinical applications. Cambridge Monographs on Applied and Computational Mathematics, 2019; https://doi.org/10.1017/9781108616096
2. Quarteroni A, Lassila T, Rossi S, Ruiz-Baier R. Integrated heart–coupling multiscale and multiphysics models for the simulation of the cardiac function. Comput Methods Appl Mech Eng 2017;314:345–407. https://doi.org/10.1016/j.cma.2016.05.031. Special Issue on Biological Systems Dedicated to William S. Klug
3. Piersanti R, Africa PC, Fedele M, Vergara C, Dede' L, Corno AF, Quarteroni A. Modeling cardiac muscle fibers in ventricular and atrial electrophysiology simulations. Comput Methods Appl Mech Eng. 2021;373:113468.
4. Salvador M, Fedele M, Africa PC, Sung E, Dede' L, Prakosa A, Chrispin J, Trayanova N, Quarteroni A. Electromechanical modeling of human ventricles with ischemic cardiomyopathy: numerical simulations in sinus rhythm and under arrhythmia. Comput Biol Med. 2021;136:104674. https://doi.org/10.1016/j.compbiomed.2021.104674.
5. Salvador M, Regazzoni F, Pagani S, Dede' L, Trayanova N, Quarteroni A. The role of mechano-electric feedbacks and hemodynamic coupling in scar-related ventricular tachycardia. Comput Biol Med, 2022;105203.
6. ten Tusscher KHWJ, Panfilov AV. Alternans and spiral breakup in a human ventricular tissue model. Am J Physiol Heart Circul Physiol. 2006;291(3):1088–100. https://doi.org/10.1152/ajpheart.00109.2006. (**PMID: 16565318**).
7. Courtemanche M, Ramirez RJ, Nattel S. Ionic mechanisms underlying human atrial action potential properties: insights from a mathematical model. Am J Physiol Heart Circul Physiol. 1998;275(1):301–21. https://doi.org/10.1152/ajpheart.1998.275.1.H301. (**PMID: 9688927**).
8. Africa PC. `life`^{x}: A flexible, high performance library for the numerical solution of complex finite element problems. SoftwareX 2022;20:101252. https://doi.org/10.1016/j.softx.2022.101252
9. Arndt D, Bangerth W, Blais B, Fehling M, Gassmöller R, Heister T, Heltai L, Köcher U, Kronbichler M, Maier M, Munch P, Pelteret JP, Proell S, Konrad S, Turcksin B, Wells D, Zhang J. The `deal.II` library, version 9.3. J Numer Math. 2021;29(3):171–86. https://doi.org/10.1515/jnma-2021-0081.
10. Regazzoni F, Salvador M, Africa PC, Fedele M, Dede' L, Quarteroni A. A cardiac electromechanical model coupled with a lumped-parameter model for closed-loop blood circulation. J Comput Phys, 2022;111083.
11. Regazzoni F, Quarteroni A. Accelerating the convergence to a limit cycle in 3d cardiac electromechanical simulations through a data-driven 0d emulator. Comput Biol Med. 2021;135: 104641. https://doi.org/10.1016/j.compbiomed.2021.104641.
12. Zingaro A, Fumagalli I, Fedele M, Africa PC, Dede' L, Quarteroni A, Corno AF. A geometric multiscale model for the numerical simulation of blood flow in the human left heart. Discrete Continuous Dyn Syst S. 2022;15(8):2391–427. https://doi.org/10.3934/dcdss.2022052.
13. Bucelli M, Dede' L, Quarteroni A, Vergara C. Partitioned and monolithic algorithms for the numerical solution of cardiac fluid-structure interaction. Commun Comput Phys. 2023;32(5):1217–56. https://doi.org/10.4208/cicp.OA-2021-0243.
14. Fumagalli I, Vitullo P, Scrofani R, Vergara C. Image-based computational hemodynamics analysis of systolic obstruction in hypertrophic cardiomyopathy. medRxiv 2021. https://doi.org/10.1101/2021.06.02.21258207
15. Stella S, Vergara C, Maines M, Catanzariti D, Africa PC, Demattè C, Centonze M, Nobile F, Del Greco M, Quarteroni A. Integration of activation maps of epicardial veins in computational cardiac electrophysiology. Comput Biol Med. 2020;127:104047. https://doi.org/10.1016/j.compbiomed.2020.104047.
16. Dede' L, Regazzoni F, Vergara C, Zunino P, Guglielmo M, Scrofani R, Fusini L, Cogliati C, Pontone G, Quarteroni A. Modeling the cardiac response to hemodynamic changes associated with covid-19: a computational study. Math Biosci Eng. 2021;18(4):3364–83.
17. Piersanti R, Regazzoni F, Salvador M, Corno AF, Dede' L, Vergara C, Quarteroni A. 3d–0d closed-loop model for the simulation of cardiac biventricular electromechanics. Comput Methods Appl Mech Eng. 2022;391: 114607. https://doi.org/10.1016/j.cma.2022.114607.

18. Streeter DD Jr, Spotnitz HM, Patel DP, Ross J Jr, Sonnenblick EH. Fiber orientation in the canine left ventricle during diastole and systole. Circ Res. 1969;24(3):339–47.
19. LeGrice IJ, Smaill BH, Chai LZ, Edgar SG, Gavin JB, Hunter PJ. Laminar structure of the heart: ventricular myocyte arrangement and connective tissue architecture in the dog. Am J Physiol Heart Circul Physiol. 1995;269(2):571–82.
20. Greenbaum RA, Ho SY, Gibson DG, Becker AE, Anderson RH. Left ventricular fibre architecture in man. Heart. 1981;45(3):248–63.
21. Lombaert H, Peyrat J, Croisille P, Rapacchi S, Fanton L, Cheriet F, Clarysse P, Magnin I, Delingette H, Ayache N. Human atlas of the cardiac fiber architecture: study on a healthy population. IEEE Trans Med Imaging. 2012;31(7):1436–47.
22. Toussaint N, Stoeck CT, Schaeffter T, Kozerke S, Sermesant M, Batchelor PG. In vivo human cardiac fibre architecture estimation using shape-based diffusion tensor processing. Med Image Anal. 2013;17(8):1243–55.
23. Ho SY, Anderson RH, Sánchez-Quintana D. Atrial structure and fibres: morphologic bases of atrial conduction. Cardiovasc Res. 2002;54:325–36.
24. Ho SY, Sánchez-Quintana D. The importance of atrial structure and fibers. Clin Anatomy Off J Am Assoc Clin Anatom Br Assoc Clin Anatom. 2009;22:52–63.
25. Sánchez-Quintana D, Pizarro G, López-Mínguez JR, Ho SY, Cabrera JA. Standardized review of atrial anatomy for cardiac electrophysiologists. J Cardiovasc Transl Res. 2013;6:124–44.
26. Roberts DE, Hersh LT, Scher AM. Influence of cardiac fiber orientation on wavefront voltage, conduction velocity, and tissue resistivity in the dog. Circ Res. 1979;44(5):701–12.
27. Punske BB, Taccardi B, Steadman B, Ershler PR, England A, Valencik ML, McDonald JA, Litwin SE. Effect of fiber orientation on propagation: electrical mapping of genetically altered mouse hearts. J Electrocardiol. 2005;38(4):40–4.
28. Eriksson TSE, Prassl AJ, Plank G, Holzapfel GA. Influence of myocardial fiber/sheet orientations on left ventricular mechanical contraction. Math Mech Solids. 2013;18(6):592–606.
29. Palit A, Bhudia SK, Arvanitis TN, Turley GA, Williams MA. Computational modelling of left-ventricular diastolic mechanics: Effect of fibre orientation and right-ventricle topology. J Biomech. 2015;48(4):604–12.
30. Guan D, Yao J, Luo X, Gao H. Effect of myofibre architecture on ventricular pump function by using a neonatal porcine heart model: from DT-MRI to rule-based methods. R Soc Open Sci. 2020;7(4): 191655.
31. Rossi S, Lassila T, Ruiz-Baier R, Sequeira A, Quarteroni A. Thermodynamically consistent orthotropic activation model capturing ventricular systolic wall thickening in cardiac electromechanics. Eur J Mech A Solids. 2014;48:129–42.
32. Bayer JD, Blake RC, Plank G, Trayanova N. A novel rule-based algorithm for assigning myocardial fiber orientation to computational heart models. Ann Biomed Eng. 2012;40(10):2243–54.
33. Doste R, Soto-Iglesias D, Bernardino G, Alcaine A, Sebastian R, Giffard-Roisin S, Sermesant M, Berruezo A, Sanchez-Quintana D, Camara O. A rule-based method to model myocardial fiber orientation in cardiac biventricular geometries with outflow tracts. Int J Numer Methods Biomed Eng. 2019;35(4):3185.
34. Hoermann JM, Pfaller MR, Avena L, Bertoglio C, Wall WA. Automatic mapping of atrial fiber orientations for patient-specific modeling of cardiac electromechanics using image registration. Int J Numer Methods Biomed Eng. 2019;35(6):3190.
35. Wong J, Kuhl E. Generating fibre orientation maps in human heart models using poisson interpolation. Comput Methods Biomech Biomed Eng. 2014;17(11):1217–26.
36. Krueger MW, Schmidt V, Tobón C, Weber FM, Lorenz C, Keller DUJ, Barschdorf H, Burdumy M, Neher P, Plank G *et al.* Modeling atrial fiber orientation in patient-specific geometries: a semi-automatic rule-based approach. In: International conference on functional imaging and modeling of the heart, 2011;223–232.
37. Ferrer A, Sebastián R, Sánchez-Quintana D, Rodríguez JF, Godoy EJ, Martínez L, Saiz J. Detailed anatomical and electrophysiological models of human atria and torso for the simulation of atrial activation. PLoS ONE. 2015;10(11):0141573.
38. Fastl TE, Tobon-Gomez C, Crozier A, Whitaker J, Rajani R, McCarthy KP, Sanchez-Quintana D, Ho SY, O'Neill MD, Plank G et al. Personalized computational modeling of left atrial geometry and transmural myofiber architecture. Med Image Anal 2018.
39. Roney CH, Bendikas R, Pashakhanloo F, Corrado C, Vigmond EJ, McVeigh ER, Trayanova NA, Niederer SA. Constructing a human atrial fibre atlas. Ann Biomed Eng 2020.
40. Beyar R, Sideman S. A computer study of the left ventricular performance based on fiber structure, sarcomere dynamics, and transmural electrical propagation velocity. Circ Res. 1984;55(3):358–75.
41. Potse M, Dubé B, Richer J, Vinet A, Gulrajani RM. A comparison of monodomain and bidomain reaction-diffusion models for action potential propagation in the human heart. IEEE Trans Biomed Eng. 2006;53(12):2425–35.
42. Nielsen P, Le Grice IJ, Smaill BH, Hunter PJ. Mathematical model of geometry and fibrous structure of the heart. Am J Physiol Heart Circul Physiol. 1991;260(4):1365–78.
43. Neic A, Gsell MAF, Karabelas E, Prassl AJ, Plank G. Automating image-based mesh generation and manipulation tasks in cardiac modeling workflows using meshtool. SoftwareX. 2020;11: 100454. https://doi.org/10.1016/j.softx.2020.100454.
44. Plank G, Loewe A, Neic A, Augustin C, Huang Y-L, Gsell MAF, Elias Karabelas JS, Nothstein M, Prassl AJ, Seemann G, Vigmond EJ. The openCARP simulation environment for cardiac electrophysiology. Comput Methods Programs Biomed. 2021;208:106223. https://doi.org/10.1016/j.cmpb.2021.106223.
45. Kovacheva E, Gerach T, Schuler S, Ochs M, Dössel O, Loewe A. Causes of altered ventricular mechanics in hypertrophic cardiomyopathy: an in-silico study. Biomed Eng Online. 2021;20(1):1–28.
46. Larrabide I, Omedas P, Martelli Y, Planes X, Nieber M, Moya JA, Butakoff C, Sebastián R, Camara O, Craene MD *et al.* Gimias: an open source framework for efficient development of research tools and clinical prototypes. In: International conference on functional imaging and modeling of the heart, 2009;417–426. Springer
47. Cooper FR, Baker RE, Bernabeu MO, Bordas R, Bowler L, Bueno-Orovio A, Byrne HM, Carapella V, Cardone-Noott L, Cooper J et al. Chaste: cancer, heart and soft tissue environment. J Open Source Softw 2020.
48. Rossi S, Gaeta S, Griffith BE, Henriquez CS. Muscle thickness and curvature influence atrial conduction velocities. Front Physiol, 1344, 2018.

49. Strocchi M, Augustin CM, Gsell MAF, Karabelas E, Neic A, Gillette K, Razeghi O, Prassl AJ, Vigmond EJ, Behar JM, Gould J, Sidhu B, Rinaldi CA, Bishop MJ, Plank G, Niederer SA. A publicly available virtual cohort of four-chamber heart meshes for cardiac electro-mechanics simulations. PLoS ONE. 2020;15:1–26. https://doi.org/10.1371/journal.pone.0235145.
50. Antiga L, Steinman DA. The vascular modeling toolkit 2008.
51. Fedele M, Quarteroni A. Polygonal surface processing and mesh generation tools for the numerical simulation of the cardiac function. Int J Numer Methods Biomed Eng. 2021;37(4):3435.
52. Guan D, Zhuan X, Holmes W, Luo X, Gao H. Modelling of fibre dispersion and its effects on cardiac mechanics from diastole to systole. J Eng Math. 2021;128(1):1–24.
53. Fedele M, Piersanti R, Regazzoni F, Salvador M, Africa PC, Bucelli M, Zingaro A, Dede' L, Quarteroni A. A comprehensive and biophysically detailed computational model of the whole human heart electromechanics. arXiv 2022. https://doi.org/10.48550/ARXIV.2207.12460
54. Chabiniok R, Wang VY, Hadjicharalambous M, Asner L, Lee J, Sermesant M, Kuhl E, Young AA, Moireau P, Nash MP, et al. Multiphysics and multiscale modelling, data-model fusion and integration of organ physiology in the clinic: ventricular cardiac mechanics. Interface Focus. 2016;6(2):20150083.
55. Anderson RH, Niederer PF, Sanchez-Quintana D, Stephenson RS, Agger P. How are the cardiomyocytes aggregated together within the walls of the left ventricular cone? J Anat. 2019;235(4):697–705.
56. Sánchez-Quintana D, López-Mínguez JR, Macías Y, Cabrera JA, Saremi F. Left atrial anatomy relevant to catheter ablation. Cardiol Res Practice 2014;2014.
57. Blausen.com staff: Medical gallery of Blausen Medical 2014. WikiJournal of Medicine (2014). https://en.wikiversity.org/wiki/WikiJournal_of_Medicine/Medical_gallery_of_Blausen_Medical_2014

Publisher's Note