MOX-Report No. 102/2024

The lifex library version 2.0

Bucelli, M.

MOX, Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

mox-dmat@polimi.it https://mox.polimi.it

The lifex library version 2.0

Michele Bucelli[a,*]

[a]*MOX Laboratory of Modeling and Scientific Computing, Dipartimento di Matematica, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133, Milano, Italy*

Abstract

This article presents updates to `life`$^{\texttt{x}}$ [Africa, SoftwareX (2022)], a `C++` library for high-performance finite element simulations of multiphysics, multiscale and multidomain problems. In this release, we introduce an additional intergrid transfer method for non-matching multiphysics coupling on the same domain, significantly optimize nearest-neighbor point searches and interface coupling utilities, extend the support for 2D and mixed-dimensional problems, and provide improved facilities for input/output and simulation serialization and restart. These advancements also propagate to the previously released modules of `life`$^{\texttt{x}}$ specifically designed for cardiac modeling and simulation, namely `life`$^{\texttt{x}}$`-fiber` [Africa et al., BMC Bioinformatics (2023)], `life`$^{\texttt{x}}$`-ep` [Africa et al., BMC Bioinformatics (2023)] and `life`$^{\texttt{x}}$`-cfd` [Africa et al., Computer Physics Communications (2024)]. The changes introduced in this release aim at consolidating `life`$^{\texttt{x}}$'s position as a valuable and versatile tool for the simulation of multiphysics systems.

Keywords: High performance computing, Finite elements, Numerical simulations, Multiphysics problems

1. Introduction

This paper discusses recent updates to `life`$^{\texttt{x}}$ [1] (`https://lifex.gitlab.io`), a `C++` library tailored at finite element simulations for multiphysics and multiscale problems (logo depicted in Figure 1). `life`$^{\texttt{x}}$ builds upon the finite element library `deal.II` [5, 6] by implementing high-level reusable utilities for common tasks such as generating or reading meshes, both tetrahedral and hexahedral, solving linear and non-linear systems of equations, preconditioning linear systems, writing output to file, or managing simulation checkpointing and restart. Many of these features are implemented by wrapping `deal.II` functionality in easy-to-use software interfaces. These are configured through human-readable and well structured parameter files in the custom `deal.II` syntax [1], so that `life`$^{\texttt{x}}$ can be used effectively to design simulation tools that require minimal coding effort from the end user, if any [2–4].

The effectiveness of the framework offered by `life`$^{\texttt{x}}$ was demonstrated by three modules specifically targeted at applications to cardiac and cardiovascular modeling: `life`$^{\texttt{x}}$`-fiber` [3], for muscle fiber generation; `life`$^{\texttt{x}}$`-ep` [4], for simulating electrophysiology; `life`$^{\texttt{x}}$`-cfd` [2], for computational fluid dynamics (CFD) simulations. The updates presented in this paper concern the general-purpose features of the library, and as such also apply to those modules.

`life`$^{\texttt{x}}$ provides the basis for a large and growing number of recent application studies [7–9, 14–23, 25–27, 30–32, 34, 36–40], which testify to the versatility and impact of the `life`$^{\texttt{x}}$ ecosystem on cardiovascular research. An up-to-date list of publications using `life`$^{\texttt{x}}$ is maintained on the dedicated page of the official website[1].

The present release aims at consolidating, optimizing and enhancing the features of `life`$^{\texttt{x}}$'s core module. We implement a new method for transferring data between non-matching meshes [12, 13] in parallel simulations (Section 2); we improve the interface and the performance of the parallel nearest-neighbor lookup of

*Corresponding author. E-mail: michele.bucelli@polimi.it

[1] `https://lifex.gitlab.io/lifex-public/publications.html`

Figure 1: The `life`^{x} logo. Image licensed under the CC-BY-SA 4.0 license.

`Geometry`	`Numerics`	`Multiphysics`	`IO`
`MeshHandler`	`NonLinearSolverHandler`	`QuadratureEvaluation`	`CSVWriter`
`MeshInfo`	`LinearSolverHandler`	`ProjectionL2`	`VTKFunction`
`DoFLocator (*)`	`PreconditionerHandler`	`InterfaceHandler`	**`OutputHandler (*)`**
`BoundaryDoFLocator (*)`	`BlockPreconditionerHandler`	**`RBFInterpolation (*)`**	**`VTKImporter (*)`**
`GeodesicDistance (*)`	`BDFHandler`		**`RestartHandler (*)`**
	`BCHandler`		
	`FixedPointAcceleration`		
	`TimeInterpolation`		
	`Laplace (*)`		

Table 1: List of `life`^{x} core's most relevant classes, grouped by category. Classes that were introduced in the new release are written in bold and marked with an asterisk `(*)`. Additional details can be found in the online documentation (`https://lifex.gitlab.io/lifex-public/index.html`).

mesh vertices (Section 3), an operation used throughout the library for many tasks, including interface coupling; we introduce support for simulations in dimensions other than 3D (Section 4); we enhance simulation checkpointing and restart through a significantly improved and standardized user interface (Section 5), and introduce several improvements to the input/output (IO) facilities in general (Section 6). All these features greatly enhance the usability and versatility of `life`^{x}, with the aim of further improving its effectiveness as a tool for multiphysics simulations.

Table 1 reports the most relevant classes implemented in `life`^{x}, grouped by category. The rest of this paper presents the features in the new release and their significance, while we refer to [1] for a more extensive description of the general `life`^{x} framework and of the features available in previous release, and to the source code repository[2] and the online documentation[3] for technical details. Unless otherwise specified, all `C++` classes discussed in this document are part of the `lifex::utils` namespace, which we omit henceforth in the interest of brevity.

2. Radial basis function interpolation for multiphysics coupling

`life`^{x} is designed with multiphysics cardiac applications in mind, and as such has a strong focus on the coupling of heterogeneous models. Until previous release, the transfer of data between different domains or different models always imposed some conformity constraints on their discretization. Interface-coupled problems can be managed through the class `InterfaceHandler` and the related functionality, requiring that the two coupled models have a conforming discretization at their interface [11]. Data transfer between volume-coupled problems (that is, problems defined on the same domain) can be managed through the class `QuadratureEvaluation` and its derived classes, which require that the two models share the same mesh, although they can be discretized with different finite element spaces [1]. The coupling of meshes of different resolution was, until now, only supported for nested grids of hexahedral elements [33].

With this release, we add support for radial basis function (RBF) interpolation between non-matching meshes, with the methods described in [12, 13, 24]. This allows to transfer data between spatial discretizations

[2] `https://gitlab.com/lifex/lifex-public`
[3] `https://lifex.gitlab.io/lifex-public/index.html`

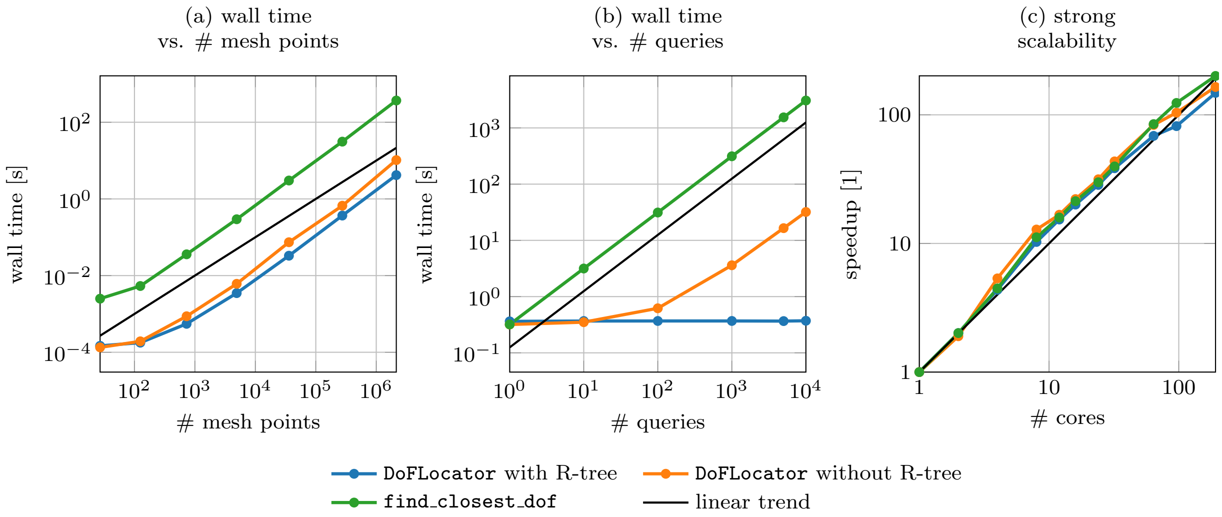


Figure 2: Performance of the new `DoFLocator` class compared to the old `find_closest_dof` function. (a) Wall time against the number of mesh points, for 100 repeated queries, using one parallel process. `DoFLocator`, both with and without R-trees, is close to 100 times faster than the old function. (b) Computational time against number of queries, for a mesh with 274 625 vertices, using one parallel process. The advantage of R-trees becomes evident as the number of queries increases. (c) Nearly ideal parallel speedup of a strong scalability study, with 100 repeated queries on a mesh with 2 146 689 vertices.

of arbitrary refinement, element shape (tetrahedral or hexahedral) and polynomial degree, thus greatly enhancing the library's flexibility in coupling heterogeneous problems. Furthermore, we support complex geometries with dedicated methods based on approximate geodesic distance [13]. This implementation of RBF interpolation demonstrates excellent parallel scalability up to thousands of cores, as discussed in [12, 13].

The new features are exposed through the class `RBFInterpolation`, providing an interface for RBF interpolation between arbitrary sets of points. Derived classes `RBFInterpolationDoFs` and `RBFInterpolationQuadrature` manage interpolation of data that is collocated at degrees of freedom (DoFs) and mesh quadrature points, respectively. All these classes can be configured extensively through the parameter file. A new example named `ExampleRBFInterpolation` showcases the new features.

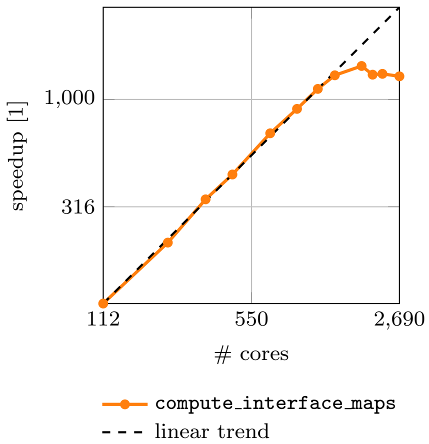
3. Point locators

The release introduces new helper classes `DoFLocator` and `BoundaryDoFLocator` that offer an interface for the task of locating the nearest DoF (or boundary DoF) to a given point in the physical space, possibly in a parallel setting, which is a key part of many algorithms in `life`^{x}.

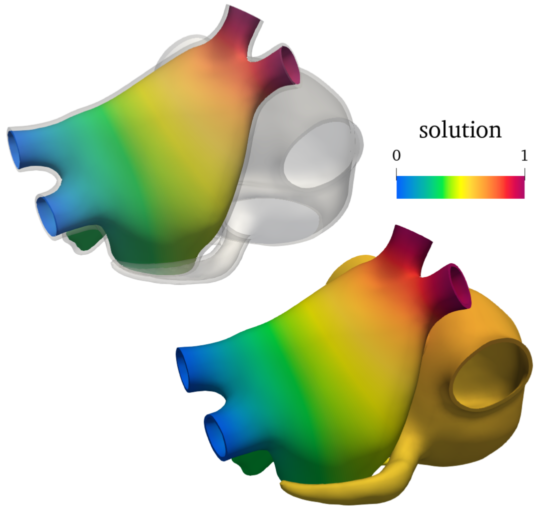
The locator classes internally build an R-tree representation of the points [29], using the implementation of R-trees from `boost::geometry::index` [10] wrapped by `deal.II`. Therefore, they provide a friendly yet computationally efficient interface for nearest-neighbor searches. A dedicated method is implemented for efficient multi-point queries, where every parallel process needs to locate a different set of points, possibly owned by other processes.

We stress that, until previous release, nearest-neighbor searches were done with a simple linear search algorithm, implemented by the `find_closest_dof` function (now removed). The current implementation significantly improves in terms of algorithmic complexity and performance, as shown in Figure 2, and can take advantage from repeated queries by reusing the same R-tree.

Additionally, the new locator classes are exploited in the coupling of domains across a common interface, as implemented by `InterfaceMap` and `InterfaceHandler`. The efficiency and parallel performance of the construction of interface maps has been significantly enhanced in this release. As depicted in Figure 3a, the task of establishing a map between the interface DoFs of two domains with a common boundary (implemented



(a)



(b)

Figure 3: (a) Strong scalability of the function `compute_interface_maps`, that establishes a mapping between the interface DoFs of two domains with a common boundary. This test couples a surface and a volume problem, with 1 049 601 and 135 398 529 DoFs, respectively, and 1 049 601 interface DoFs. (b) Example of coupling a surface and a volume problem through their common interface, as implemented by the new example `InterfaceCoupling2D3D`. We first solve a Laplace-Beltrami problem on a portion of the domain's boundary (top), and then solve a Laplace problem in the whole domain (bottom), with Dirichlet conditions taken from the surface problem. Atrial model taken from [28].

by the function `compute_interface_maps`) shows ideal parallel scalability up to approximately 800 interface DoFs per process.

4. 1D, 2D and mixed-dimensional problems

With this release, we improve $\texttt{life}^{\texttt{x}}$'s support for 1D and 2D problems, and mixed-dimensional problems in general. The user can now specify the spatial dimension through the CMake `LIFEX_DIM` parameter (which defaults to 3). Many classes of general purpose are now templated over the physical and spatial dimensions, following the same convention as `deal.II`. These include `MeshHandler`, `MeshInfo`, `OutputHandlerBase` and its derivatives, `Laplace` and `GeodesicDistance`.

Most notably, with this improvement we extend the applicability of $\texttt{life}^{\texttt{x}}$ utilities to problems defined on surfaces (as seen in the new `LaplaceBeltramiExample`). Additionally, the `InterfaceHandler`-related utilities now allow to couple problems of mixed dimensions, such as a problem defined on a surface with another defined on a volume for which that surface is part of the boundary. This feature is demonstrated by the new example `InterfaceCoupling2D3D` (Figure 3b).

5. Checkpointing and restart

High-performance computing (HPC) enviroments typically limit the maximum duration of a job to wall-times that are shorter than the duration of large-scale simulations. It is therefore crucial for a library such as $\texttt{life}^{\texttt{x}}$ to allow splitting computations over multiple jobs, overcoming these limitations.

To this end, $\texttt{life}^{\texttt{x}}$ allows to write the simulation state to a file (a process also referred to as *checkpointing*, or *serialization*), from which, at a later time, the simulation itself can be restarted. In this release, we significantly reworked this process, improving its robustness and reliability, enhancing its support for multiphysics simulations, and significantly simplifying its user interface.

All this is implemented through a new helper class named `RestartHandler`. The class can collect data from different problems, allowing to store multiple fields or scalar values in a single `.h5` file [35]. We remark that the previous release would create multiple files for each model, which could lead to confusion and clutter, and would not allow to include scalars in the serialized files, so that they would need to be saved and restored separately.

Conversely, the `RestartHandler` class offers a clean interface to store and retrieve all the data needed for restart. The class acts by keeping a list of references to data that needs to be serialized or deserialized. Such data can be easily registered through the methods `RestartHandler::attach_scalar` and `RestartHandler::attach_vector`, for scalar types and parallel vectors (or block vectors), respectively. A helper `RestartHandler::attach_bdf_handler` facilitates serialization and restart for time-dependent problems relying on the `BDFHandler` class [1]. Serialization is done by calling the `RestartHandler::serialize` method, while restart is performed through the `RestartHandler::restart` method. Simulation and restart for multiple models can be easily centralized by having each model write to the same instance of `RestartHandler`.

All `life`^{x} tutorials have been extended to exemplify the use of the `RestartHandler` class. Most notably, tutorial 6 demonstrates its use in the context of a multiphysics simulation in which the different sub-models are managed by separate classes.

From the user's perspective, checkpointing and restart are configured in two dedicated subsections of the parameter file:

```
subsection Serialization
  set Enable                      = true
  set Serialization basename      = restart
  set Serialize every n timesteps = 1000
end

subsection Restart
  set Enable                 = true
  set Restart basename       = out_dir/restart
  set Restart timestep index = 1000
end
```

We stress that, differently from previous release, the user need not specify the initial time or initial timestep number of the restarting simulation, as these will be retrieved from the serialized data. Overall, this makes the process of restarting much simpler and less error-prone.

6. Input/output enhancements

On top of the previously discussed `RestartHandler`, we introduced new classes to centralize IO tasks that are common between multiple applications of `life`^{x}. This has a twofold purpose: on the one hand, it enforces a standardized interface for those tasks, ensuring in particular that all applications share the same parameter file structure. On the other hand, this centralization greatly facilitates any future extension.

6.1. Output of problem solutions to file

Data output has been centralized to the new class `OutputHandler`, wrapping `deal.II`'s data writer class `dealii::DataOut`. With respect to previous release, we exposed output in `.pvtu/.vtu` format (on top of the already available `.xdmf/.h5` format). Indeed, we have observed that parallel output to `.h5` files may occasionally lead to deadlocks due to issues with parallel filesystems. The `.pvtu/vtu` format, where each process writes its data in an independent file, offers an effective workaround in those situations. We point out that `.pvtu/vtu` output usually occupies more disk space than `.xdmf/.h5` output, due to the latter allowing to filter out duplicate internal vertices.

6.2. Reading data from VTK files

Many applications are based on reading functional data to be used as parameters for numerical models implemented in `life`^{x} [15, 30, 31]. A new class `VTKImporter` facilitates reading and remapping data from the well-established VTK file formats (`.vtk`, `.vtp` and `.vtu`), and optionally serializing the imported data to a binary file for later reuse. The class supports all the types of VTK functions offered by `VTKFunction` and `VTKPreprocess` [1] (linear projections, closest-point projections and signed distance evaluation), but additionally takes care of standardizing the parameter file sections that configure these operations.

6.3. *Fixed memory occupation peak when reading meshes*

Until previous release, when reading tetrahedral meshes from file, all parallel processes would read the mesh in its entirety, and discard the portions attributed to other processes only after partitioning. This would lead to a very high peak memory occupation, often higher than the available memory, thus frequently resulting in the simulation being killed during initialization. We introduce a new parameter `Reading group size` to the `MeshHandler` class, which allows to reduce the number of processes that read the entire mesh (based on `create_description_from_triangulation_in_groups` from `deal.II`'s `TriangulationDescription::Utilities` namespace). This has proven crucial in supporting very large-scale simulations.

7. Additional improvements

In addition to multiple bufgixes and performance improvements, the new release includes the following changes:

- `life`$^{\texttt{x}}$ is now updated to use `deal.II` version 9.5.1;
- a new helper class `Laplace` provides a simple interface for solving Laplace and Laplace-Beltrami problems (that is, $-\Delta u = 0$ in a certain domain Ω). The class is meant to be used for algorithms that require solving the Laplace equation as an intermediate step, such as the ones discussed in [3];
- users can specify a custom set of default parameters to each instance of `PreconditionerHandler`. This is particularly useful since the optimal preconditioner configuration may vary between different problems: such a configuration can be built into the source code for each problem, without requiring the user to manually adjust the parameter file;
- a new class `GeodesicDistance` allows to compute an edge-based approximation of the geodesic distance and of the shortest path between points within a domain, exposed through the new app `shortest_path`.

8. Conclusions

The `life`$^{\texttt{x}}$ library offers a comprehensive set of tools to facilitate the development of multiphysics finite element simulations. The 2.0 release described in this paper extends the library's functionality, by improving its multiphysics coupling capabilities and its support for simulations of different dimensionalities. Furthermore, it improves the efficiency and parallel scalability of fundamental tasks such as parallel point location and communication between interface-coupled problems. Finally, the new release provides an improved and standardized interface for several tasks related to input and output. All these changes significantly enhance the capabilities of `life`$^{\texttt{x}}$ and its applicability to large-scale problems, and consolidate its position as a valuable framework for simulating multiphysics systems.

Acknowledgements

`life`$^{\texttt{x}}$ was developed under the scientific supervision of Profs. Luca Dede' (Politecnico di Milano, Milano, Italy) and Alfio Quarteroni (Politecnico di Milano, Milano, Italy). The author wishes to thank all the `life`$^{\texttt{x}}$ users and developers for their software contributions, testing and feedback, and especially Pasquale Claudio Africa (SISSA, Trieste, Italy) for his years as `life`$^{\texttt{x}}$ maintainer and his invaluable support. The `life`$^{\texttt{x}}$ logo was designed by Silvia Pozzi (Artiness).

The present research is part of the activities of "Dipartimento di Eccellenza 2023–2027", MUR, Italy, Dipartimento di Matematica, Politecnico di Milano. The author has received support from the project PRIN2022, MUR, Italy, 2023–2025, 202232A8AN "Computational modeling of the heart: from efficient numerical solvers to cardiac digital twins". The author acknowledges his membership to INdAM GNCS - Gruppo Nazionale per il Calcolo Scientifico (National Group for Scientific Computing, Italy), and INdAM GNCS project CUP E53C23001670001. The author acknowledges ISCRA for awarding this project access to the LEONARDO supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CINECA (Italy).

References

[1] P. C. Africa. lifex: A flexible, high performance library for the numerical solution of complex finite element problems. *SoftwareX*, 20:101252, 2022.

[2] P. C. Africa, I. Fumagalli, M. Bucelli, A. Zingaro, M. Fedele, L. Dede', and A. Quarteroni. lifex-cfd: An open-source computational fluid dynamics solver for cardiovascular applications. *Computer Physics Communications*, 296:109039, 2024.

[3] P. C. Africa, R. Piersanti, M. Fedele, L. Dede', and A. Quarteroni. lifex-fiber: an open tool for myofibers generation in cardiac computational models. *BMC Bioinformatics*, 24(1):143, 2023.

[4] P. C. Africa, R. Piersanti, F. Regazzoni, M. Bucelli, M. Salvador, M. Fedele, S. Pagani, L. Dede', and A. Quarteroni. lifex-ep: a robust and efficient software for cardiac electrophysiology simulations. *BMC Bioinformatics*, 24(1):389, 2023.

[5] D. Arndt, W. Bangerth, M. Bergbauer, M. Feder, M. Fehling, J. Heinz, et al. The `deal.II` library, version 9.5. *Journal of Numerical Mathematics*, 31(3):231–246, 2023.

[6] D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, et al. The deal.II finite element library: Design, features, and insights. *Computers & Mathematics with Applications*, 81:407–422, 2021.

[7] N. Barnafi, F. Regazzoni, and D. Riccobelli. Reconstructing relaxed configurations in elastic bodies: Mathematical formulations and numerical methods for cardiac modeling. *Computer Methods in Applied Mechanics and Engineering*, 423:116845, 2024.

[8] L. Bennati, V. Giambruno, F. Renzi, V. Di Nicola, C. Maffeis, G. Puppini, G. B. Luciani, and C. Vergara. Turbulent blood dynamics in the left heart in the presence of mitral regurgitation: a computational study based on multi-series cine-MRI. *Biomechanics and Modeling in Mechanobiology*, 22(6):1829–1846, 2023.

[9] L. Bennati, G. Puppini, V. Giambruno, G. B. Luciani, and C. Vergara. Image-based computational fluid dynamics to compare two repair techniques for mitral valve prolapse. *Annals of Biomedical Engineering*, 2024.

[10] Boost. Boost C++ Libraries. http://www.boost.org/, 2024. Last accessed November 13, 2024.

[11] M. Bucelli, L. Dede', A. Quarteroni, and C. Vergara. Partitioned and monolithic algorithms for the numerical solution of cardiac fluid-structure interaction. *Communications in Computational Physics*, 32(5):1217–1256, 2023.

[12] M. Bucelli, F. Regazzoni, L. Dede', and A. Quarteroni. Preserving the positivity of the deformation gradient determinant in intergrid interpolation by combining RBFs and SVD: Application to cardiac electromechanics. *Computer Methods in Applied Mechanics and Engineering*, 417:116292, 2023.

[13] M. Bucelli, F. Regazzoni, L. Dede', and A. Quarteroni. Robust radial basis function interpolation based on geodesic distance for the numerical coupling of multiphysics problems. *SIAM Journal on Scientific Computing (in press)*, 2024.

[14] M. Bucelli, A. Zingaro, P. C. Africa, I. Fumagalli, L. Dede', and A. Quarteroni. A mathematical model that integrates cardiac electrophysiology, mechanics, and fluid dynamics: Application to the human left heart. *International Journal for Numerical Methods in Biomedical Engineering*, 39(3):e3678, 2023.

[15] E. Capuano, F. Regazzoni, M. Maines, S. Fornara, V. Locatelli, D. Catanzariti, S. Stella, F. Nobile, M. Del Greco, and C. Vergara. Personalized computational electro-mechanics simulations to optimize cardiac resynchronization therapy. *Biomechanics and Modeling in Mechanobiology*, 2024.

[16] E. Centofanti and S. Scacchi. A comparison of algebraic multigrid bidomain solvers on hybrid CPU–GPU architectures. *Computer Methods in Applied Mechanics and Engineering*, 423:116875, 2024.

[17] L. Cicci, S. Fresca, A. Manzoni, and A. Quarteroni. Efficient approximation of cardiac mechanics through reduced-order modeling with deep learning-based operator approximation. *International Journal for Numerical Methods in Biomedical Engineering*, 40(1):e3783, 2024.

[18] A. Corda, S. Pagani, and C. Vergara. Influence of acute myocardial ischemia on arrhythmogenesis: a computational study. *medRxiv*, pages 2024–11, 2024.

[19] E. Criseo, I. Fumagalli, A. Quarteroni, S. M. Marianeschi, and C. Vergara. Computational haemodynamics for pulmonary valve replacement by means of a reduced fluid-structure interaction model. *International Journal for Numerical Methods in Biomedical Engineering*, 40(9):e3846, 2024.

[20] A. Crispino, L. Bennati, and C. Vergara. Cardiac hemodynamics computational modeling including chordae tendineae, papillaries, and valves dynamics. *bioRxiv preprint*, 2024.

[21] L. Crugnola and C. Vergara. Inexact block lu preconditioners for incompressible fluids with flow rate conditions. *arXiv preprint arXiv:2411.03929*, 2024.

[22] L. Crugnola, C. Vergara, L. Fusini, I. Fumagalli, G. Luraghi, A. Redaelli, and G. Pontone. Computational hemodynamic indices to identify transcatheter aortic valve implantation degeneration. *Computer Methods and Programs in Biomedicine*, 2024.

[23] G. R. de Souza, M. J. Grote, S. Pezzuto, and R. Krause. Explicit stabilized multirate methods for the monodomain model in cardiac electrophysiology. *arXiv preprint arXiv:2401.01745*, 2024.

[24] S. Deparis, D. Forti, and A. Quarteroni. A rescaled localized radial basis function interpolation on non-cartesian and nonconforming grids. *SIAM Journal on Scientific Computing*, 36(6):A2745–A2762, 2014.

[25] F. Duca, D. Bissacco, L. Crugnola, C. Faitini, M. Domanin, F. Migliavacca, S. Trimarchi, and C. Vergara. Computational analysis to assess hemodynamic forces in descending thoracic aortic aneurysms. *bioRxiv*, pages 2024–11, 2024.

[26] M. Falanga, C. Cortesi, A. Chiaravalloti, A. Dal Monte, C. Tomasi, and C. Corsi. A digital twin approach for stroke risk assessment in atrial fibrillation patients. *Heliyon*, 10(20), 2024.

[27] M. Fedele, R. Piersanti, F. Regazzoni, M. Salvador, P. C. Africa, M. Bucelli, A. Zingaro, L. Dede', and A. Quarteroni. A comprehensive and biophysically detailed computational model of the whole human heart electromechanics. *Computer Methods in Applied Mechanics and Engineering*, 410:115983, 2023.

[28] A. Ferrer, R. Sebastián, D. Sánchez-Quintana, J. F. Rodriguez, E. J. Godoy, L. Martinez, and J. Saiz. Detailed anatomical and electrophysiological models of human atria and torso for the simulation of atrial activation. *PloS one*, 10(11):e0141573, 2015.

[29] Y. Manolopoulos, A. N. Papadopoulos, and Y. Theodoridis. *R-Trees: Theory and Applications*. Springer Science & Business Media, 2006.

[30] G. Montino Pelagi, A. Baggiano, F. Regazzoni, L. Fusini, M. Alì, G. Pontone, et al. Personalized pressure conditions and calibration for a predictive computational model of coronary and myocardial blood flow. *Annals of Biomedical Engineering*, 52(5):1297–1312, 2024.

[31] G. Montino Pelagi, F. Regazzoni, J. M. Huyghe, A. Baggiano, M. Alì, S. Bertoluzza, G. Valbusa, G. Pontone, and C. Vergara. Modeling cardiac microcirculation for the simulation of coronary flow and 3D myocardial perfusion. *Biomechanics and Modeling in Mechanobiology*, pages 1–26, 2024.

[32] R. Piersanti, R. Bradley, S. Y. Alid, A. Quarteroni, L. Dede, and N. A. Trayanova. Defining myocardial fiber bundle architecture in atrial digital twins. *arXiv preprint arXiv:2410.11601*, 2024.

[33] F. Regazzoni, M. Salvador, P. C. Africa, M. Fedele, L. Dedè, and A. Quarteroni. A cardiac electromechanical model coupled with a lumped-parameter model for closed-loop blood circulation. *Journal of Computational Physics*, 457:111083, 2022.

[34] F. Renzi, G. Puppini, G. B. Luciani, and C. Vergara. Investigating the right heart hemodynamics in the Tetralogy of Fallot: a computational study. *bioRxiv preprint*, 2024.

[35] The HDF Group. Hierarchical Data Format, version 5.

[36] E. Zappon, A. Manzoni, and A. Quarteroni. A non-conforming-in-space numerical framework for realistic cardiac electrophysiological outputs. *Journal of Computational Physics*, 502:112815, 2024.

[37] E. Zappon, M. Salvador, R. Piersanti, L. Regazzoni, Francesco Dede', and A. Quarteroni. An integrated heart–torso electromechanical model for the simulation of electrophysiological outputs accounting for myocardial deformation. *Computer Methods in Applied Mechanics and Engineering*, 427:117077, 2024.

[38] A. Zingaro, Z. Ahmad, E. Kholmovski, K. Sakata, L. Dede', A. K. Morris, A. Quarteroni, and N. A. Trayanova. A comprehensive stroke risk assessment by combining atrial computational fluid dynamics simulations and functional patient data. *Scientific Reports*, 14(1):9515, 2024.

[39] A. Zingaro, M. Bucelli, R. Piersanti, F. Regazzoni, L. Dede', and A. Quarteroni. An electromechanics-driven fluid dynamics model for the simulation of the whole human heart. *Journal of Computational Physics*, 504:112885, 2024.

[40] A. Zingaro, C. Vergara, L. Dede', F. Regazzoni, and A. Quarteroni. A comprehensive mathematical model for cardiac perfusion. *Scientific Reports*, 13(1):14220, 2023.

MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

101/2024 Bonetti, S.; Corti, M.
Unified discontinuous Galerkin analysis of a thermo/poro-viscoelasticity model

100/2024 Farenga, N.; Fresca, S.; Brivio, S.; Manzoni, A.
On latent dynamics learning in nonlinear reduced order modeling

99/2024 Ragni, A.; Masci, C.; Paganoni, A. M.
Analysis of Higher Education Dropouts Dynamics through Multilevel Functional Decomposition of Recurrent Events in Counting Processes

98/2024 Castiglione, C.; Arnone, E.; Bernardi, M.; Farcomeni, A.; Sangalli, L.M.
PDE-regularised spatial quantile regression

97/2024 Ferro, N.; Mezzadri, F.; Carbonaro, D.; Galligani, E.; Gallo, D.; Morbiducci, U.; Chiastra, C.; Perotto, S.
Designing novel vascular stents with enhanced mechanical behavior through topology optimization of existing devices

95/2024 Zacchei, F.; Rizzini, F.; Gattere, G.; Frangi, A.; Manzoni, A.
Neural networks based surrogate modeling for efficient uncertainty quantification and calibration of MEMS accelerometers

93/2024 Conti, P.; Kneifl, J.; Manzoni, A.; Frangi, A.; Fehr, J.; Brunton, S.L.; Kutz, J.N.
VENI, VINDy, VICI - a variational reduced-order modeling framework with uncertainty quantification

96/2024 Brivio, S.; Fresca, S.; Manzoni, A.
PTPI-DL-ROMs: Pre-trained physics-informed deep learning-based reduced order models for nonlinear parametrized PDEs

94/2024 Franco, N.R.; Fresca, S.; Tombari, F.; Manzoni, A.
Deep Learning-based surrogate models for parametrized PDEs: handling geometric variability through graph neural networks

91/2024 Ciaramella, G.; Kartmann, M.; Mueller, G.
Solving Semi-Linear Elliptic Optimal Control Problems with L1-Cost via Regularization and RAS-Preconditioned Newton Methods