POLITECNICO
MILANO 1863

# Deep learning-based reduced order models for nonlinear parametrized PDEs: application to cardiac electrophysiology

Advisor:
**Prof. Alfio Quarteroni**

Coadvisor:
**Prof. Andrea Manzoni**
**Prof. Luca Dedè**

Chair of the Doctoral Program:
**Prof. Irene Sabadini**

Doctoral dissertation of:
**Stefania Fresca**

Politecnico di Milano
MOX – Dipartimento di Matematica
Mathematical Models and Methods in Engineering – XXXIII cycle

*A voi due,*
*Cinzia e Simone*

# Sommario

Ottenere in modo rapido e accurato la soluzione di un problema differenziale dipendente da parametri è lo scopo delle tecniche di riduzione computazionale, o modelli di ordine ridotto (ROMs). Per fare ciò, le caratteristiche essenziali del comportamento della soluzione di un problema vengono descritte mediante la soluzione di un problema ridotto, le cui dimensioni sono notevolmente inferiori a quelle di un problema discretizzato con un classico metodo di tipo high-fidelity (o full order), come ad esempio quelli ottenuti mediante il metodo di Galerkin-elementi finiti o con l'Analisi Isogeometrica. Tali tecniche di discretizzazione conducono a un sistema di grandi dimensioni da risolvere per raggiungere una determinata accuratezza e risultano dunque impraticabili ogni qualvolta occorra calcolare tale soluzione ripetutamente, al variare di un insieme di parametri di ingresso.

L'idea che costituisce il nucleo delle tecniche di riduzione computazionale, come i metodi a basi ridotte, è l'assunzione - spesso verificata nella realtà - che il comportamento di un sistema anche complesso possa essere ben descritto da una combinazione lineare di un numero esiguo di modi dominanti, come nel caso della proper orthogonal decomposition (POD). Tali modelli, tuttavia, possono rivelarsi inefficienti nella risoluzione di equazioni a derivate parziali (PDEs) non lineari tempo-dipendenti parametrizzate, in particolare nel caso di problemi caratterizzati da strutture coerenti che si propagano nel tempo, come accade per l'elettrofisiologia (EP) cardiaca sia in scenari fisiologici che patologici. Questi sistemi difficilmente possono essere ricondotti a problemi a bassa dimensionalità la cui soluzione risulti sufficientemente accurata mediante ROMs convenzionali come, ad esempio, i metodi a basi ridotte per PDEs parametrizzate (POD-Galerkin ROMs). Ciò è principalmente dovuto alla grande variabilità che caratterizza l'insieme delle soluzioni al variare dei parametri del problema e alla natura non lineare della mappa input-output che si intende ricostruire numericamente. Puntando all'efficienza dei modelli di ordine ridotto, in questa Tesi viene proposta una nuova generazione di ROMs non intrusivi e non lineari, basati su algoritmi di deep learning (DL), come le reti neurali feedforward, convoluzionali e autoencoder.

In tali modelli di ordine ridotto, o DL-ROMs, sia lo spazio in cui cercare le candidate soluzioni, che la dinamica ridotta, vengono descritti mediante reti neurali, e costruiti in modo non intrusivo attraverso algoritmi di DL. Tali algoritmi prevedono l'allenamento delle reti su un insieme di snapshots, ovvero di soluzioni ottenute mediante il modello ad alta fedeltà per differenti valori dei parametri e diversi istanti di tempo. I risultati numerici mostrano che la tecnica DL-ROM permette di catturare accuratamente complessi processi di propagazione di fronti nell'ambito dell'EP fisiologica e patologica. Inoltre, effettuando una riduzione dimensionale preliminare sull'insieme di snapshots, tramite randomized POD, e utilizzando un opportuno addestramento preliminare delle reti (o pretraining) è possibile accelerare i tempi di addestramento delle reti, e dunque di costruzione dei modelli ridotti, oltre che ridurre notevolmente la complessità della rete. L'accuratezza e l'efficienza di questa strategia, che prende il nome di POD DL-ROM, sono state valutate su numerosi problemi a derivate parziali parametrizzati (comprese le equazioni dell'elastodinamica non lineare per la meccanica dei solidi e le equazioni di Navier-Stokes non stazionarie per la fluidodinamica). L'interesse primario di questa Tesi è rivolto all'EP cardiaca, il cui scopo è la descrizione dell'attività bioelettrica del cuore e delle sue disfunzioni. Considerando sia casi test prototipali che geometrie realistiche, condizioni fisiologiche e patologiche come la tachicardia e la fibrillazione atriale, oltre che tecniche ad alta fedeltà differenti (quali ad esempio il metodo degli elementi finiti o l'Analisi Isogeometrica), le tecniche introdotte in questa Tesi hanno dimostrato di poter risolvere tali problemi, una volta addestrate le reti neurali, per ciascun nuovo scenario in tempo reale, anche in contesti complessi quali la descrizione dei fenomeni di rientro e di rottura dei fronti, che modellizzano i principali meccanismi scatenanti delle aritmie cardiache.

# Abstract

Conventional reduced order models (ROMs) relying on the assumption of modal linear superimposition, such as proper orthogonal decomposition (POD), might reveal inefficient when dealing with nonlinear time-dependent parametrized PDEs, especially for problems featuring coherent structures propagating over time, such as physiological and pathological cardiac electrophysiology (EP). Indeed, these systems can hardly be reduced to accurate but lower dimensional problems by conventional ROMs such as, e.g., POD-Galerkin ROMs in the framework of reduced basis methods for parametrized PDEs. This is primarily due to the strong variability characterizing the solution manifold (with respect to the problem parameters) as well as to the nonlinear nature of the input-output maps that we intend to reconstruct numerically. To enhance ROM efficiency, we propose a new generation of non-intrusive, nonlinear ROMs, based on deep learning (DL) algorithms, such as convolutional, feedforward and autoencoder neural networks. In the proposed DL-ROM, both the nonlinear trial manifold and the nonlinear reduced dynamics are learnt in a non-intrusive way by relying on DL algorithms trained on a set of full order model (FOM) snapshots, obtained for different parameter values at different time instants. Numerical results show that the resulting DL-ROM technique allows to accurately capture complex fronts propagation processes, both in physiological and pathological scenarios in cardiac EP. Performing then a prior dimensionality reduction on FOM snapshots through randomized POD, and using a suitable multi-fidelity pretraining, enable, when dealing with large-scale FOMs, to speed up training times, and decrease the network complexity, substantially. Accuracy and efficiency of this strategy, which we refer to as POD DL-ROM, are assessed on different parametrized PDE problems (including nonlinear structural mechanics and fluid dynamics). The main focus of this Thesis is cardiac EP, that is the description of the electrical activity of the heart and its dysfunctions. Ranging from benchmark cases to realistic geometries both in physiological and pathological conditions such as atrial tachycardia and fibrillation, and considering snapshots arising from both finite element method and NURBS-based Isogeometric Analysis discretizations, DL-ROMs and POD DL-ROMs are shown to be able to solve, after the training stage, these problems for any new scenario in real-time, even in extremely challenging contexts such as re-entry and re-entry break-up problems, modeling those triggering phenomena related with cardiac arrhythmias.

# Contents

# List of Figures

# List of Tables

# Introduction

Computational cardiac electrophysiology (EP) studies the electrical activity of the heart with the aim of modeling and numerically approximating physiological and pathological conditions, such as cardiac arrhythmias, standing as the focus of several efforts by the scientific community, see, e.g., [Prakosa et al., 2018, Vigmond et al., 2002, Trayanova, 2011, Vigmond et al., 2008, Niederer et al., 2009, Niederer et al., 2011, Strocchi et al., 2020]. Simulating the electrical behavior of the heart, from the cellular scale to the tissue level, relies on the numerical approximation of coupled nonlinear dynamical systems, such as, e.g. the Monodomain or the Bidomain equations, see, e.g., [Colli Franzone et al., 2005, Colli Franzone et al., 2006], coupled with suitable ionic models, such as the FitzHugh-Nagumo [FitzHugh, 1961, Nagumo et al., 1962], the Aliev-Panfilov [Aliev and Panfilov, 1996, Nash and Panfilov, 2004], the Roger-McCulloch [Rogers and McCulloch, 1994], the ten Tusscher-Panfilov [ten Tusscher and Panfilov, 2006] or the Mitchell and Schaeffer models [Mitchell and Schaeffer, 2003]. The formulation and the numerical approximation of mathematical models for describing both cellular phenomena and tissue-scale behaviors represent nowadays an extremely active field, see, e.g., the reaction-Eikonal model [Neic et al., 2017], the Bueno-Orovio [Bueno-Orovio et al., 2008], the Courtemanche-Ramirez-Nattel [Courtemanche et al., 1998] and the ToR-ORd [Tomek et al., 2019] ionic models. These systems describe the propagation of the electrical signal in the heart and the cardiac action potential (AP), that is the polarization/depolarization cycle occurring at every heartbeat that models the time evolution of the electrical potential across the cell membrane, as well as a set of ionic variables. The solution of cardiac EP systems entails mathematical, i.e. nonlinearity and coupling of the equations, and numerical, i.e. high computational costs related to small time-step sizes, unless the system might be unstable, and mesh sizes, in order to capture the steep fronts, critical issues.

Multiple solutions of these systems, corresponding to different model inputs parameters and data, as conductivities, ionic model parameters, applied currents, are required to evaluate outputs of clinical interest, such as activation maps, AP duration and electrograms, computed starting from the solution of numerical models. Cardiac EP models are unavoidably hampered by several uncertainties, due to the several model assumptions, the intrinsic (or physiological) variability, that is the natural differences among individuals, and the observational uncertainty, the measurements error in experimental data or the lack of information. The latter two sources of uncertainty result in parameters, boundary/initial conditions and geometry uncertainties. In this setting, sensitivity analysis and uncertainty quantification (UQ) provide essential tools to account for inter- and intra-subject variability [Mirams et al., 2016, Johnstone et al., 2016, Hurtado et al., 2017, Clayton et al., 2020]. Moreover, model personalization is achievable through the solution of data-assimilation and inverse problems, wherein some unknown quantities characterizing the mathematical model must be estimated from measurements [Dhamala et al., 2018, Quaglino et al., 2018, Johnston et al., 2018, Pathmanathan et al., 2019, Levrero-Florencio et al., 2020].

All these instances can be cast either in a *multi-query* or *real-time* context. In the former case, the input-output map is repetitively evaluated in order to perform multi-scenario analysis, to deal with uncertainties and with inter- and intra-subject variability and to consider specific pathological scenarios; in the latter one, outputs of interest must be computed in a very limited amount of time, in view of the integration in the clinical setting. Performing the numerical approximation of cardiac EP problems in a multi-query context or solving them in real-time, is unaffordable by means of traditional high-fidelity techniques, or full order models (FOMs), such as the Galerkin-finite element (FE) method [Quarteroni and Valli, 1994] and Isogeometric Analysis (IGA) [Cottrell et al., 2009]. To achieve computational efficiency, multi-query analysis and real-time problems must rely on suitable *surrogate* models which can be built according to different strategies (see, e.g., [Niederer et al., 2020] for a recent review on the topic). In [Coveney et al., 2020, Longobardi et al., 2020, Lei et al., 2020, Ayed et al., 2019] model emulators, aiming at approximating an input-output map by fitting a set of training data, are obtained by means of Polynomial Chaos Expansions, Gaussian Process (GP) regression or artificial neural networks (ANNs), possibly including physical laws in their definition [Sahli Costabal et al., 2020]. Surrogate models may also consist in lower-fidelity models, such as, for instance, the Eikonal [Colli Franzone et al., 2014] and the reaction-Eikonal models [Neic et al., 2017], and reduced order models (ROMs) which potentially are capable of more accurate approximations than data fitting techniques, yielding more significant computational savings than lower-fidelity models.

Reduced order modeling techniques aim at replacing the FOM by a reduced order model (ROM), featuring a much lower dimension, yet capable to express the physical features of the problem at hand. Projection-based methods are a widespread family of ROM techniques, among which the reduced basis (RB) method (relying, e.g., on proper orthogonal decomposition (POD) [Chatterjee, 2000]), under the form of either POD-Galerkin or POD-Petrov-Galerkin methods, represents one of the most popular options [Quarteroni et al., 2016]. The basic assumption underlying projection-based methods is that the solution of a parameter-dependent PDE lies on a low-dimensional manifold, which can be approximated by a *linear trial manifold* spanned by a set of basis functions [Benner et al., 2017, Benner et al., 2015], built from a set of FOM snapshots employing, e.g., POD. In this case, the ROM approximation is given by the linear superimposition of POD modes, whose degrees of freedom (depending both on time and parameters) result from the solution of a low-dimensional (nonlinear, dynamical) system, obtained through a (Petrov-)Galerkin projection onto a linear test subspace, which might coincide with the trial subspace. Being able to assemble such a ROM efficiently, for any new parameter instance, is granted at the price of a further *hyper-reduction* stage on the FOM arrays, usually exploiting techniques like the (discrete) empirical interpolation method (DEIM) [Chaturantabut and Sorensen, 2010] or the gappy POD [Willcox, 2006].

ROMs for parametrized PDEs rely on a suitable offline-online computational splitting: computationally expensive tasks required to build the low-dimensional subspace, and assemble all the ROM arrays, are performed once for all during the so-called *offline* (or *ROM training*) stage, then allowing to compute – ideally – in an extremely efficient way the ROM approximation for any new parameter value, during the so-called *online* (or *ROM testing*) stage. This computational strategy, however, breaks down if *(i)* the dimension of the linear trial subspace becomes very large (compared to the intrinsic dimension of the solution manifold being approximated), or *(ii)* the hyper-reduction stage, used to approximate parameter-dependent nonlinear terms, requires linear subspaces whose dimension becomes very large, too, in order to provide an approximation to FOM arrays sufficiently accurate. These can be recurrent issues when dealing with nonlinear, time-dependent parametrized PDEs, and *(i)* we aim at building a ROM able to provide problem approximations uniformly accurate over the whole parameter space, *(ii)* the parametrized problem features coherent structures (e.g., transport or wave phenomena) that propagate over time and strongly depends on parameters. Last, but not least, ensuring ROM stability might require additional care, e.g., when dealing with fluid flows using a mixed formulation (e.g., a velocity-pressure formulation for Navier-Stokes equations) [Dal Santo et al., 2019, Ballarin et al., 2015, Rozza and Veroy, 2007, Rozza et al., 2013]. A possible way to overcome the limitations shown by ROMs relying on a global linear trial manifold consists in using local reduced bases, built through POD after the set of snapshots has been split into clusters, according to suitable clustering (or unsupervised learning) algorithms [Amsallem et al., 2012, Amsallem et al., 2015].

Cardiac EP represents an extremely challenging test bed for traditional reduced order modeling techniques. This is primarily due to the high variability characterizing the solution manifold (with respect to the problem parameters), as well as to the nonlinear nature of the input-output maps that we intend to reconstruct numerically; indeed, cardiac EP models feature coherent structures that propagate over time. In particular, as soon as re-entry and re-entry break-up, the most recognized cellular mechanisms sustaining atrial tachycardia (AT) and atrial fibrillation (AF) [Nattel, 2002], are considered, wavefronts show an abnormal but still regular, in the former case, and chaotic and disorganized, in the latter one, activation patterns. These systems can hardly be reduced to lower dimensional problems by conventional ROMs such as, the RB method. At the best of our knowledge, a reliable, efficient and accurate ROM for parametrized physiological and pathological problems in cardiac EP is still lacking. An example is provided in [Pagani et al., 2018] where a local POD-Galerkin ROM has been proposed to handle physiological cardiac EP. Moreover, no attempt to construct a comprehensive reduced order modeling framework to efficiently deal with Bidomain equations or pathological scenarios, such as re-entry and re-entry break-up solutions, has been made yet. There are instead examples in which chaotic dynamical systems have been solved by means of deep learning algorithms. For instance, in [Pathak et al., 2018a, Pathak et al., 2018b], a hybrid forecasting scheme, based on reservoir computing in conjunction with knowledge-based models, is successfully applied to prototype spatiotemporal chaotic problems, as the Kuramoto-Sivashinsky equation in a chaotic regime. Moreover, a DL model for a model-free forecast of a chaotic dynamical system from noisy observations is developed in [Yeo, 2017].

In this Thesis, we propose a new generation of non-intrusive, nonlinear ROM techniques based on deep learning (DL) algorithms, which we refer to as DL-ROM, to deal with the construction of efficient ROMs in order to tackle parameter-dependent PDEs; in particular, we focus on PDEs featuring wave phenomena, such as, cardiac EP equations both in physiological and pathological scenarios, although several other scenarios dealing with different models are considered for the numerical assessment of the proposed techniques. Several recent works have shown possible applications of DL algorithms for solving PDEs – thanks to their ability of effectively approximating nonlinear maps, and by their ability to learn from data and generalize to unseen data – both from a theoretical [Kutyniok et al., 2019, Opschoor et al., 2020, Laakmann and Petersen, 2020, Petersen and Voigtlaender, 2018] and a computational standpoint. For example, Karniadakis and coauthors replace the FOMs by physics-informed neural networks (PINNs) [Raissi and Karniadakis, 2018, Raissi et al., 2017a, Raissi et al., 2017b, Raissi et al., 2019] trained by minimizing the residual of the PDEs, computed by means of automatic differentiation [Baydin et al., 2018]. In particular, in [Raissi, 2018] this framework is applied to the Kuramoto-Sivashinsky equation, in a chaotic regime, however, the algorithm leads to unsatisfactory approximations. A similar approach to PINNs can be found in [Han et al., 2017] and [Yang and Perdikaris, 2018] where physics-informed deep generative models, that is generative models based on DL algorithms, aiming at computing the solution of PDEs, are presented. DL techniques for parametrized PDEs have also been proposed in various recent works. In [Guo and Hesthaven, 2018, Guo and Hesthaven, 2019, Hesthaven and Ubbiali, 2018, Kast et al., 2020] feedforward neural networks have been employed to model the reduced dynamics in a less intrusive way, that is, avoiding the costs entailed by projection-based ROMs, but still relying on a linear trial manifold built, e.g., through POD. In [Kani and Elsheikh, 2017, Mohan and Gaitonde, 2018, Wan et al., 2018, Pulch and Youssef, 2020, Bērziņš et al., 2020] the construction of ROMs for nonlinear, time-dependent problems has been replaced by the evaluation of regression models based on ANNs. A similar approach can be found in [Regazzoni et al., 2019], where ANNs, such as feedforward neural networks, have been employed to model the reduced dynamics in a data-driven way, starting from input-output pairs generated through the FOM. In [San and Maulik, 2018, Wang et al., 2020] the authors address closure problems by means of feedforward and recurrent neural networks, that is the effects of POD truncated modes are modeled by DL models. In [Dal Santo et al., 2020] a neural network embedding a RB solver is presented and applied to steady parametrized problems. A first effort to include governing equations in the formulation of the ROM is presented in [Chen et al., 2020] where the linear ROM degrees of freedom are approximated by means of a feedforward neural network trained by minimizing the mean squared residual error. Similar results can be found in [Kani and Elsheikh, 2018] where residual neural networks are employed, having them the same definition of the forward Euler time scheme, to solve subsurface multi-phase flow. In [González and Balajewicz, 2018, Lee and Carlberg, 2020, Kim et al., 2020] the reduced trial manifold where the approximation is sought is modeled through ANNs, thus avoiding the linear superimposition of POD modes, on a minimum residual formulation to derive the ROM [Lee and Carlberg, 2020, Kim et al., 2020], or without considering an explicit parameter dependence in the differential problem [González and Balajewicz, 2018]. In all these works, coupled problems have never been considered. Moreover, very often DL techniques have been exploited to address problems which require only a moderate dimension of linear subspaces when dealing with projection-based ROMs.

To overcome the limitations of projection-based ROMs, we propose a strategy to construct DL-ROMs for nonlinear time-dependent parametrized PDEs in a non-intrusive way, exploiting deep neural networks as main building block, and a set of FOM snapshots. The proposed DL-ROM technique merges data-driven and physics-based models. Indeed, it exploits snapshots taken from a set of FOM solutions (for selected parameter values and time instances) and deep neural network architectures to learn, in a non-intrusive way, both *(i)* the *nonlinear trial manifold* where the ROM solution is sought, and *(ii)* the nonlinear reduced dynamics. In a linear ROM built, e.g., through POD, the former quantity is nothing but a set of basis functions, while the latter task corresponds to the projection stage in the subspace spanned by these basis functions. The nonlinear trial manifold is learnt by means of the decoder function of a convolutional autoencoder (AE) neural network, whereas the reduced dynamics through a (deep) feedforward neural network (DFNN), and the encoder function of the convolutional AE.

In this Thesis, we show that DL-ROMs outperform POD-Galerkin ROMs – even involving local reduced bases – regarding both numerical accuracy (for the same ROM dimension) and computational efficiency during the online (or testing) stage, when applied to problems that are typically challenging for the RB method (such as, e.g., linear transport equation, nonlinear diffusion-reaction equations whose solution develops moving fronts depending on parameters) or problems featuring reduced bases with usually large dimension. Moreover, a key aspect, yet not addressed by DL-ROMs and investigated in the second part of this work, deals with the enhancement of the computational efficiency of DL-ROMs during the offline (or training) stage, which is also strongly related with the curse of dimensionality. In particular, a strategy is proposed to enhance DL-ROMs in order to make the offline training stage dramatically faster, allowing for much larger FOM dimensions, without affecting the number of networks parameters to be estimated and, ultimately, network complexity. This strategy exploits *(i)* dimensionality reduction of FOM snapshots through randomized POD (rPOD) [Szlam et al., 2014], to be considered as the action of the first *layer* of the convolutional AE, rather than the way to generate the linear trial manifold, as done instead in traditional POD-Galerkin ROMs, and *(ii)* a suitable multi-fidelity pretraining stage [Goodfellow et al., 2016], where different models (built, e.g., by considering coarser discretizations or simplified physical models) can be efficiently combined, to iteratively initialize the network parameters. These substantial enhancements of the DL-ROM technique provide a new way to build DL-based ROMs, which we refer to POD DL-ROM.

(POD) DL-ROMs are then shown to be able to effectively handle parametrized problems in cardiac EP, accounting for both physiological and pathological conditions, in order to provide fast and accurate solutions. Whether DL-ROM is computationally efficient during the testing stage, that is for any new scenario unseen during the training stage, POD DL-ROM also enables fast training stages, that is it improves the weakest aspect and still takes advantage of the key properties of DL-ROM. Firstly, we demonstrate that our (POD) DL-ROM provides accurate results by constructing ROMs with extremely low-dimension in prototypical test cases. These tests all exhibit the relevant physical features which make the numerical approximation of parametrized problems in cardiac EP a challenging task. In particular, we consider snapshots arising from both a FE method and IGA [Cottrell et al., 2009] spatial approximations. IGA has been successfully applied to cardiac EP, modeled through the Monodomain or the Bidomain equations, in [Patelli et al., 2017, Pegolotti et al., 2019, Bucelli et al., 2020] where basis functions with high polynomial degree and global high order continuity in the computational domain have been considered. The authors show that the use of such basis functions is beneficial to the accurate approximation of the solution of cardiac EP problems. It is also worthy to highlight that so far only few works provide a combination of IGA with reduced order modeling. For example, in [Garotta et al., 2020, Salmoiraghi et al., 2016] IGA is embedded as FOM option in a ROM, where this latter is built by means of POD, and a pipeline integrated with free-from deformation, used for geometrical parametrization, is proposed. RB-IGA ROMs have been presented in [Manzoni et al., 2015, Rinaldi et al., 2015] and applied to steady potential flows and shell structural problems, respectively. In [Zhu et al., 2017a, Zhu et al., 2017b] IGA is used in combination with POD for reduced order modeling of linear parabolic PDEs and the time-dependent parameterized acoustic wave equation. Moreover, IGA snapshots are employed in [Haghighat et al., 2020] to train PINNs in order to solve the linear elasticity equation. Our interest in IGA is related to the ability of high order polynomials, with high order global continuity, to control and limit numerical dispersion thus accurately capturing wavefronts [Dedè et al., 2015, Pegolotti et al., 2019] and the smothness in the representation of the computational domain [Cottrell et al., 2009].

Finally, the performance of the POD DL-ROM technique is assessed on relevant test cases in cardiac EP on realistic geometries, both in physiological and pathological scenarios. These examples demonstate to be remarkable challenging tasks for reduced order modeling due to the steep wavefronts, the complex activation patterns associated to pathological scenarios, the high FOM dimension and the complexity of the geometries. (POD) DL-ROMs show to yield accurate and extremely efficient numerical approximations. This is particularly useful in view of the evaluation of patient-specific features to enable the integration of computational methods in current clinical practice; indeed outputs of clinical interest, such as activation maps, APs, electrograms and ablation targets can be more efficiently evaluated by the POD DL-ROM than by a FOM, while maintaining a high level of accuracy.

## Original contributions

This Thesis presents several original contributions from both a methodological and an applicative standpoints. The original methodological contribution consists in the development of a new generation of non-intrusive, nonlinear techniques to build ROMs for parametrized PDEs, based on pioneering DL algorithms, which we refer to as DL-ROM. The aim is to efficiently and accurately handle nonlinear time-dependent parametrized PDEs featuring coherent structures which propagate over time, such as transport, wave, or convection-dominated phenomena. The strategy we propose to enhance DL-ROMs, in order to make the offline training stage faster, which we refer to as POD DL-ROM, yield efficient numerical approximations to nonlinear time-dependent parametrized (vectorial) PDEs, by means of a prior dimensionality reduction, obtained through rSVD, and a suitable multi-fidelity pretraining, leading to the possibility to solve in real-time, during the online testing stage, parametrized PDEs modeling physical phenomena whose time scale is less than a second. From an applicative standpoint, (POD) DL-ROM has been successfully applied to different problems, with a particular emphasis on physiological and pathological cardiac EP, modeled through the Monodomain or the Bidomain equations coupled with suitable ionic models, and to different spatial discretization, such as the FE method and IGA. The proposed POD DL-ROM framework can efficiently perform the training and testing phases and provide real-time solutions to parametrized EP problems, thus enabling multi-scenario analysis in physiological and pathological cases. At the best of our knowledge, there have not been efforts in triyng to reduce the computational complexity associated to the re-entry and the re-entry break-up problems, which then represents a further innovation feature of this Thesis.

## Thesis organization

The thesis is organized as follows:

- **Chapter 1.** We provide a brief introduction on cardiac electrophysiology and the most relevant related pathologies, such as atrial tachycardia and atrial fibrillation. We review the most relevant mathematical models for cardiac electrophysiology and the complete derivation of the spatial (and temporal) discretization of the models we use to describe the electrical activity of the heart. We perform several numerical tests, which, at the best of our knowledge, are an example of the most advanced and complex problems tackled by means of Isogeometric Analysis in the cardiac electrophysiology context, in order to highlight the strong variability of the solution of these systems with respect to input parameters, such as conductivities, ionic models parameters and applied currents, motivating in this way the need of suitable reduced order models.

- **Chapter 2.** For the sake of computational efficiency, we show how to design nonlinear reduced order models by reinterpreting the classical ideas behind linear reduced order models for parametrized PDEs. Moreover, we review useful facts about deep learning and provide an overview of the models we rely on, to construct our new reduced order models, namely deep feedforward, convolutional and autoencoder neural networks.

- **Chapter 3.** We detail the construction of DL-ROMs, whose accuracy is numerically assessed by considering three different test cases of increasing complexity (with respect to the parametric dependence and the nature of the PDE) and by setting up nonlinear reduced order models whose dimension is nearly equal (if not equal) to the intrinsic dimension of the solution manifold that we aim at approximating.

- **Chapter 4.** We assess the computational performances of the proposed DL-ROM strategy on five relevant test cases in cardiac electrophysiology. This choice of numerical tests is aimed at highlighting the computational performance of the DL-ROMs in challenging electrophysiology problems, namely pathological cases in portion of cardiac tissues, such as the presence of an ischemic region and the figure of eight re-entry, or physiological scenarios on realistic left ventricle geometries.

- **Chapter 5.** We extend the DL-ROM to the POD-enhanced version, which we refer to as POD DL-ROM, to make the offline training stage dramatically faster, by means of prior dimensionality reduction, achieved through randomized singular value decomposition, and multi-fidelity pretraining, and we assess the numerical accuracy and efficiency on three different test cases, namely advection-diffusion-reaction, elastodynamics and Navier-Stokes equations, in order to highlight the robstuness of the approach.

- **Chapter 6.** We test the proposed POD DL-ROM strategy on benchmark test cases in cardiac electrophysiology, by highlighting the improvements introduced by the technique. We investigate the use of pretraining in three different scenarios: increasing the dimension of the full order model, enlarging the dimension of the parameter space and varying the geometry of the problem.
- **Chapter 7.** We assess the POD DL-ROM on challenging large-scale physiological and pathological examples on realisitic geometries, that is the three-dimensional Zygote left ventricle and the idealized NURBS left atrium surface geometries, by providing to POD DL-ROM snapshots arising from both finite-element method and Isogeometric Analysis discretizations, and by efficiently carrying out the training phase and providing real-time approximations at the testing stage.

Chapter 1

# Cardiac electrophysiology: mathematical and numerical modeling

This Chapter aims at providing a brief introduction on the mathematical and numerical models describing cardiac electrophysiology (EP). We introduce the basic principles of the heart phyisiology and function, and its electrical activity, as well as the main arrhythmias related to the propagation of the electrical signal in the atria, atrial tachycardia (AT) and fibrillation (AF). We review the most relevant mathematical models for cardiac EP depending on the scale at hand: at the microscopic level, systems of ordinary differential equations (ODEs) describe the cellular action potential generated by ionic currents through the cellular membrane, while at the macroscopic level, PDEs characterize the propagation of the action potential on the cardiac tissue. We recall the basic numerical techniques for the approximation of cardiac EP equations. Finally, we perform several numerical tests showing the strong dependence of the solution of cardiac EP equations on problems parameters and the need to perform multi-query analysis in order to account for different scenarios.

## 1.1 Overview of cardiac physiology

The heart is a double pump comprised of four chambers: two atria, the left atrium (LA) and the right atrium (RA), and two ventricles, the left ventricle (LV) and the right ventricle (RV). The atrioventricular septum separates the atria from the ventricles and let blood flows from the former to the latter through the tricuspid valve in the right part and through the mitral valve in the left part of the heart [Jarvik, 2004]. Non-oxygenated blood enters the RA through the superior and inferior venae cavae and gets pumped first into the right ventricle, then through the pulmonary valve and into the pulmonary circulation, where it is oxygenated by the lungs. The pulmonary veins let the flow coming from the lungs enter the left part of the heart, from which the blood is pumped again into the aorta and to the systemic circulation of the body [Altman and Dittmer, 1971, Opie, 2004]. The scheme of the heart anatomy is shown in Figure 1.1.

Muscle contraction and relaxation drive the pump function of the heart. In particular, tissue contraction is triggered by electrical signals self-generated in the heart and propagated through the myocardium thanks to the excitability of the cardiac cells, the cardiomyocites, see, e.g., [Colli Franzone et al., 2014, Klabunde, 2011]. When suitably stimulated, cardiomyocites produce a variation of the potential across the cellular membrane, called *transmembrane potential*. Its evolution in time is usually referred to as *action potential* (AP), involving a depolarization and a polarization in the early stage of every heartbeat. The action potential

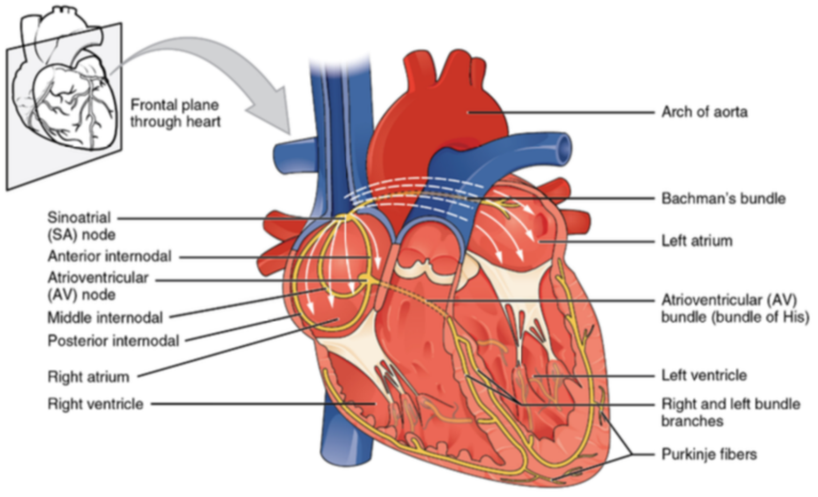


Figure 1.1: Anatomy of the anterior view of a frontal section of the heart.

is generated by several ion channels (e.g., calcium, sodium, potassium) that open and close, and by the resulting ionic currents crossing the membrane. Five phases of the AP are identified (see Figure 1.2) [Colli Franzone et al., 2014]. During phase 0 (depolarization), the $Na^+$ ionic channels of the sarcolemma, the lipid membrane that encloses cardiomyocytes, open, allowing a free flow of positive ions into the cell. Consequently, the transmembrane potential passes from a negative resting value of $-84$ mV to positive values. We denote with phase 1 the rapid decrease of potential due to an outward flow of $K^+$ and $Cl^-$ ions, occurring after the inactivation of the $Na^+$ channels. Phase 2 is characterized by both an inward current – caused by the transit of $Ca^{2+}$ – and an outward current – caused by the transit of $K^+$. This balance maintains the potential almost constant; usually, we refer to this phase with the term "plateau". The repolarization of the cell – phase 3 – is a consequence of the closing of the $Ca^{2+}$ channels. At this stage, the outward current is still due to the flow of $K^+$ that causes the potential to return to negative values of $-80/-85$ mV. During phase 4 the potential is kept at a constant value of $-84$ mV. Some of the $K^+$ ionic channels remain open, in order to guarantee the correct concentrations of ions outside and inside the cell. The cardiomyocyte stays in the resting phase until the next electric stimulation. The electrical signal propagates from one cardiomyocite to the surrounding ones because of the gap junctions located at the binding sites of adjacent cells.

The electric excitation of the heart starts in the RA at the sinoatrial node [Brooks and Lu, 1972], the natural pacemaker of the heart because of the ability of its cells to autonomously excite themselves, and travels across the atrial cardiac tissue. The signal travels from the RA to the LA through four muscular bundles [Sakamoto et al., 2005]: the Bachmann's bundle, the anterior septum, the posterior septum and the coronary sinus musculature. When the excitation front reaches the atrioventricular node, located at the basis of the RA, the signal is transmitted from the atria to the ventricles after a delay of about 100 ms, due to the slow conduction characterizing the cells in this area, as it travels along the bundle of His and Purkinje fibers [Dzialowski and Crossley, 2015]. Such delay is important to establish the synchronized contraction of atria and ventricles and to determine the right cardiac rhythm.

### 1.1.1 Atrial tachycardia and atrial fibrillation

Heart rhythm alterations (arrhythmias) affect millions of people [Alonso and Bengtson, 2014] and consist in conditions in which the heartbeat is irregular. There are two major types

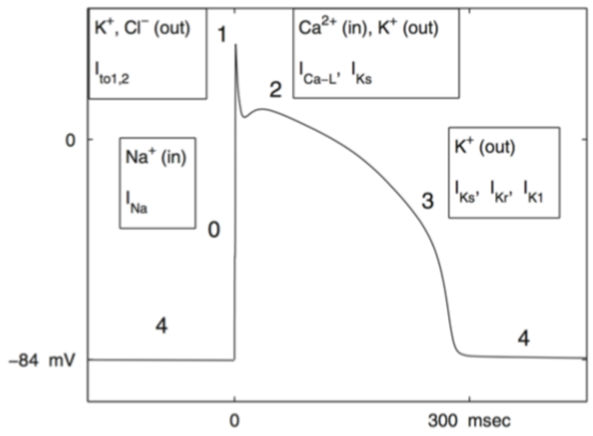


Figure 1.2: Evolution of the transmembrane potential vs. time during the AP [Colli Franzone et al., 2014].

of arrhytmias involving the upper chambers (atria) of the heart: atrial tachycardia and fibrillation. Atrial tachycardia (AT) occurs when the atria beat too quickly, in a fast but usually regular way, at rate higher than 100 beats per minute. ATs may be classified into two broad categories: focal and macro-re-entrant ATs [García-Cosío et al., 2012]. Focal ATs are defined as arrhythmias that arise from a circumscribed site of early activation and propagate to the atria in a centrifugal pattern. In macro-re-entrant ATs, atrial activation occurs in a continuous, uninterrupted manner because of a wavefront rotating around an obstacle consisting in anatomical structures (venous or valvular orifices), scars, or areas of functional block. In general, in these cases, the diameter of the circuit is greater than or equal to 2 cm [García-Cosío et al., 2012]. Atrial Fibrillation (AF) is the most common type of cardiac arrhythmia, affecting the 2% of world population [Morillo et al., 2017], and verifies when the heart electrical signal propagates in a rapid and irregular way throughout the atria [Nishida and Nattel, 2014]. During AF, atrial cells fire at rates up to 200-600 times per minute with respect to 60-80 beats per minute in healthy conditions at rest. Three clinical type of AF can be distinguished [Ogawa et al., 2018]: paroxysmal AF (episodes of arrhythmia that terminate spontaneously), persistent AF (episodes that continue for more than 7 days and are not self-terminating), and permanent AF (ongoing long-term episodes). Paroxysmal forms of AF show a predominance of local triggers/drivers, particularly from pulmonary veins (PVs). As AF becomes more persistent and eventually permanent, reentry substrates (initially functional and then structural) predominate [Iwasaki et al., 2011] (see Figure 1.3). AF is the major cause of heart failure and stroke [Kannel et al., 1982]. Indeed, it may lead to heart failure because the heart contraction is no longer effective. Moreover, it increases the risk of blood clots formation inside the atria and, if those clots travel through the bloodstream, it might ultimately lead to stroke. AF is the single most important cause of ischemic stroke in people older than 75-years old.

The mechanisms sustaining AT and AF have not been clearly identified, however macro-re-entry and re-entry break-up can be seen as cellular mechanisms undergoing the pathologies [Nattel, 2002]. Re-entry is a cellular mechanism where the electric signal does not complete the normal circuit, but rather follows an alternative path looping back upon itself and developing a self-perpetuating rapid and abnormal activation. Re-entries can be divided into two major types: anatomical and functional re-entry [Colli Franzone et al., 2014]. In the first case, re-entries develop around an anatomical obstacle, such as fibrosis, scar or slow conduction regions [Boyle et al., 2018, Saha et al., 2018]. In the following, we will focus on functional re-entries and to understand this mechanism we consider two zones of tissue, I

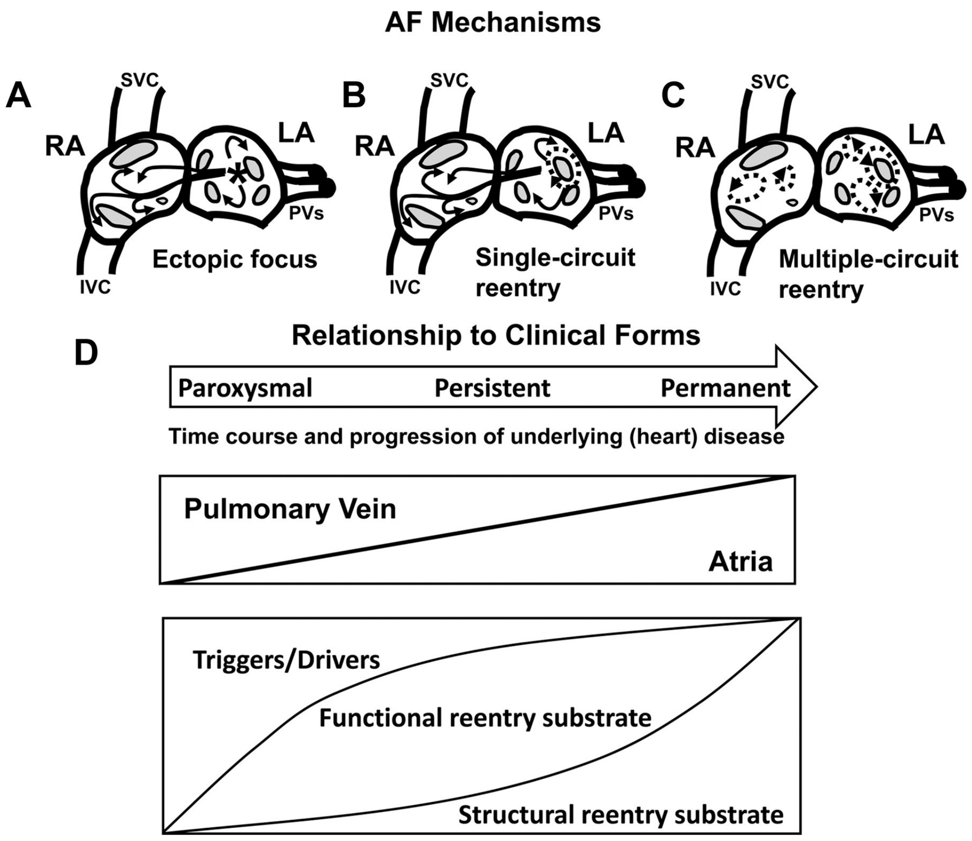


Figure 1.3: Clinical AF forms and relation to mechanisms [Iwasaki et al., 2011].

and II, which are connected as in Figure 1.4, an ectopic complex arising in zone II during the refractory period (RP) of zone I (the period in which the cells in zone I do not respond to a stimulus) will initially fail to activate zone I. The stimulus may propagate through an alternative path to return to zone I when its RP is over causing reactivation at this site. At this point, the signal leaves zone I and, if the time to return to zone II is sufficiently long, zone II will be reactived and the process can continue indefinitely. AT may be maintained by

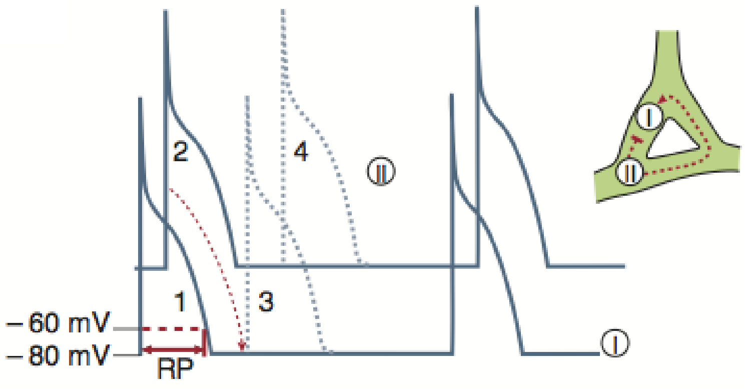


Figure 1.4: Re-entry occurring between two tissue zones, I and II [Nattel, 2002].

stable macro re-entry whereas, during an AF episode, multiple coexisting unstable re-entries can be observed, that is a single spiral wave is fragmented into a spatiotemporally chaotic pattern comprising many wavelets of various sizes, the so-called re-entry break-up. The major determinant of re-entry behavior, break-up or not, is the action potential duration (APD) restitution [Clayton and Taggart, 2005]. It refers to the curve relating APD to the previous diastolic interval (DI), the time between the end of the previous AP and the start of the next one (a sketch is reported in Figure 1.5 (left)). A way to measure the APD restitution curve is shown in Figure 1.5 (right) where the interval between the two stimuli, S1 and S2, is reduced until a wavefront can be elicited. The slope of the APD restitution curve is a measure of the recovery processes of all the ion channels. Numerical studies of alternans instability in two dimensions show that instability due to a steep restitution curve can cause spiral break-up [Panfilov and Holden, 1990, Karma, 1993]. In particular, in [Qu et al., 2010] the authors show that a steep APD restitution curve, that is the restitution relation has a slope steeper than 1, causes a dynamical instability of the system relating the APD to the DI, that is the equilibrium point becomes unstable.

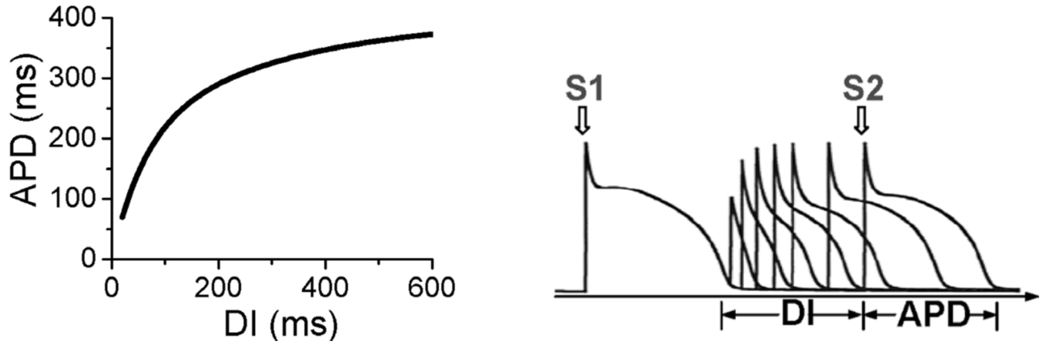


Figure 1.5: Left: APD restitution curve [Qu et al., 2010]. Right: S1-S2 protocol for determining APD restitution [Qu et al., 2010].

Several treatment options of AT and AF are available, including drug therapy, but the available drugs are not specific for atrial electrical activity and can have profound effects on ventricular EP, cardioversion and catheter ablation [Narayan et al., 2012a]. The latter is a procedure used to destroy the arrhythmia driving tissue [Narayan et al., 2012b]. A series of catheters are put into a blood vessel in the arm, groin or neck. The wires are guided into the heart and a special system sends radiofrequency energy which destroys small areas of heart tissue. Catheter ablation normally consists in isolating the PVs and destroying rotors cores but it is still to be proved that ablating the tips is useful [Allessie and de Groot, 2014, Narayan and Jalife, 2014]. Identifying the optimal target of ablation is far from being trivial, and the issue remains open and stands as the focus of several efforts by the scientific community.

## 1.2 Mathematical models for cardiac electrophysiology

The electrical activation of the heart, which drives its contraction, is the result of two processes [Quarteroni et al., 2017, Quarteroni et al., 2019, Colli Franzone et al., 2014]: the generation of ionic currents through the cellular membrane producing a local AP, at the microscopic scale; and the propagation of the AP from cell to cell in the form of a transmembrane potential, at the macroscopic scale. This latter process can be described by means of PDEs, suitably coupled with systems of ODEs modeling the ionic currents in the cells.

In order to model the propagation of the electric signal in the heart, we may consider the so-called Bidomain equations [Colli Franzone et al., 2014, Geselowitz and Miller III, 1983] in a domain $\Omega \subset \mathbb{R}^d$, with $d = 2, 3$, representing a portion of the myocardium, considered as a continuum composed by two interpenetrating domains, the intra- and the extracellular spaces. Each point $\mathbf{x} \in \Omega$ is associated with the intracellular potential $u_i$, the extracellular potential $u_e$, and the transmembrane potential $u = u_i - u_e$.

Coupling the parabolic-elliptic formulation of the Bidomain model for the transmembrane potential $u = u(\mathbf{x}, t)$ and the extracellular potential $u_e = u_e(\mathbf{x}, t)$ with a phenomenological[1] model for the ionic currents – involving a single gating variable $w = w(\mathbf{x}, t)$ – results in the following nonlinear time-dependent system

$$\begin{cases} \dfrac{\partial u}{\partial t} - \operatorname{div}(\mathbf{D}_i \nabla u) - \operatorname{div}(\mathbf{D}_i \nabla u_e) + I_{ion}(u, w) = I^i_{app} & (\mathbf{x}, t) \in \Omega \times (0, T), \\ -\operatorname{div}(\mathbf{D}_i \nabla u) - \operatorname{div}((\mathbf{D}_i + \mathbf{D}_e) \nabla u_e) = I^i_{app} + I^e_{app} & (\mathbf{x}, t) \in \Omega \times (0, T), \\ \dfrac{\partial w}{\partial t} + g(u, w) = 0 & (\mathbf{x}, t) \in \Omega \times (0, T), \\ \mathbf{D}_i \nabla (u + u_e) \cdot \mathbf{n} = 0 & (\mathbf{x}, t) \in \partial\Omega \times (0, T), \\ (\mathbf{D}_i + \mathbf{D}_e) \nabla u_e \cdot \mathbf{n} + \mathbf{D}_i \nabla u \cdot \mathbf{n} = 0 & (\mathbf{x}, t) \in \partial\Omega \times (0, T), \\ u(\mathbf{x}, 0) = u_0, \ w(\mathbf{x}, 0) = w_0 & \mathbf{x} \in \Omega. \end{cases} \tag{1.1}$$

Here $t$ and $u$ may denote a rescaled and dimensionless time and trasmembrane potential, depending on the ionic model considered[2], $\mathbf{n}$ denotes the outward directed unit vector normal to the boundary $\partial\Omega$ of $\Omega$, whereas $I^i_{app} = I^i_{app}(\mathbf{x}, t)$ and $I^e_{app} = I^e_{app}(\mathbf{x}, t)$ are the intra- and the extracellular applied currents representing, e.g., the initial activation of the tissue. The parabolic nonlinear diffusion-reaction equation for $u$ is two-way coupled with the ODE system, which must be in principle solved at any point $\mathbf{x} \in \Omega$; indeed, the reaction term $I_{ion}$ and the function $g$ depend on both $u$ and $w$. The most common choices for the two functions $I_{ion}$ and $g$ in order to efficiently reproduce the AP are, e.g., the FitzHugh-Nagumo [FitzHugh, 1961, Nagumo et al., 1962], the Aliev-Panfilov [Aliev and Panfilov, 1996, Nash and Panfilov, 2004], the Roger-McCulloch [Rogers and McCulloch, 1994] or the Mitchell and Schaeffer models [Mitchell and Schaeffer, 2003]. The diffusivity tensor $\mathbf{D}_i, \mathbf{D}_e$ usually depends on the fibers-sheet structure of the tissue, affecting directional conduction velocities and directions. In particular, by assuming an axisymmetric distribution of the fibers, the intra- and extracellular conductivity tensors take the form

$$\begin{aligned} \mathbf{D}_i(\mathbf{x}) &= \sigma^i_t \, I + (\sigma^i_l - \sigma^i_t) \, \mathbf{f}_0 \otimes \mathbf{f}_0, \\ \mathbf{D}_e(\mathbf{x}) &= \sigma^e_t \, I + (\sigma^e_l - \sigma^e_t) \, \mathbf{f}_0 \otimes \mathbf{f}_0, \end{aligned} \tag{1.2}$$

where $\sigma^i_l, \sigma^e_l$ and $\sigma^i_t, \sigma^e_t$ are the conductivities in the fibers and the transversal directions, for the intra- and extracellular conductivity tensors.

A simplified model, compared to the Bidomain model, is given by the Monodomain equation [Colli Franzone et al., 2014], written only in terms of the transmembrane potential $u$. Indeed, by assuming that the intra- and the extracellular conductivity tensors are such that $\mathbf{D}_i = \lambda \mathbf{D}_e$, it is possible to simplify the Bidomain equations (1.1). By setting

$$\mathbf{D} = \frac{1}{1 + \lambda} \mathbf{D}_i \qquad \text{and} \qquad I_{app} = \frac{\lambda}{1 + \lambda} I^i_{app} + \frac{1}{1 + \lambda} I^e_{app},$$

[1]Ionic models may be divided into three categories [Sundnes et al., 2006]: phenomenological, first generation, and second generation models. Phenomenological models describe the AP from the perspective of an external observer, without accounting for the underlying physiology. On the contrary, first and second generation models attempt to include a description of the mechanisms of the cell; while the former are based on simplified formulations to approximate the electrical behavior, the latter use instead sophisticated techniques allowing to represent sub-cellular processes. In the following, we will focus only on phenomenological models.

[2]In the A-P ionic model, dimensional times and potentials are given by $\tilde{t} = 12.9t$ $[ms]$ and $\tilde{u} = (100u - 80)$ $[mV]$. The transmembrane potential ranges from the resting state of $-80$ mV to the excited state of $+20$ mV.

the resulting coupled Monodomain system reads

$$\begin{cases} \dfrac{\partial u}{\partial t} - \operatorname{div}(\mathbf{D}\nabla u) + I_{ion}(u,w) = I_{app}(\mathbf{x},t) & (\mathbf{x},t) \in \Omega \times (0,T), \\ \dfrac{\partial w}{\partial t} + g(u,w) = 0 & (\mathbf{x},t) \in \Omega \times (0,T), \\ \nabla u \cdot \mathbf{n} = 0 & (\mathbf{x},t) \in \partial\Omega \times (0,T), \\ u(\mathbf{x},0) = u_0,\ w(\mathbf{x},0) = w_0 & \mathbf{x} \in \Omega, \end{cases} \tag{1.3}$$

where the conductivity tensor is defined as

$$\mathbf{D}(\mathbf{x}) = \sigma_t\, I + (\sigma_l - \sigma_t)\, \mathbf{f}_0 \otimes \mathbf{f}_0. \tag{1.4}$$

When a simple phenomenological ionic model is considered, such as the FitzHugh-Nagumo, the Aliev-Panfilov (A-P) [Aliev and Panfilov, 1996] or the Roger-McCulloch (R-M) [Rogers and McCulloch, 1994] model, the ionic current takes the form of a cubic nonlinear function of $u$ and a single (dimensionless) gating variable plays the role of a recovery function, allowing to model refractoriness of cells. In this work, we focus on the simple phenomenological A-P and R-M ionic models in order to decrease the computational costs associated to the solution of (1.1) and (1.3). The first consists in taking

$$\begin{aligned} I_{ion}(u,w) &= Ku(u-a)(u-1) + uw, \\ g(u,w) &= \Big(\epsilon_0 + \frac{c_1 w}{c_2 + u}\Big)(-w - Ku(u-b-1)), \end{aligned} \tag{1.5}$$

where the parameters $K$, $a$, $b$, $\varepsilon_0$, $c_1$, $c_2$ are related to the cell. Here $a$ represents an oscillation threshold, whereas the weighting factor $\varepsilon_0 + \frac{c_1 w}{c_2+u}$ was introduced in [Aliev and Panfilov, 1996] to tune the restitution curve to experimental observations by adjusting the parameters $c_1$ and $c_2$; see, e.g., [Clayton et al., 2011, Quarteroni et al., 2017, Quarteroni et al., 2019, Colli Franzone et al., 2014] for a detailed review.

For the R-M ionic model we rely on the following variant provided in [Rogers and McCulloch, 1994]

$$\begin{aligned} I_{ion}(u,w) &= Gu\Big(1 - \frac{u}{u_{th}}\Big)\Big(1 - \frac{u}{u_p}\Big) + \eta_1 uw, \\ g(u,w) &= \eta_2\Big(\frac{u}{u_p} - \eta_3 w\Big), \end{aligned} \tag{1.6}$$

where $G$, $\eta_1$, $\eta_2$, $\eta_3$ are positive coefficients, $v_{th}$ is a threshold potential, and $v_p$ is the peak potential.

The coupled systems (1.1) and (1.3) depend on several parameters representing either functional or geometric data such as, e.g., material properties, initial and boundary conditions, or the shape of the domain. Relevant physical situations are those in which input parameters affect the diffusivity matrix $\mathbf{D}$ (through the conduction velocities) and the applied current $I_{app}$; for previous analyses focused instead on the gating variable dynamics (through $g$) and the ionic current $I_{ion}$, see, e.g., [Pagani et al., 2018].

## 1.3 Numerical approximation of Monodomain and Bidomain equations

In this Section we provide the algebraic formulation arising from the spatial and time discretization of systems (1.3) and (1.1). Regarding the spatial discretization, we consider either the Galerkin-finite element (FE) method [Quarteroni and Valli, 1994] or NURBS-based Isogeometric Analysis (IGA) in the formulation of the Galerkin method [Quarteroni, 2013]. As for example, we present the discretization of the Monodomain equation (1.3) by means of the FE method and the Bidomain equations (1.1) by means of IGA. We will briefly discuss the pros & cons of the methods at the beginning of the next Section.

### 1.3.1 Discretization of the Monodomain equation: Galerkin-FE method

Here we provide the complete derivation of the spatial (and temporal) discretization of system (1.3). Hereon, whenever clear, we omit the dependence on the spatial variables $\mathbf{x}$.

We first state the weak formulation of problem (1.3), which stands at the basis of the numerical approximation of the problem, obtained with the Galerkin-FE method. The weak formulation of problem (1.3) reads: given $I_{app}(t) \in L^2(\Omega)$, find, for all $t \in (0, T)$, $u(t) \in X = H^1(\Omega)$ and $w(t) \in L^2(\Omega)$ such that

$$\begin{aligned}
\int_\Omega \left( \frac{\partial u}{\partial t} + I_{ion}(u, w) \right) \psi d\mathbf{x} + \int_\Omega \mathbf{D}\nabla u \cdot \nabla \psi d\mathbf{x} = \int_\Omega I_{app}(t) \psi d\mathbf{x} \qquad & \forall \psi \in H^1(\Omega), \\
\int_\Omega \frac{\partial w}{\partial t} \eta d\mathbf{x} = \int_\Omega g(u, w) \eta d\mathbf{x} \qquad & \forall \eta \in L^2(\Omega), \\
u(0) = u_0, \quad w(0) = w_0, &
\end{aligned}$$

where $\Omega$ is a Lipschitz domain of $\mathbb{R}^d$, $d = 2, 3$.

By introducing a triangulation $\mathcal{T}_h$, such that $\Omega = \text{int}( \cup_{K \in \mathcal{T}_h} K )$ (we are assuming, for the sake of simplicity, that the domain $\Omega$ and the computational domain $\Omega_h$ correspond), we apply the Galerkin-FE method on a finite-dimensional space $X_h \subset X(\Omega)$, where $X_h = X_h^r = \{v_h \in \mathcal{C}^0(\overline{\Omega}) : v_{h|K} \in \mathbb{P}^r, \forall K \in \mathcal{T}_h\}$, that is the space of globally continuous functions which are polynomials of order $r$ on each triangle of $\mathcal{T}_h$, of (usually very large) dimension $\dim(X_h) = N_h$; here by $h$ we denote a parameter related to the mesh size of the computational grid. By denoting with $\{\varphi_i\}_{i=1}^{N_h}$ a set of Lagrangian basis functions of the FE space $X_h$ (see Figure 1.6 for an example of basis function), we express the discrete approximation of $u(\mathbf{x}, t)$ and $w(\mathbf{x}, t)$ by

$$u_h(\mathbf{x}, t) = \sum_{i=1}^{N_h} u_i(t) \varphi_i(\mathbf{x}), \qquad w_h(\mathbf{x}, t) = \sum_{i=1}^{N_h} w_i(t) \varphi_i(\mathbf{x}),$$

where the vectors $\mathbf{u}_h = [u_1, \dots, u_{N_h}]^T$ and $\mathbf{w}_h = [w_1, \dots, w_{N_h}]^T$ are obtained by solving the following discrete system: find $\mathbf{u}_h = \mathbf{u}_h(t)$ and $\mathbf{w}_h = \mathbf{w}_h(t)$ such that

$$\begin{cases}
\mathbf{M} \dfrac{\partial \mathbf{u}_h}{\partial t} + \mathbf{A}\mathbf{u}_h + \mathbf{I}_{ion}(t, \mathbf{u}_h, \mathbf{w}_h) = \mathbf{I}_{app}(t) & t \in (0, T), \\
\dfrac{\partial \mathbf{w}_h}{\partial t} = g(\mathbf{u}_h, \mathbf{w}_h) & t \in (0, T), \\
\mathbf{u}_h(0) = \mathbf{u}_0, \quad \mathbf{w}_h(0) = \mathbf{w}_0. &
\end{cases}$$

Here we denote the mass matrix, the stiffness matrix and the activation term by

$$(\mathbf{M})_{ij} = \int_\Omega \varphi_i \varphi_j d\mathbf{x}, \quad (\mathbf{A})_{ij} = \int_\Omega \mathbf{D}\nabla \varphi_i \cdot \nabla \varphi_j d\mathbf{x}, \quad (\mathbf{I}_{app}(t))_j = \int_\Omega I_{app}(t) \varphi_j d\mathbf{x},$$

respectively; the vectors accounting for the ionic terms are instead given by

$$(\mathbf{I}_{ion}(\mathbf{u}_h, \mathbf{w}_h))_j = \int_\Omega I_{ion}(\mathbf{u}_h, \mathbf{w}_h) \varphi_j d\mathbf{x}, \quad (\mathbf{g}(\mathbf{u}_h, \mathbf{w}_h))_j = \int_\Omega g(\mathbf{u}_h, \mathbf{w}_h) \varphi_j d\mathbf{x}.$$

Concerning the treatment of nonlinear terms and time discretization, we rely on a semi-implicit, first order, one-step scheme [Colli Franzone and Pavarino, 2004]. Given a partition $(t^k, t^{k+1})$, with $k = 0, \dots, N_t - 1$, of $(0, T)$ into $N_t$ subintervals of length $\Delta t$, at each time-step $t^{k+1}$ the nonlinear vector $\mathbf{I}_{ion}$ is evaluated around the solution already computed at time $t^k$. In this way, we can decouple the PDE from the ODE, thus obtaining a linear

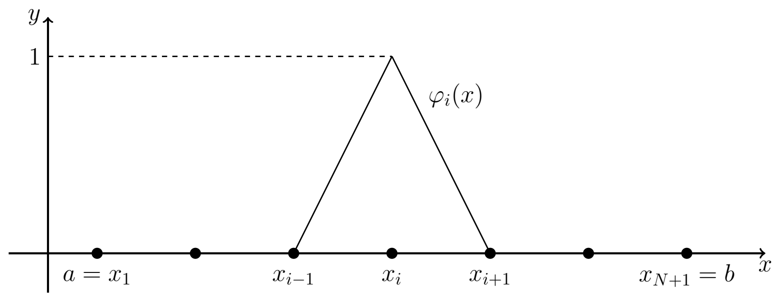


Figure 1.6: Basis function of $X_h^1$ related to node $x_i$ over the interval $(a, b)$ [Quarteroni, 2013].

system to be solved at each time step. Moreover, a ionic current interpolation strategy is used to evaluate the ionic current term, so that only the nodal values are used to build a (piecewise linear) interpolant of the ionic current. This is one of the two most common ways to deal with the evaluation of the ionic current at the quadrature nodes, and ultimately with the numerical integration of the ionic term $I_{ion}$, which go under the name of *state variable interpolation* (SVI) and *ionic current interpolation* (ICI), see, e.g., [Pathmanathan et al., 2010, Pathmanathan et al., 2012]:

- When using SVI, the variables $v_h$, $w_h$ are evaluated at the quadrature nodes $\bar{\mathbf{x}}_q$, $q = 1, \dots, N_Q$, so that

$$\int_\Omega I_{ion}(\mathbf{u}_h, \mathbf{w}_h)\varphi_j d\mathbf{x} \approx \sum_{q=1}^{N_Q} I_{ion}\left(\sum_{i=1}^{N_h} u_i\varphi_i(\mathbf{x}_q), \sum_{i=1}^{N_h} w_i\varphi_i(\mathbf{x}_q)\right)\boldsymbol{\omega}_q\varphi_j(\mathbf{x}_q),$$

  where $\{\boldsymbol{\omega}_q\}_{q=1}^{N_Q}$ denote the corresponding quadrature weights. This approach corresponds to the standard Galerkin-FE method.

- When relying on ICI [Pathmanathan et al., 2010], the currents are first evaluated in the degrees of freedom, and then interpolated at the quadrature nodes, so that

$$\int_\Omega I_{ion}(\mathbf{u}_h, \mathbf{w}_h)\varphi_j d\mathbf{x} \approx \sum_{q=1}^{N_Q} \Big(\sum_{i=1}^{N_h} I_{ion}\left(u_i, w_i\right)\varphi_i(\mathbf{x}_q)\Big)\boldsymbol{\omega}_q\varphi_j(\mathbf{x}_q),$$

  in order to reduce the computational cost associated to the assembly of the ionic currents term, compared to the SVI case.

Moreover, we also remark that since the ionic currents (zero order term) dominates the diffusion (second order term), a known numerical issue [Burman and Ern, 2003] might occur, ultimately causing numerical instabilities; to avoid them, we replace the mass matrix with a lumped mass matrix.

In conclusion, the fully discrete version of the (1.3) reads as: find $\mathbf{u}_h^{k+1} = \mathbf{u}_h(t^{k+1})$ and $\mathbf{w}_h^{k+1} = \mathbf{w}_h(t^{k+1})$ such that $\mathbf{u}_h^{(0)} = \mathbf{u}_0$, $\mathbf{w}_h^{(0)} = \mathbf{w}_0$ and, for $k = 0, \dots, N_t - 1$,

$$\begin{cases} \dfrac{\mathbf{w}_h^{k+1} - \mathbf{w}_h^k}{\Delta t} - \mathbf{g}(\mathbf{u}_h^k, \mathbf{w}_h^{k+1}) = \mathbf{0}, \\ \mathbf{M}\dfrac{\mathbf{u}_h^{k+1} - \mathbf{u}_h^k}{\Delta t} + \mathbf{A}\mathbf{u}_h^{k+1} + \mathbf{I}_{ion}(\mathbf{u}_h^k, \mathbf{w}_h^{k+1}) - \mathbf{I}_{app}^{k+1} = \mathbf{0}. \end{cases} \tag{1.7}$$

All the numerical simulations related to the solution of (1.7) are performed using the FE MATLAB library IHeart.

### 1.3.2 Discretization of the Bidomain equations: Galerkin-NURBS-IGA method

In FE methods differential problems are solved on polygonal or polyhedral meshes. However, there are many examples of practical applications, for example in computational mechanics, in which the domain of interest can not be exactly or smoothly represented by means of polygons or polyhedra. In such cases mesh generation is the typical intermediate step linking the geometry design to a suitable solution of the PDE defined on such geometry, i.e. the computational domain. Mesh generation usually constitutes the most critical aspect of the workflow for two reasons. The first and most important problem is that meshing the geometry often introduces a level of approximation [Cottrell et al., 2009] (see Figure 1.7). This is a particularly delicate matter when the geometry presents specific features (e.g. curved surfaces) that are hardly preserved by the discretization. Secondly, the accuracy of the approximate solution heavily depends on the quality of the mesh. IGA offers a valid alternative to methods requiring mesh generation. The smoothness in the representation of the computational domain, which has proved to be relevant in cardiac EP (see, e.g, [Patelli et al., 2017, Pegolotti et al., 2019]), makes the employment of B-splines and non-uniform rational B-splines (NURBS) basis functions extremely suited for the spatial approximation of several families of PDEs, also in virtue of the regularity of these basis functions. In Figure 1.8 we compare a three-dimensional idealized LA geometry meshed with tetrahedra, with a two-dimensional NURBS surface of the idealized LA built starting from B-spline basis functions of degree $p = 2$, together with control polyhedra.

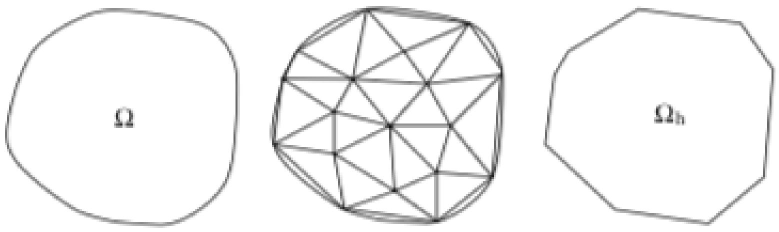


Figure 1.7: Example of meshing a non polygonal domain [Quarteroni, 2013].

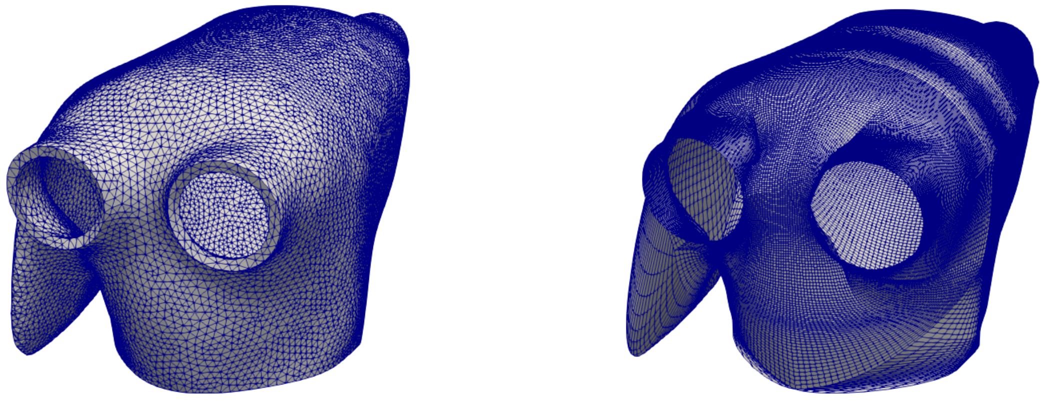


Figure 1.8: Three-dimensional LA computational domain meshed with tetrahedra (left) and two-dimensional NURBS LA computational domain (right).

NURBS-based IGA [Cottrell et al., 2009] is a concept in which the same NURBS basis functions employed for the design of the computational domain are used to construct the finite-dimensional space in which the numerical approximated solution of the PDE lays. Globally high order continuous polynomials have proved to control and limit numerical dispersion [Dedè et al., 2015], which may lead to artificial fractionated potential fronts, when dealing with regular solutions, as the sharp but smooth fronts arising in cardiac EP [Pegolotti et al., 2019]. As our interest is tackle pathological EP, we deem correctly capture these fronts to be very important. We believe IGA can play a role in accurately representing pathological EP as tachycardia and fibrillation. Moreover, by considering, for instance, [Pegolotti et al., 2019] where the authors consider NURBS basis functions with high polynomial degree, $p$, and global high order continuity, $\mathcal{C}^{p-1}$, in the computational domain, it has been proved that the use of such IGA basis functions is beneficial, in terms of accuracy/efficiency, for the approximation of the solution of the Bidomain equations, and it also limits dispersion effects typical of traveling wave phenomena. The same conclusions are drawn in [Patelli et al., 2017] for the solution of the Monodomain equation. The smoothness of the computational domain, together with the regularity of the NURBS basis functions, make IGA particularly suited for surface problems requiring high order polynomials to be used [Pegolotti et al., 2019].

Here we provide the fully discrete version of (1.1). Again, whenever clear, we omit the dependence on the spatial variables $\mathbf{x}$.

Provided a knot vector $\Xi = \{\xi_1, \dots, \xi_N\}$, we denote $\{\widehat{R}_j^p\}_{j=1}^n$ the set of univariate B-spline piecewise polynomials of degree $p$ built by means of the Cox-de Boor recursion formula [de Boor, 1972]; it holds that $N = n + p + 1$, where $n$ is the number of basis functions composing the B-spline basis. We remark that we rely only on open knot vectors, i.e knot vectors in which the first and last elements have the same multiplicity $p+1$. In this way, the knot vector $\Xi$ determines the polynomial degree $p$. It also determines the regularity, i.e. the number of continuous derivatives, of the basis functions over the knots through the multiplicity of the internal knots. More precisely, given an internal knot $\xi_i$, with multiplicity $m_i$, the resulting basis functions are $\mathcal{C}^{p-m_i}$ continuous over $\xi_i$. The univariate NURBS basis functions are generated from B-splines by considering a set of weights $\{\omega_j\}_{j=1}^n$ and the weighting function $W = \sum_{i=1}^n \widehat{R}_i^p \omega_i$, and by using the definition

$$\widehat{N}_j^p = \frac{\widehat{R}_j^p \omega_j}{W},$$

where $\omega_j \in \mathbb{R}$ and $\omega_j \geq 0$ for $j = 1, \dots, n$ (see Figure 1.9 for an example of univariate NURBS basis functions). The use of NURBS basis functions is motivated by geometrical needs; indeed since B-splines are piecewise polynomials, they can not exactly represent common geometries such as circles, cylinders, and conic sections in general, which can be instead represented by choosing appropriate weights to be associated with the B-splines. Multivariate B-splines and NURBS are built by means of the tensor product rule among univariate basis functions.

The weak formulation of problem (1.1) reads: given $I_{app}(t) \in L^2(\Omega)$, find $u(t) \in X = H^1(\Omega)$, $u_e(t) \in X \setminus \mathbb{R}$, the latter being the space of functions of $X$ with zero mean value on $\Omega$, and $w(t) \in L^2(\Omega)$ such that, for all $t \in (0, T)$,

$$\begin{aligned}
\int_\Omega \left( \frac{\partial u}{\partial t} + I_{ion}(u, w) \right) \psi d\mathbf{x} + \int_\Omega \mathbf{D}_i \nabla (u + u_e) \cdot \nabla \psi d\mathbf{x} &= \int_\Omega I_{app}^i(t) \psi d\mathbf{x} \qquad \forall \psi \in H^1(\Omega), \\
\int_\Omega \mathbf{D}_i \nabla u \cdot \nabla \psi d\mathbf{x} + \int_\Omega (\mathbf{D}_i + \mathbf{D}_e) \nabla u_e \cdot \nabla \psi d\mathbf{x} &= \int_\Omega (I_{app}^i(t) + I_{app}^e(t)) \psi d\mathbf{x} \qquad \forall \psi \in H^1(\Omega), \\
\int_\Omega \frac{\partial w}{\partial t} \eta d\mathbf{x} &= \int_\Omega g(u, w) \eta d\mathbf{x} \qquad \forall \eta \in L^2(\Omega), \\
u(0) = u_0, \quad w(0) &= w_0,
\end{aligned}$$

where $\Omega$ is a surface in $\mathbb{R}^3$.

Let us consider the multivariate NURBS basis functions $\{\widehat{N}_j\}_{j=1}^n$ and the invertible map-

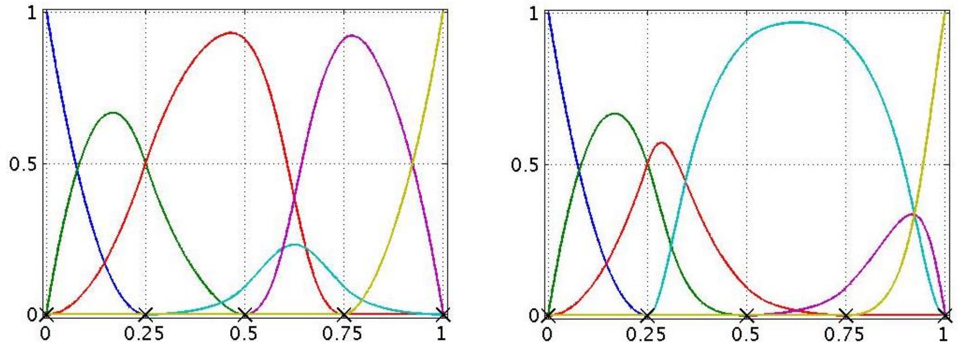


Figure 1.9: Examples of univariate NURBS basis functions with knot vector $\Xi = \{\{0\}^3, 1/4, 1/2, 3/4, \{1\}^3\}$, degree $p = 2$, number of elements $n_{el} = 4$, number of basis functions $n = 6$, globally $\mathcal{C}^1$ continuous and NURBS weights $\boldsymbol{\omega} = (1, 1, 1, \omega_4, 1, 1)$ with $\omega_4 = 0.1$ (left) and $\omega_4 = 0.1$ (right).

ping $\mathbf{x} : \widehat{\Omega} \to \Omega \subset \mathbb{R}^3$ defined as

$$\mathbf{x}(\mathbf{s}) = \sum_{j=1}^{n} \widehat{N}_j(\mathbf{s})\mathbf{B}_j,$$

where $\mathbf{B}_j \in \mathbb{R}^3$ is the so-called control points vector. We then apply Galerkin NURBS-based IGA on a finite-dimensional space $X_h \subset X(\Omega)$ spanned by the functions $\{N_j\}_{j=1}^n$, where $N_j = \widehat{N}_j \circ \mathbf{x}^{-1}$, i.e. $X_h = X \cap \{N_j\}_{j=1}^n$, of (usually very large) dimension $\dim(X_h) = n$. We express the discrete approximation of $u(\mathbf{x}, t)$, $u_e(\mathbf{x}, t)$ and $w(\mathbf{x}, t)$ by

$$u_h(\mathbf{x}, t) = \sum_{j=1}^{n} u_j(t) N_j(\mathbf{x}), \qquad u_{e,h}(\mathbf{x}, t) = \sum_{j=1}^{n} u_{e,j}(t) N_j(\mathbf{x}), \qquad w_h(\mathbf{x}, t) = \sum_{j=1}^{n} w_j(t) N_j(\mathbf{x}),$$

where the vectors $\mathbf{u}_h = [u_1, \ldots, u_n]^T$, $\mathbf{u}_{e,h} = [u_{e,1}, \ldots, u_{e,n}]^T$ and $\mathbf{w}_h = [w_1, \ldots, w_n]^T$ are obtained by solving the following discrete system: find $\mathbf{u}_h = \mathbf{u}_h(t)$, $\mathbf{u}_{e,h} = \mathbf{u}_{e,h}(t)$ and $\mathbf{w}_h = \mathbf{w}_h(t)$ such that

$$\begin{cases} \mathbf{M}\dfrac{\partial \mathbf{u}_h}{\partial t} + \mathbf{A}_i \mathbf{u}_h + \mathbf{A}_i \mathbf{u}_{e,h} + \mathbf{I}_{ion}(t, \mathbf{u}_h, \mathbf{w}_h) = \mathbf{I}^i_{app}(t) & t \in (0, T), \\ \mathbf{A}_i \mathbf{u}_h + \mathbf{A}\mathbf{u}_{e,h} = \mathbf{I}^i_{app}(t) + \mathbf{I}^e_{app}(t) & t \in (0, T), \\ \dfrac{\partial \mathbf{w}_h}{\partial t} = g(\mathbf{u}_h, \mathbf{w}_h) & t \in (0, T), \\ \mathbf{u}_h(0) = \mathbf{u}_0, \quad \mathbf{w}_h(0) = \mathbf{w}_0. \end{cases}$$

Here we denote the mass matrix, the stiffness matrices and the activation term by

$$(\mathbf{M})_{ij} = \int_\Omega N_i N_j d\mathbf{x}, \quad (\mathbf{A}_i)_{ij} = \int_\Omega \mathbf{D}_i \nabla N_i \cdot \nabla N_j d\mathbf{x},$$

$$(\mathbf{A})_{ij} = \int_\Omega (\mathbf{D}_i + \mathbf{D}_e) \nabla N_i \cdot \nabla N_j d\mathbf{x}, \quad (\mathbf{I}_{app}(t))_j = \int_\Omega I_{app}(t) N_j d\mathbf{x},$$

respectively; the vectors accounting for the ionic terms are instead given by

$$(\mathbf{I}_{ion}(\mathbf{u}_h, \mathbf{w}_h))_j = \int_\Omega I_{ion}(\mathbf{u}_h, \mathbf{w}_h) N_j d\mathbf{x}, \quad (\mathbf{g}(\mathbf{u}_h, \mathbf{w}_h))_j = \int_\Omega g(\mathbf{u}_h, \mathbf{w}_h) N_j d\mathbf{x}.$$

The null mean condition on the extracellular potential is given by $\mathbf{B}^T \mathbf{u}_{e,h}(t) = \mathbf{0}$.

Concerning the treatment of the nonlinear term, by considering $\Omega = \cup_{i=1}^{n_{el}} E_i$, that is the domain is composed by elements $E_i$, with $i = 1, \ldots, n_{el}$, and by means of the Gauss-Legendre quadrature formula with $s = (p+1)(q+1)$ quadrature nodes ($p$ and $q$ being the order of the piecewise polynomials in the two parametric directions) the integral is numerically computed as follow. Let $\Phi_i : (-1,1)^2 \to E_i$ be the transformation from the reference element to the $i^{th}$ element for the Gauss-Legendre quadrature formula, and $\{\hat{\mathbf{x}}_q^j\}_{j=1}^s$ with $\{\omega_q^j\}_{j=1}^s$ be the corresponding quadrature nodes and weights, the nonlinear term is computed by following the SVI approach

$$\int_\Omega I_{ion}(\mathbf{u}_h, \mathbf{w}_h) N_l d\mathbf{x} \approx \sum_{i=1}^{n_{el}} \sum_{q=1}^{s} I_{ion}\left(u_h(\mathbf{x}_q^{i,j}), w_h(\mathbf{x}_q^{i,j})\right) N_l(\mathbf{x}_q^{i,j})\, \boldsymbol{\omega}_q^j |\det(J_i)|,$$

where $J_i = \partial\Phi_i/\partial\hat{\mathbf{x}}$ is the Jacobian matrix of $\Phi_i$ with respect to the reference spatial variable and $\mathbf{x}_q^{i,j} = \Phi_i^{-1}(\hat{\mathbf{x}}_q^j)$.

In order to derive the fully discrete version of the (1.1), we consider a first order splitting scheme with semi-implicit treatment of the nonlinear term obtained by means of the Backward Differentation Formulas (BDF) of order 2 [Quarteroni et al., 2008]. In particular, in order to decrease the computational complexity of implicit time discretization, we employ the extrapoleted values of $\mathbf{u}_h(t)$ and $\mathbf{w}_h(t)$ obtained by linear combination of the solutions at previous time instants (see [Pegolotti et al., 2019] for further details). Given a partition $(t^k, t^{k+1})$, with $k = 0, \ldots, N_t - 1$, of $(0, T)$ into $N_t$ subintervals of length $\Delta t$, by means of the first order splitting scheme, we solve the problem find: $\mathbf{u}_h^{k+1} = \mathbf{u}_h(t^{k+1})$, $\mathbf{u}_{e,h}^{k+1} = \mathbf{u}_{e,h}(t^{k+1})$ and $\mathbf{w}_h^{k+1} = \mathbf{w}_h(t^{k+1})$ such that $\mathbf{u}_h^{(0)} = \mathbf{u}_0$, $\mathbf{w}_h^{(0)} = \mathbf{w}_0$ and, for $k = 0, \ldots, N_t - 1$,

$$\begin{cases} \dfrac{\mathbf{w}_h^{k+1} - \mathbf{w}_{h,BDF}^{k+1}}{\Delta t} - \dfrac{1}{\alpha_0}\mathbf{g}(\mathbf{u}_{h,*}^{k+1}, \mathbf{w}_{h,*}^{k+1}) = \mathbf{0}, \\ \mathbf{M}\dfrac{\mathbf{u}_h^{k+1} - \mathbf{u}_{h,BDF}^{k+1}}{\Delta t} + \mathbf{A}_i\mathbf{u}_h^{k+1} + \mathbf{A}_i\mathbf{u}_{e,h}^{k+1} + \mathbf{I}_{ion}(\mathbf{u}_{h,*}^{k+1}, \mathbf{w}_h^{k+1}) - \mathbf{I}_{app}^{i,k+1} = \mathbf{0}, \\ \mathbf{A}_i\mathbf{u}_h^{k+1} + \mathbf{A}\mathbf{u}_{e,h}^{k+1} - \mathbf{I}_{app}^{i,k+1} - \mathbf{I}_{app}^{e,k+1} = \mathbf{0}, \\ \mathbf{B}^T\mathbf{u}_{e,h}^{k+1} = \mathbf{0}, \end{cases} \tag{1.8}$$

where $\mathbf{u}_{h,*}^{k+1}$ and $\mathbf{w}_{h,*}^{k+1}$ are the extrapolated values of $\mathbf{u}_h^{k+1}$ and $\mathbf{w}_h^{k+1}$.

All the numerical simulations related to the solution of (1.8) are performed using the IGA C++ library isoGlib [Bartezzaghi, 2014].

## 1.4 Numerical results on atrial tachycardia and atrial fibrillation

In this work we want to handle parametrized problems arising in cardiac EP. In particular, we are interested in setting up a computational strategy to understand, study and perform multi-scenario analysis in the pathological context. In this Section, after a brief discussion about the pros & cons of the two numerical methods employed in the previous Section, i.e. the FE and IGA methods, we provide the numerical results, obtained by solving the Bidomain equations by means of NURBS-based IGA, both on a square slab of cardiac tissue and on idealized LA surface geometry, related to abnormal conditions of propagation of the electrical signal, such as re-entry and re-entry break-up, cellular mechanisms introduced in Subsection 1.1.1.

### 1.4.1 Comparison between FE and NURBS-based IGA methods

In the previous Section, we provide the full discretizion of systems (1.3) and (1.1) by means of the FE and NURBS-based IGA methods, respectively. In Table 1.1 we compare the properties of the two numerical methods considered. NURBS-based IGA smoothly represents the computational domain, starting from, for example, medical images, due to the use of NURBS basis functions with high polynomial degree and high order global continuity, with respect to the FE method, which instead exploit polyhedra elements. In addition to the smooth representation of the geometry, high order polynomials with high order continuity require a lower number of DOFs, when dealing with smooth solutions, with respect to the number required by the FE method, fixed the polynomials degree and a certain level of accuracy. The latter features make NURBS-based IGA highly suitable for surface problem which must be solved by means of high polynomials degree, such as the wave equation. Each univariate NURBS basis function has support in $p+1$ knot spans whereas a one-dimensional Lagrangian basis function has support in 2 mesh elements. That is, high order NURBS basis functions have a larger support than Lagrangian basis functions (this holds also for $d > 1$), thus resulting in higher computational times needed for the assembly of matrices and vectors appearing in systems (1.3) and (1.1) and higher memory consumption.

| | FE method | NURBS-based IGA method |
|---|---|---|
| polynomials order | $r \to N_h = (rn_{el} + 1)^d$ | $p \to n = (n_{el} + p)^d$ |
| global continuity | $\mathcal{C}^0$ | $\mathcal{C}^{p-m}$ |
| geometry representation | $\Omega_h = \text{int}(\ \cup_{K \in \mathcal{T}_h} K\ )$ with $K$ polyhedra | NURBS |
| surface problems | not suitable | suitable |
| basis functions support ($d = 1$) | 2 | $p + 1$ |

Table 1.1: Comparison between the properties of FE and IGA methods.

In virtue of the previous analysis, NURBS-based IGA represents a valid option to the Lagrangian FE method. In particular, we decided to handle pathological cardiac EP by means of IGA in order to take advantage of the properties of high order, with high global continuity, NURBS basis functions in controlling and limiting numerical dispersion, and in achieving higher accurracies, by means of a lower number of DOFs, if compared to high order Lagrangian basis functions, and to employ the NURBS surface representation of the LA. The moderate number of DOFs employed by NURBS-based IGA, for a prescribed degree of accuracy, make them amenable to the solution of the Bidomain equations, a more complete and larger system, being both $u$ and $u_e$ unknowns of the problem, with respect to the Modomain equation. We will solve the Monodomain equation, which usually requires a lower computational time to be solved with respect to the Bidomain equations, by means of the FE method, with piecewise linear polynomials, that is $r = 1$, for the development of the methodologies presented in the following Chapters. Indeed, in the former stage, we prefer to deal with a first order time scheme and spatial discretization method to decrease the computational costs associated to the solution of system (1.3). Moreover, we do not neglect that P1 finite elements have been extensively and successfully used for the numeriacal approximation of cardiac EP problems [Prakosa et al., 2018, Vigmond et al., 2002, Trayanova, 2011, Vigmond et al., 2008, Niederer et al., 2009, Niederer et al., 2011, Strocchi et al., 2020].

### 1.4.2 Numerical simulation of re-entry

We now consider the two-dimensional coupled PDE-ODE nonlinear system consisting of the Bidomain equation (1.1) coupled with the R-M ionic model (1.6) in a square slab of cardiac tissue $\Omega = (0, 2\ \text{cm})^2$. The tissue is composed by fibers laid parallel to the longitudinal direction, i.e. $\mathbf{f}_0 = (1, 0)^T$. The intra- and extracellular conductivities are set equal to $\sigma_l^i = 2 \times 10^{-3}\ \Omega^{-1}\text{cm}^{-1}$, $\sigma_t^i = 3.1 \times 10^{-4}\ \Omega^{-1}\text{cm}^{-1}$, $\sigma_l^e = 2 \times 10^{-3}\ \Omega^{-1}\text{cm}^{-1}$ and $\sigma_t^e = 1.3 \times 10^{-3}\ \Omega^{-1}\text{cm}^{-1}$ [Nagaiah et al., 2013].

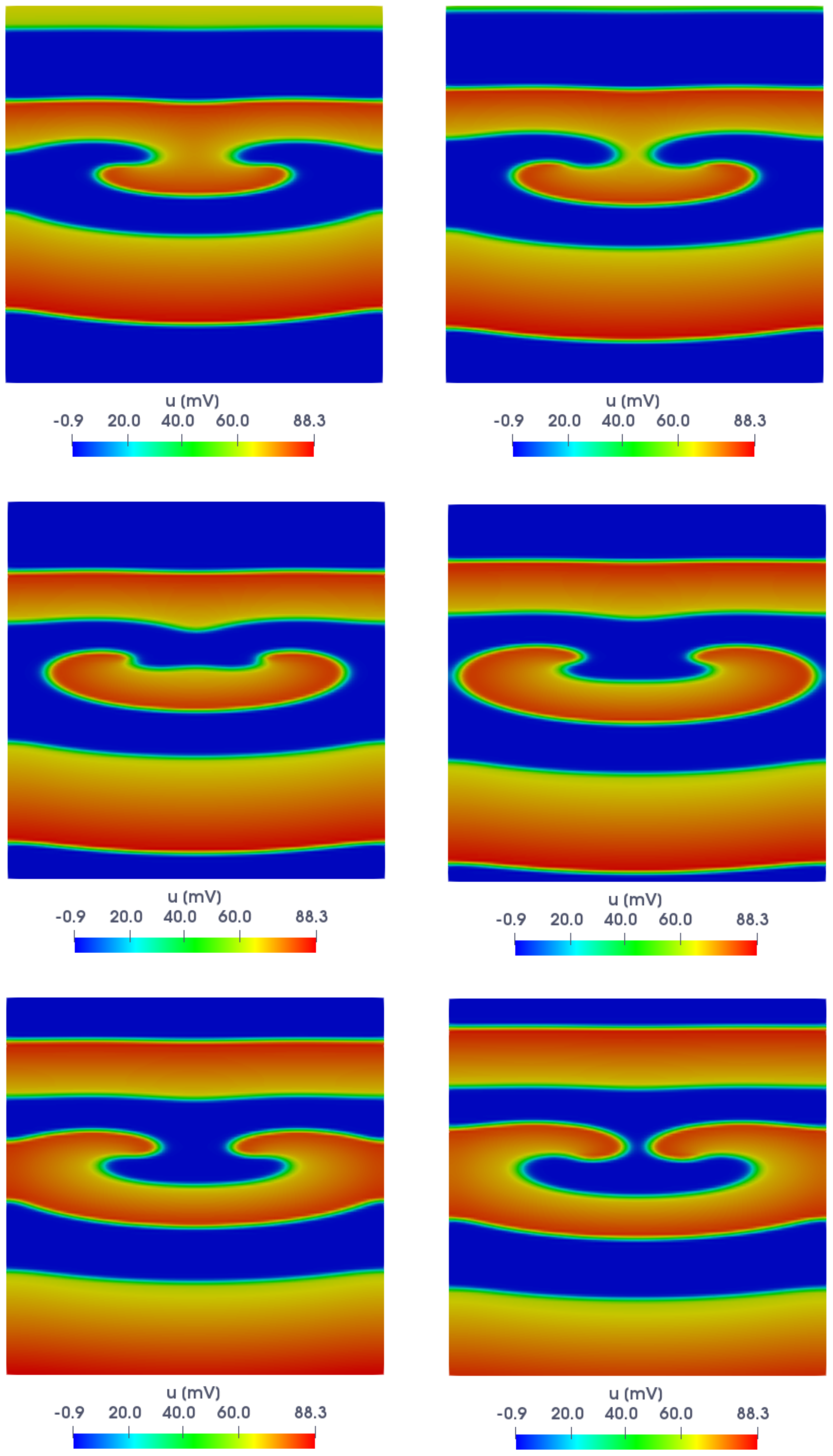


Figure 1.10: Figure of eight re-entry induced by S1-S2 protocol in a square slab of cardiac tissue at $t = 360, 370, 380, 390, 400, 410$ ms.

The parameters of the R-M ionic model are given by $u_{th} = 13$ mV, $v_p = 100$ mV, $G = 1.5$ ms$^{-1}$, $\eta_1 = 4.4$ ms$^{-1}$, $\eta_2 = 1.2 \times 10^{-2}$ and $\eta_3 = 1$ [Gerardo-Giorda, 2007]. The equations have been discretized in space through P2/C1 IGA by considering $n = 130 \times 130 = 16900$ NURBS basis functions, and in time by means of the explicit BDF of order 2 over the interval $(0, T)$, with $T = 600$ ms and a time-step $\Delta t = 0.1$ ms. In order to induce the re-entry we apply the S1-S2 stimulation protocol [Nagaiah et al., 2013, Colli Franzone et al., 2014]. In particular, a first stimulus (S1) is applied at the bottom edge of the domain, i.e.

$$I^{i,1}_{app}(\mathbf{x}, t) = C\mathbf{1}_{\Omega_1}(\mathbf{x})\mathbf{1}_{[t^i_1,t^f_1]}(t), \tag{1.9}$$

where $C = 100$ mA, $\Omega_1 = \{\mathbf{x} \in \Omega : y \leq 0.002\}$, $t^i_1 = 0$ ms and $t^f_1 = 5$ ms. A second stimulus (S2) under the form

$$I^{i,2}_{app}(\mathbf{x}, t) = C\mathbf{1}_{\Omega_2}(\mathbf{x})\mathbf{1}_{[t^i_2,t^f_2]}(t), \tag{1.10}$$

with $C = 100$ mA, $\Omega_2 = \{\mathbf{x} \in \Omega : (x-1)^2 + (y-1)^2 \leq (0.3)^2\}$, $t^i_2 = 183$ ms and $t^f_2 = 188$ ms, is then applied. The stimulus S1 induces a wavefront travelling towards the top edge of the domain. Then, S2 is applied at the center of the domain at a particular time in which the cells below are excitable whereas the tissue above is still refractory. The extracellular applied current $I^e_{app}$ is set instead equal to 0.

The solution consists in a rapid, periodic and self-sustained activation pattern composed by two stable spiral waves, called rotors, as shown in Figure 1.10. This particular kind of re-entry is called *figure of eight re-entry*.

Through the numerical solution of (1.1) it is possible to compute outputs of clinical interest such as activation maps (ACs), electrograms (EGMs) and rotors cores trajectories. For example, given the electric potential $u = u(\mathbf{x}, t)$, the (unipolar) activation map at a point $\mathbf{x} \in \Omega$ is evaluated as the minimum time at which the AP peak reaches $\mathbf{x}$ [Stella et al., 2020], that is

$$AC(\mathbf{x}) = \arg \min_{t\in(0,T)} \left( u(\mathbf{x}, t) = \max_{t\in(0,T)} u(\mathbf{x}, t) \right).$$

The resulting activation map is shown in Figure 1.11 (left). By exploiting the extracellular potential, solution of the (1.1), it possible to directly compute (unipolar) EGMs. In particular, we compute the average of the extracellular potential over the elctrode surface and, to take into account dispersion effects due to the measurement process, we employ a Gaussian kernel as follow [Plonsey and Barr, 1988]

$$EGM(\bar{\mathbf{x}}, t) = \int u_e(\mathbf{x}, t) \exp\Big(\frac{||\mathbf{x} - \bar{\mathbf{x}}||^2}{\sigma}\Big) d\mathbf{x},$$

where $\sigma = 2 \times (0.06)^2$ is chosen such that the support of the Gaussian kernel matches the dimension of the electrode, i.e. 4 mm. It is still to be proved that ablating rotors tips is useful; however, the position, at each time instant, of the cores may be an important insight to be used by clinicians during ablation intervention. For this reason, we compute the trajectories of the tips which are determined, over a period of 300 ms, by tracking the positions of the points laying on the contourline of the transmembrane potential $v = 50$ mV which possess maximum and minimum curvatures (with sign), as in [Patelli et al., 2017]. In Figure 1.12 (left) we show the tips position at $t = 550$ ms and in Figure 1.12 (right) the trajectory of the left tip with minimum curvature.

Finally, we investigate the use of an alternative stimulation protocol to induce the re-entry: the cross-field one [Hu et al., 2013]. It consists in applying the S1 stimulus as in (1.9) whereas the S2 is provided by

$$I^{i,2}_{app}(\mathbf{x}, t) = C\mathbf{1}_{\Omega_2}(\mathbf{x})\mathbf{1}_{[t^i_2,t^f_2]}(t),$$

where $C = 100$ mA, $\Omega_2 = \{\mathbf{x} \in \Omega : x \leq 1\}$, $t^i_2 = 160$ ms and $t^f_2 = 165$ ms.

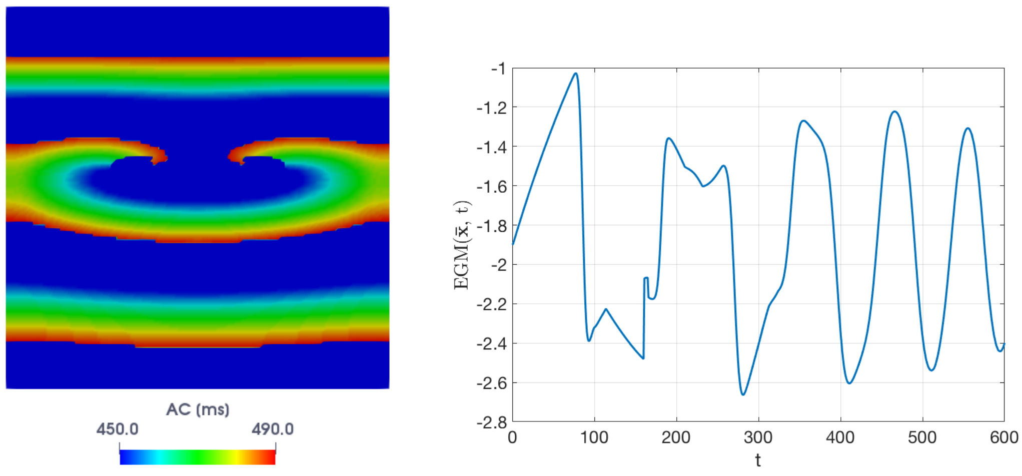


Figure 1.11: Left: Activation map for the figure of eight re-entry. Right: EGM computed at $\bar{\mathbf{x}} = (1.5, 1.5)$ cm.

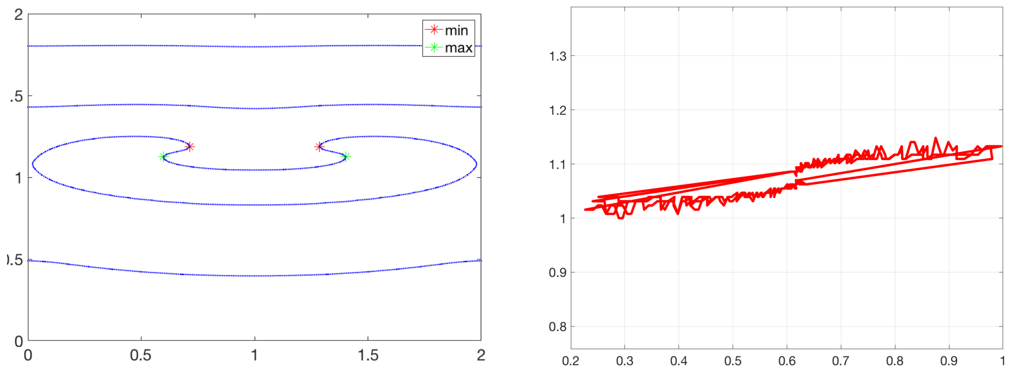


Figure 1.12: Left: Tips position at $t = 550$ ms. Right: Trajectory of the left spiral wave tip over 300 ms.

In Figure 1.13 we show the solution at $t = 345$ ms and $t = 495$ ms. This time, the solution consists in a single spiral wave, or rotor, surrounding a core, which is a zone of tissue unexcited but excitable.

### 1.4.3 Numerical simulation of re-entry break-up

In order to reproduce the re-entry break-up we consider the Bidomain equations (1.1) coupled with the A-P ionic model (1.5) in a square slab of cardiac tissue $\Omega = (0, 200 \text{ mm})^2$. The tissue is composed by fibers laid parallel to the longitudinal direction, i.e. $\mathbf{f}_0 = (1, 0)^T$. The intra- and extracellular conductivities are all set equal to 1 $\Omega^{-1}\text{cm}^{-1}$ and the parameters of the A-P ionic model are given by $K = 8$, $a = 0.1$, $\epsilon_0 = 0.01$, $b = 0.1$, $c_2 = 0.3$ and $c_1 = 0.05$ [ten Tusscher, 2004]. The time and the transmembrane potential are in dimensionless units according to the original formulation of the A-P ionic model. The equations have been discretized in space through P2/C1 IGA by considering $n = 202 \times 202 = 40804$ DOFs and in time by means of the explicit BDF of order 2 over the interval $(0, T)$, with $T = 1500$ and a time-step $\Delta t = 0.2$. In the case of the A-P model the most relevant dynamical change (for spiral wave stability) as $c_1$ is varied is the steepness of the APD restitution curve, where maximum slope increases as $c_1$ is decreased [ten Tusscher, 2004]. We choose a value of $c_1$

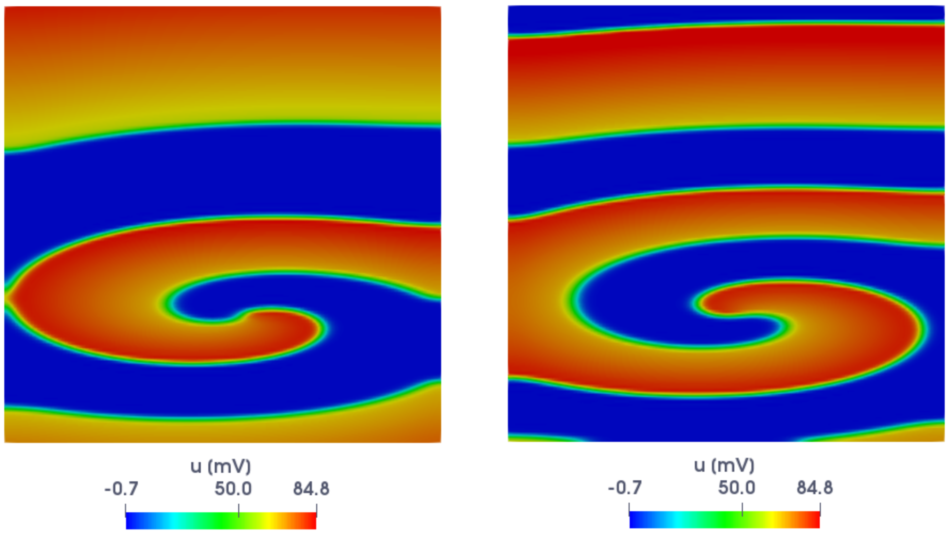


Figure 1.13: Spiral wave induced by the cross-field protocol in a square slab of cardiac tissue at $t = 345$ ms (left) and $t = 495$ ms (right).

for which the re-entry break-up is observed and in order to induce the re-entry we apply the cross-field stimulation protocol. In particular, the two stimuli are provided by (1.9) and (1.10) where $C = 1$ mA, $\Omega_1 = \{\mathbf{x} \in \Omega : y \leq 1\}$, $\Omega_2 = \{\mathbf{x} \in \Omega : x \leq 100\}$, $t_1^i = 0$, $t_2^i = 143$, $t_1^f = 5$ and $t_2^f = 148$.

In Figure 1.14 we show the solution at $t = 1100$ and $t = 1300$. At the beginning, the spiral wave is stable but, due to its high rotation frequency, the DI becomes shorter with respect to normal conditions. In this way the cells are characterized by a steep APD restitution curve (see Figure 1.5), meaning that for little changes in the DI there is a great variability in the APD (see Subsection 1.1.1). Therefore, cells close to each other have different responses to excitation. This causes conduction blocks and the subsequent spiral breaking. Indeed, almost at $t = 500$, the spiral wave starts to break-up and the activation pattern becomes completely chaotic over the time.

### 1.4.4 Numerical simulations on an idealized left atrium

We are interested in reproducing the previous behaviors (see Subsections 1.4.2 and 1.4.3) on an idealized LA geometry. We rely on a surface representation of the LA motivated by the fact that the cardiac tissue in the atria is thin and the transmural activation differences along the thickness can be assumed to be negligible. The LA is built as a single NURBS patch starting from B-splines basis functions of degree $p = 2$, for further information on its construction we refer the reader to [Patelli et al., 2017]. The intra- and extracellular conductivities are set equal to $\sigma_l^i = 3.1 \times 10^{-4}\ \Omega^{-1}\text{cm}^{-1}$, $\sigma_t^i = 2 \times 10^{-2}\ \Omega^{-1}\text{cm}^{-1}$, $\sigma_l^e = 1.3 \times 10^{-4}\ \Omega^{-1}\text{cm}^{-1}$ and $\sigma_t^e = 2 \times 10^{-3}\ \Omega^{-1}\text{cm}^{-1}$. The direction of the cardiac fibers is determined by following the same strategy adopted in [Patelli et al., 2017, Rossi et al., 2014]. The equations have been discretized in space by means of P2 NURBS basis functions, the majority with a global C1 continuity, with $n = 61732$. We apply again the S1-S2 protocol in order to induce the re-entry.

For the figure of eight re-entry test case, the S1 stimulus is applied in correspondence of one pulmonary vein and takes the form

$$I_{app}^{i,1}(\mathbf{x}, t) = C\mathbf{1}_{\Omega_1}(\mathbf{x})\mathbf{1}_{[t_1^i, t_1^f]}(t),$$

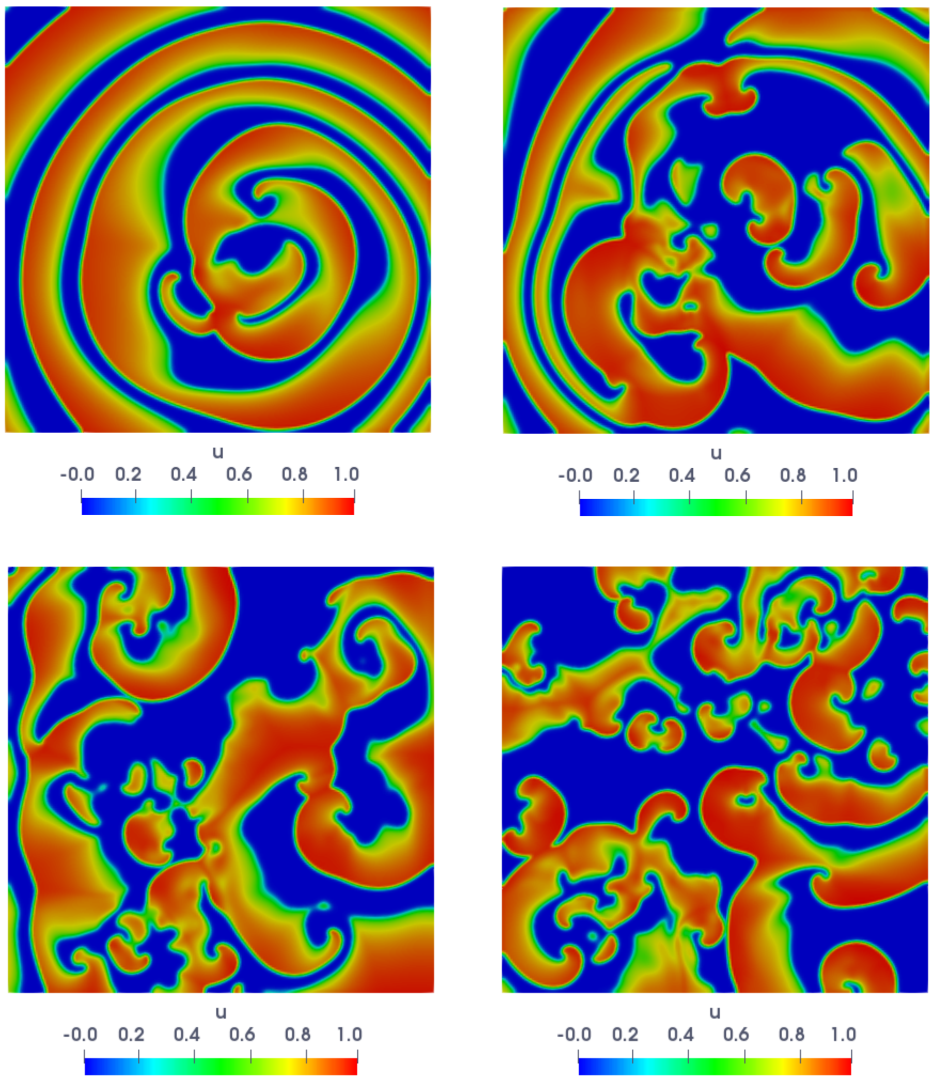


Figure 1.14: Re-entry break-up in a square slab of cardiac tissue at $t = 500, 700, 900, 1100$.

where $C = 100$ mA, $\Omega_1 = \{\mathbf{x} \in \Omega : x \geq 0, z \geq 2.7\}$, $t_1^i = 0$ ms and $t_1^f = 5$ ms. Provided the position of the posterior septum $\bar{\mathbf{x}} = (\bar{x}, \bar{y}, \bar{z})^T = (1.40, -0.66, -1.61)^T$, one of the four points at which the interatrial conduction is believed to happen, the S2 is given by

$$I_{app}^{i,2}(\mathbf{x}, t) = C\mathbf{1}_{\Omega_2}(\mathbf{x})\mathbf{1}_{[t_2^i, t_2^f]}(t),$$

with $C = 100$ mA, $\Omega_2 = \{\mathbf{x} \in \Omega : (x - \bar{x})^2 + (y - \bar{y})^2 + (z - \bar{z})^2 \leq (0.5)^2\}$, $t_2^i = 230$ ms and $t_2^f = 235$ ms. In Figure 1.15 we show the figure of eight re-entry solution on the idealized LA at $t = 1000, 1100$ and $1200$ ms.

For the simulation of re-entry break-up, the location of the two stimuli is the same used in the previous example. The only difference regards the stimulation times of the S2 which are $t_2^i = 145$ and $t_2^f = 150$. The solution is reported in Figure 1.16 together with the activation map in Figure 1.17.

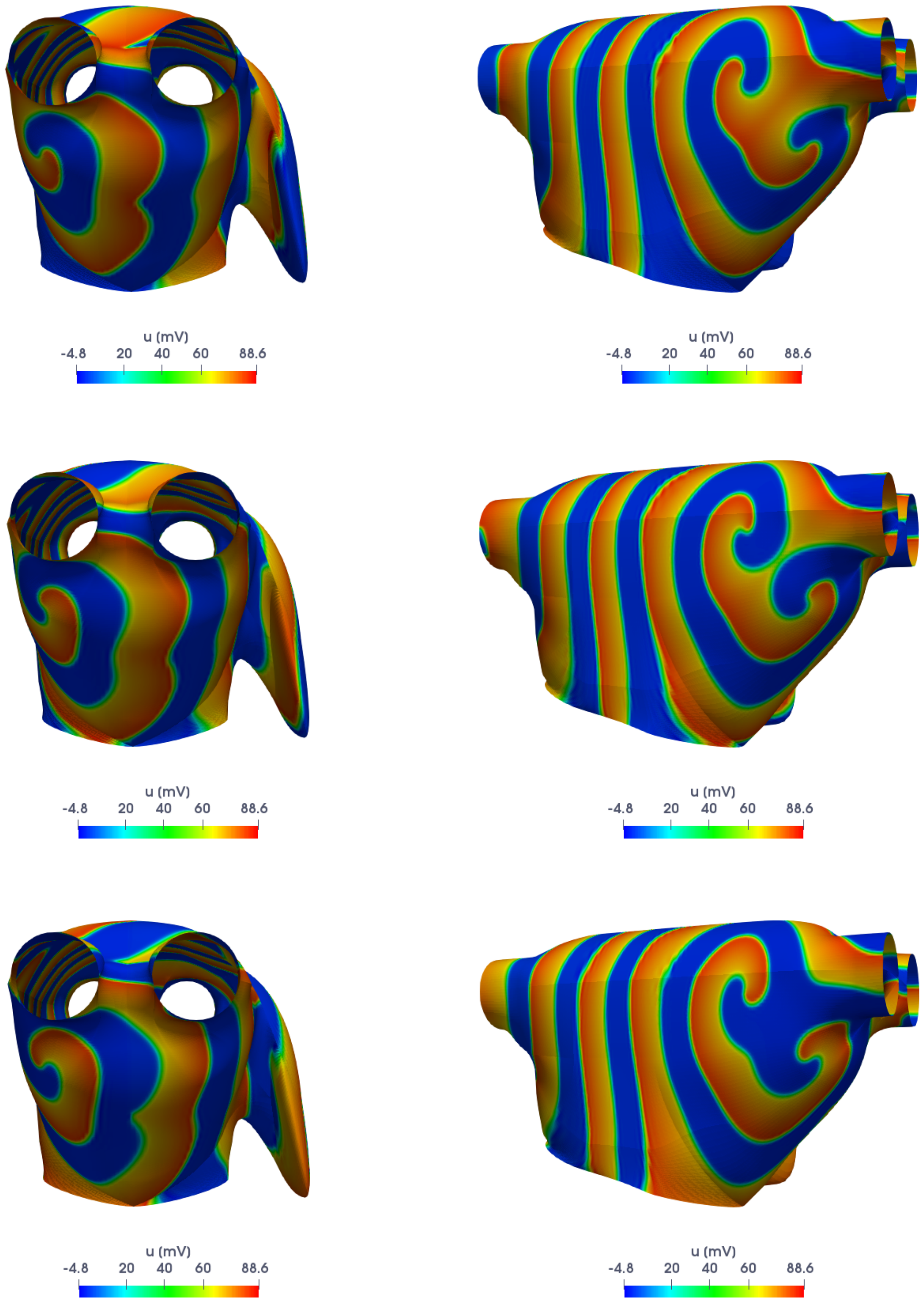


Figure 1.15: Figure of eight re-entry on an idealized LA geometry at $t = 1000, 1100, 1200$ ms.

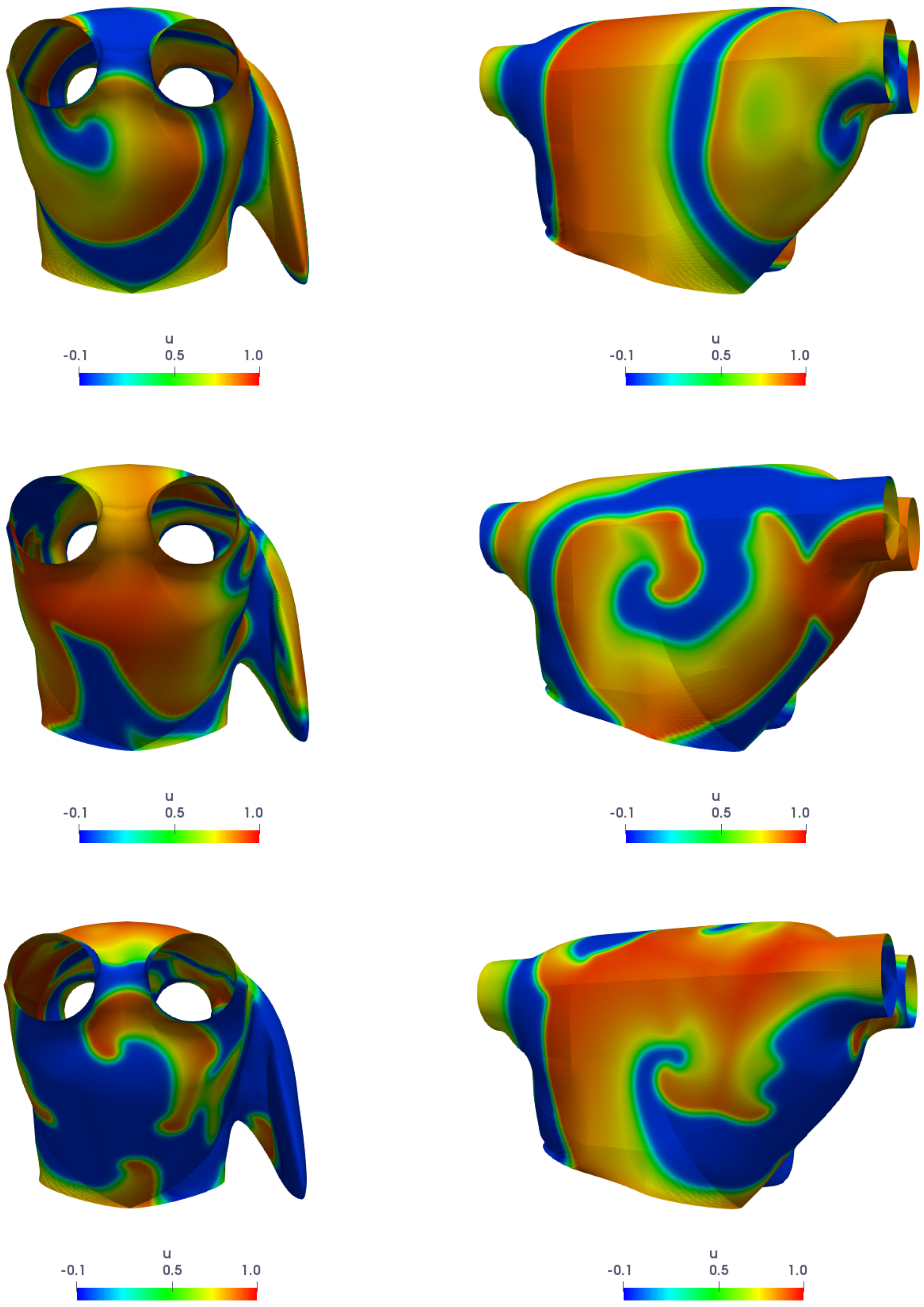


Figure 1.16: Re-entry break-up on an idealized LA geometry at $t = 400, 800, 1200$.

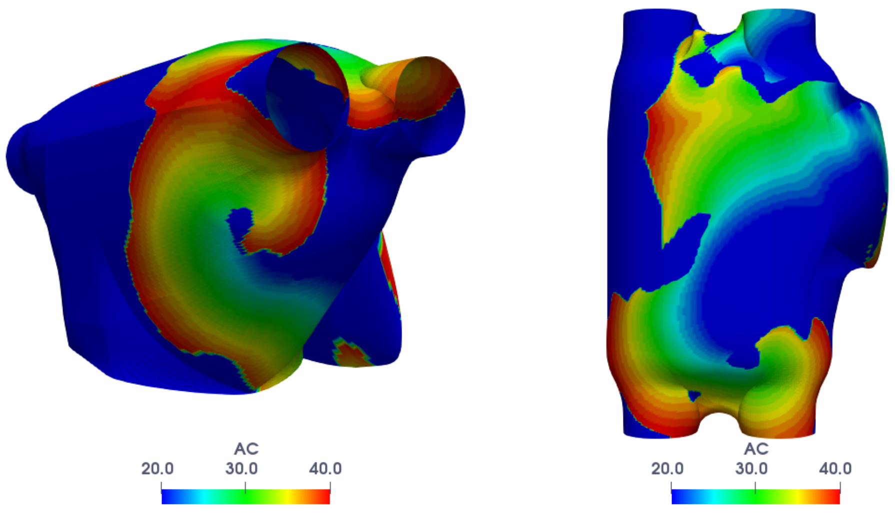


Figure 1.17: Re-entry break-up activation map.

## 1.5 Variability of the solution with respect to problem parameters

The solution of cardiac EP problems usually requires small time-step sizes, unless the system might be unstable, and mesh sizes, in order to capture the steep fronts (see, e.g., [Trayanova, 2011, Plank et al., 2008] where the mesh size $h$ required to solve, by means of linear finite elements, the Monodomain equation coupled with the ten Tusscher-Panfilov ionic model [ten Tusscher and Panfilov, 2006] on a three-dimensional slab geometry, both in physiological and pathological conditions, is provided). Consequently, solving equations (1.3) and (1.1) by means of the FE and the NURBS-based IGA methods can easily entails prohibitive computational costs, especially if, due our interest, parameters have a determinant role in developing pathological conditions and then the equations must be solved for several parameters values. The goal of this Section is to model and simulate scenarios of clinical interest, such as, for example, arrhytmogenic properties of the cardiac tissue or different stimulation locations, by highlighting the strong variability of the solution of equations (1.1) with respect to the problem parameters and then the need to develop suitable computational strategies to efficiently solve these parametrized problems in cardiac EP.

In Table 1.2 we report the numerical experiments presented down in this Section together with the properties and the parameters under investigation.

1. *Longitudinal intracellular conductivity* $\sigma_l^i$: we set the value of this parameter to $\sigma_l^i = 2 \times 10^{-2}\ \Omega^{-1}\text{cm}^{-1}$ (left) and $\sigma_l^i = 2 \times 10^{-4}\ \Omega^{-1}\text{cm}^{-1}$ (right) starting from the reference one, equal to $\sigma_l^i = 2 \times 10^{-3}\ \Omega^{-1}\text{cm}^{-1}$. By modifying this parameter we are changing the conductivity properties of the tissue in the fibers direction, as shown in Figure 1.18. We remark that, in order to make sure that the particular shape of the re-entry associated to $\sigma_l^i = 2 \times 10^{-4}\ \Omega^{-1}\text{cm}^{-1}$ is not related to numerical instabilities, due to a coarse mesh, we increase $n$ by setting it equal to $258 \times 258 = 66564$.

2. *Transversal intracellular conductivity* $\sigma_t^i$: we set the value of this parameter to $\sigma_t^i = 1 \times 10^{-3}\ \Omega^{-1}\text{cm}^{-1}$ (left) and $\sigma_t^i = 3.1 \times 10^{-5}\ \Omega^{-1}\text{cm}^{-1}$ (right) starting from the reference one, equal to $\sigma_t^i = 3.1 \times 10^{-4}\ \Omega^{-1}\text{cm}^{-1}$. By modifying this parameter we are changing the conductivity properties of the tissue in the transversal direction to the fibers, as shown in Figure 1.19.

| test number | property | parameter |
|---|---|---|
| 1. | longitudinal intracellular conductivity | $\sigma_l^i$ |
| 2. | transversal intracellular conductivity | $\sigma_t^i$ |
| 3. | magnitude of the S2 intracellular applied current | $I_{app}^{i,2}$ |
| 4. | different restitution properties | $\eta_2$ |
| 5. | dispersion of refractoriness | $\eta_2$ |
| 6. | S1-S2-S3 cross-field protocol | $I_{app}^{i,3}$ |
| 7. | APD restitution properties | $c_1$ |
| 8. | fibrosis | $\mathbf{D}_i$ |
| 9. | S1-S2 locations inverted | $I_{app}^{i,1}, I_{app}^{i,2}$ |
| 10. | S1-S2 locations on PVs | $I_{app}^{i,1}, I_{app}^{i,2}$ |
| 11. | ionic properties | $c_1$ |
| 12. | sustainment on long time | $T$ |

Table 1.2: Numerical tests gallery.

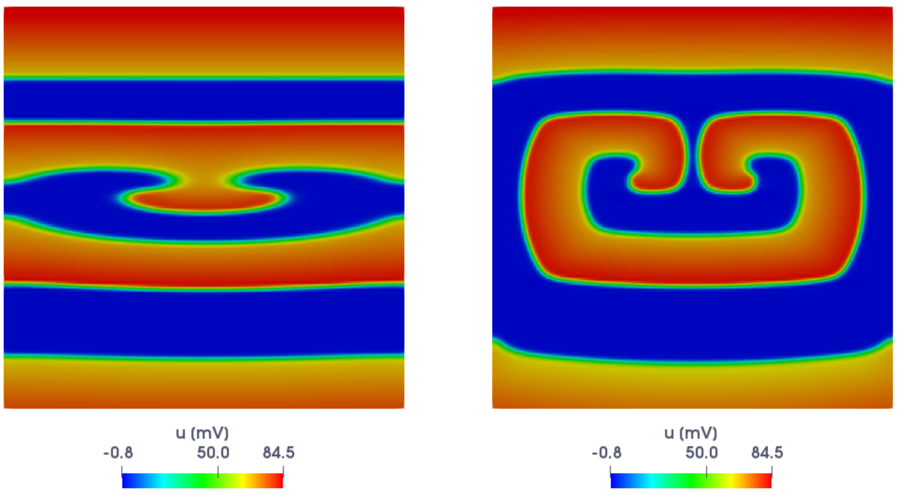


Figure 1.18: Figure of eight re-entry for $\sigma_l^i = 2 \times 10^{-2}\ \Omega^{-1}\text{cm}^{-1}$ (left) and $\sigma_l^i = 2 \times 10^{-4}\ \Omega^{-1}\text{cm}^{-1}$ (right) at $t = 430$ ms.

3. *Magnitude of the S2 intracellular applied current* $I_{app}^{i,2}$: we change the value of $C$ in (1.10) from 100 mA to $C = 10$ mA (left) and $C = 1000$ mA (right). In the first case, the magnitude of the applied current is too low in order to excitate the tissue, the applied stimulus is not able to start an AP, and the re-entry is not elicited, as shown in Figure 1.20.

4. *Different restitution properties in the domain*: we modify the $\eta_2$ coefficient in the R-M ionic model in the square subdomain in Figure 1.21 (left) by changing it to $6 \times 10^{-3}$ from the reference value $\eta_2 = 1.2 \times 10^{-2}$ [Clayton and Taggart, 2005]. By varying this parameter we modify the restitution properties of the tissue, i.e. the APD and the RP. In particular, increasing the value of $\eta_2$ reflects in a larger APD (see Figure 1.21 (right)). The presence of the square region with a different RP modify the shape and the re-entry direction of propagation, as shown in Figure 1.22.

5. *Dispersion of refractoriness*: this is a property of the myocardium for which the RP is spatially heterogeneous. Dispersion of refractoriness improves the arrhytmogenic propensity of the tissue, that is it favors the initiation and sustainment of re-entries.

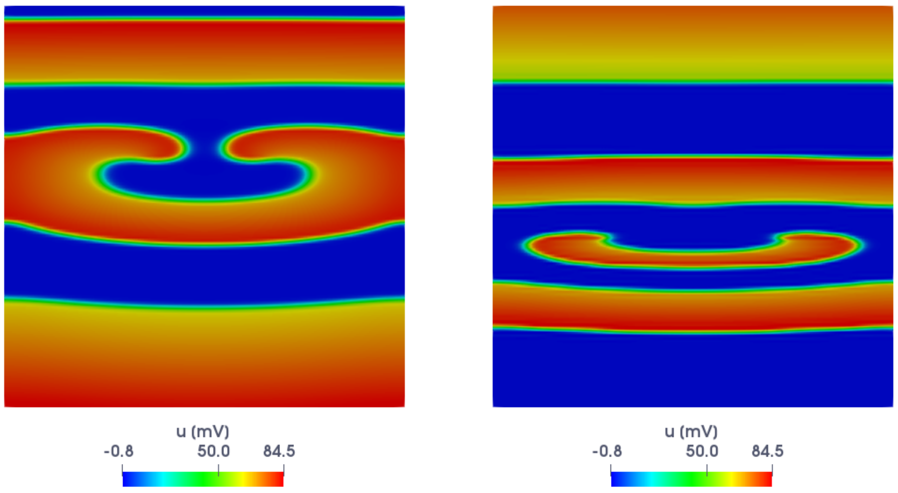


Figure 1.19: Figure of eight re-entry for $\sigma_l^t = 1 \times 10^{-3}\ \Omega^{-1}\text{cm}^{-1}$ (left) and $\sigma_t^i = 3.1 \times 10^{-5}\ \Omega^{-1}\text{cm}^{-1}$ (right) at $t = 400$ ms.

In normal cardiac tissue, the size of the S2 needs to be larger than the one needed in diseased tissue in order to create rotors. Therefore, a healthy heart is highly protected, whereas a diseased heart is susceptible to re-entrant arrhythmias. The classic explanation for this difference is that re-entry initiation is greatly facilitated if an excitation wave propagates into an arrhythmogenic tissue characterized by dispersion of refractoriness [Karma, 2013]. In the R-M ionic model, the parameter responsible for RP is $\eta_2$ and, in order to model dispersion of refractoriness, we set it equal to

$$\eta_2(\mathbf{x}) = 0.006(1 + |\sin(1\pi x) + \cos(10\pi y)|), \tag{1.11}$$

with mean almost equal to 0.012, that is the physiological value of this parameter. We apply the S1-S2 stimulation protocol by setting $\Omega_2 = \{\mathbf{x} \in \Omega : (x-1)^2 + (y-1)^2 \leq (0.15)^2\}$ and compare the results obtained by considering healthy tissue and myocardium with dispersion of refractoriness. By referring to Figures 1.23, 1.24 and 1.25 and by considering the healthy tissue, it happens that the distance between the two tips is too small and after one rotation the spirals collapse whereas, in the other case, due to the presence of cells with a shorter RP, the re-entry is sustained.

6. *S1-S2-S3 cross-field stimulation protocol*: we apply a second S2 and a third S3 stimuli of the form
$$I_{app}^{i,j}(\mathbf{x}, t) = C\mathbf{1}_{\Omega_j}(\mathbf{x})\mathbf{1}_{[t_j^i, t_j^f]}(t),$$
where $C = 100$ mA, $\Omega_2 = \{\mathbf{x} \in \Omega : x \leq 0.75\}$ and $\Omega_3 = \{\mathbf{x} \in \Omega : x \geq 1.5\}$, $t_2^i = 140$ ms, $t_2^f = 145$ ms, $t_3^i = 165$ ms and $t_3^f = 170$ ms. We show the solution obtained by applying the S1-S2-S3 stimulation protocol in Figure 1.26.

7. *APD restitution properties in the domain*: we consider the Bidomain equations (1.1) coupled with A-P ionic model (1.5). As highlighted in Section 1.1.1, as $c_1$ is decreased the restitution properties of the tissue changes and re-entry break-up happens. We set the parameter $c_1$ equal to 0.5 in a circular subregion of the domain, in order to assist to break-up, and elsewhere to $c_1 = 0.12$. The resulting solution is depicted in Figure 1.27 (left). The presence of the steep APD restitution curve region (shaded circular region) introduces disorder also in the part of the domain where we impose $c_1 = 0.12$, as confirmed by Figure 1.27 (right) in which we show the spectrum of the AP, computed

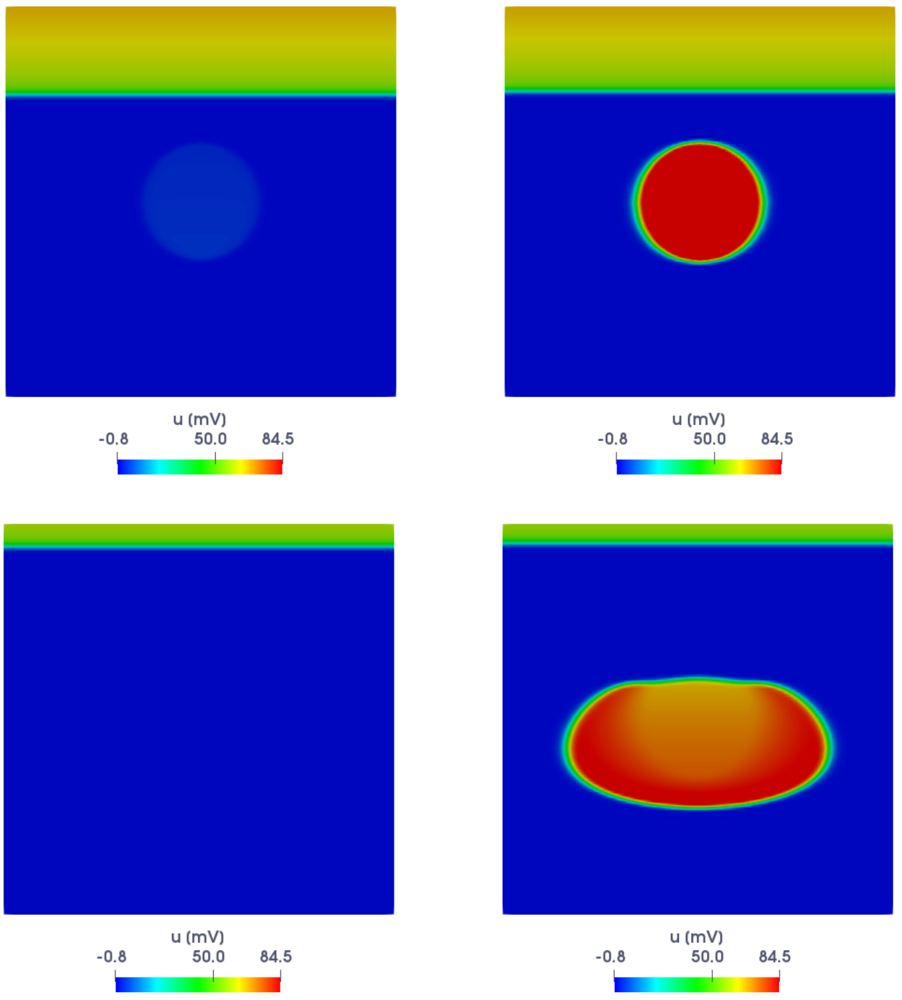


Figure 1.20: Figure of eight re-entry for $C = 10$ mA (left) and $C = 1000$ mA (right) at $t = 184$ ms (above) and $t = 204$ ms (bottom).

through FFT, at different points. In particular, we consider a point in the circular region, one close and one far from the subdomain. We also report the spectrum of the AP associated to a normal beat and to a stable spiral wave. The spectrum of the AP associated to the stable spiral wave shows a unique dominant frequency and the one related to the normal beat different dominant frequency of decreasing intensity. The results concerning the presence of different APD restitution properties in the domain show the same chaotic behavior, meaning that a subregion with a steep APD restitution curve is able to induce an irregular pattern in the remaining part of the tissue, that is to spread the disorder to the remaining part of the domain.

8. *Fibrosis*: we focus on equations (1.3) coupled with (1.5) and set $c_1 = 0.12$. We modify the intracellular conductivity tensor, by multiplying it with random numbers in the range $(0, 1)$ [Nagaiah et al., 2013], in order to model structural heterogeneities associated to fibrosis. As observed in [ten Tusscher, 2004], the presence of fibrosis suppresses spiral wave break-up, as shown in Figure 1.28. In presence of fibrosis, the spiral waves have a longer DI and lie on the part of the restitution curve that has a slope smaller than 1 and the spiral waves stay stable. Moreover, the presence of fibrosis increases the

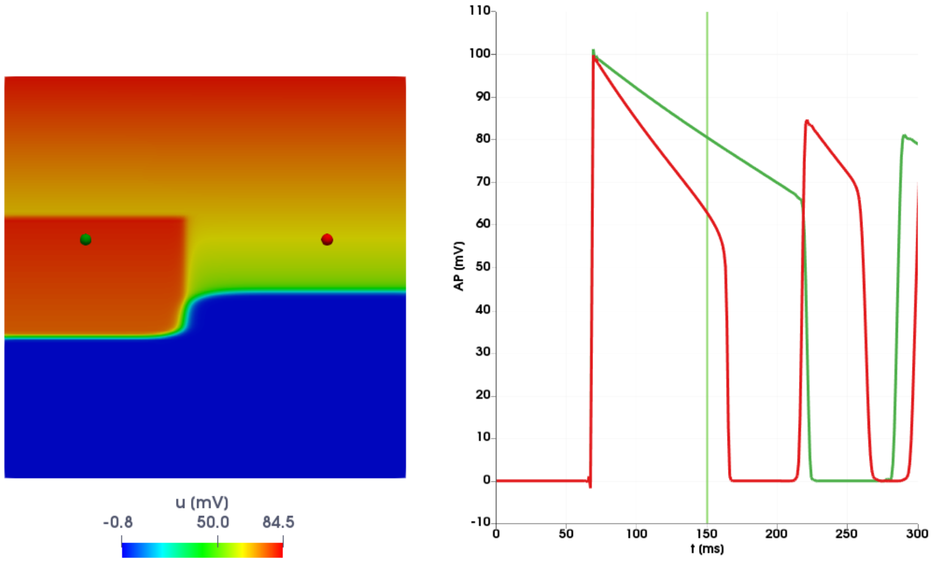


Figure 1.21: Left: Transmembrane potential at $t = 150$ ms. Right: APs for $P_1 = (0.49, 1.18)$ cm (green) and $P_2 = (1.61, 1.18)$ cm (red).

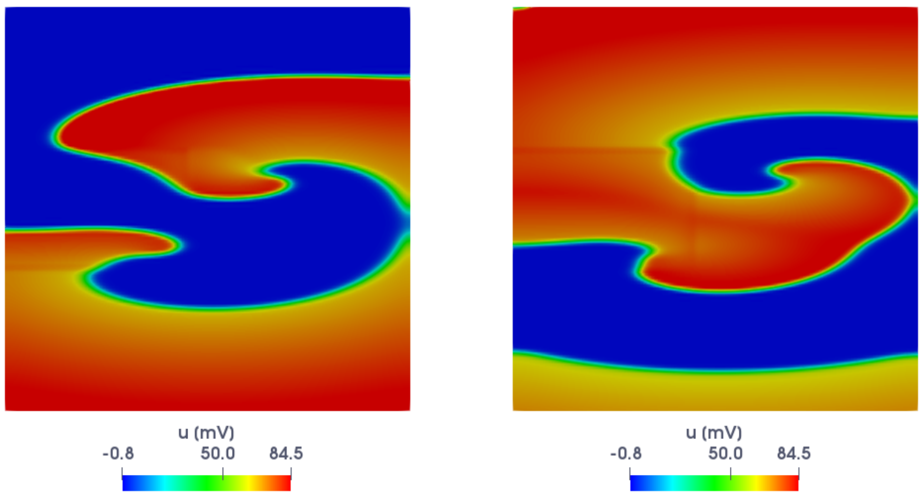


Figure 1.22: Figure of eight re-entry in a domain with different restitution properties at $t = 270$ ms (left) and $t = 310$ ms (right).

arrhytmogenic propensity of the tissue. Re-entries occur when their wavelength WL (WL = conduction velocity (CV) $\times$ RP [Qu et al., 1999]) is smaller than the length of the circuit. By setting $c_1$ equal to 0.14 and considering a healthy portion of tissue the re-entry extinguishes. The presence of fibrosis reduces both CV and RP and the re-entry is maintained, as shown in Figure 1.29.

9. *Inverted stimuli on LA*: we invert the two stimuli S1-S2. We first apply a physiological stimulus in correspondence of the posterior septum and then an ectopic complex arises in correspondence of one PV. It is clear that the location of the figure of eight re-entry is related to the location of the stimulus S2 and changes from the one in Section 1.4, as shown in Figure 1.30.

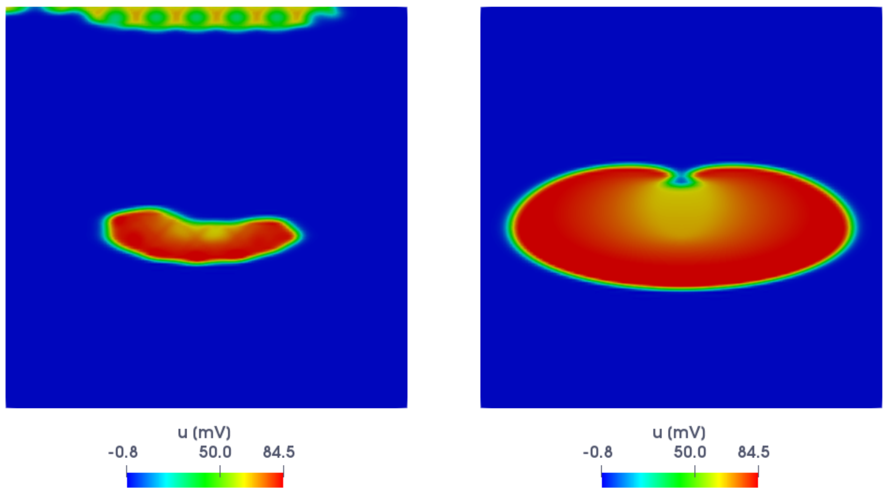


Figure 1.23: Figure of eight re-entry with dispersion of refractoriness (left) and in healthy tissue (right) at $t = 220$ ms.

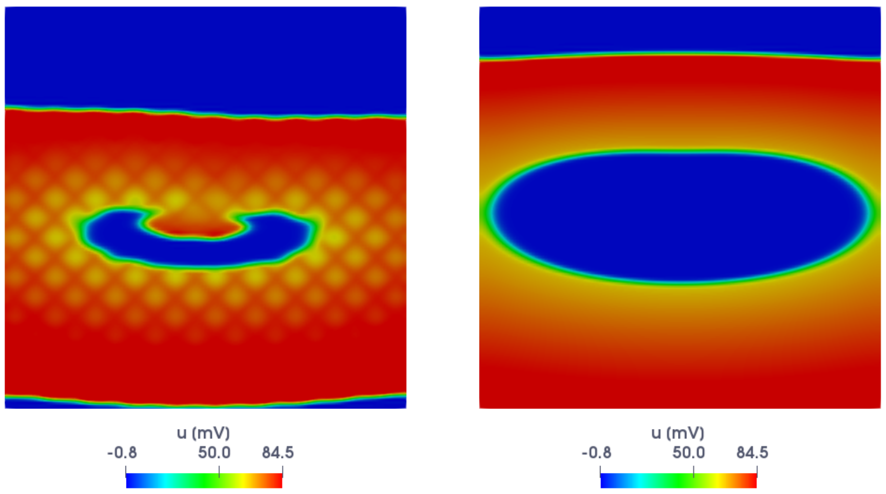


Figure 1.24: Figure of eight re-entry with dispersion of refractoriness (left) and in healthy tissue (right) at $t = 280$ ms.

10. *S1-S2 stimuli on PVs on LA*: we applied both the first S1 and the second S2 stimuli on the PVs. The resulting solution is reported in Figure 1.31.

11. *Ionic properties on LA*: we focus on equations (1.1) coupled with (1.5) and we vary the value of the ionic parameter $c_1$ from 0.05 to 0.03. By comparing Figure 1.32 with Figure 1.16 it is clear that break-up happens in both cases but the activation pattern is different.

12. *Re-entry break up on large final time $T$ on LA*: we consider again the Bidomain equations (1.1) coupled with A-P ionic model (1.5) and we perform a simulation by setting the final time $T = 10000$ in order to show that the re-entry break-up does not extinguish and the self-sustained activation pattern lasts until this large $T$. The results are shown in Figure 1.33.

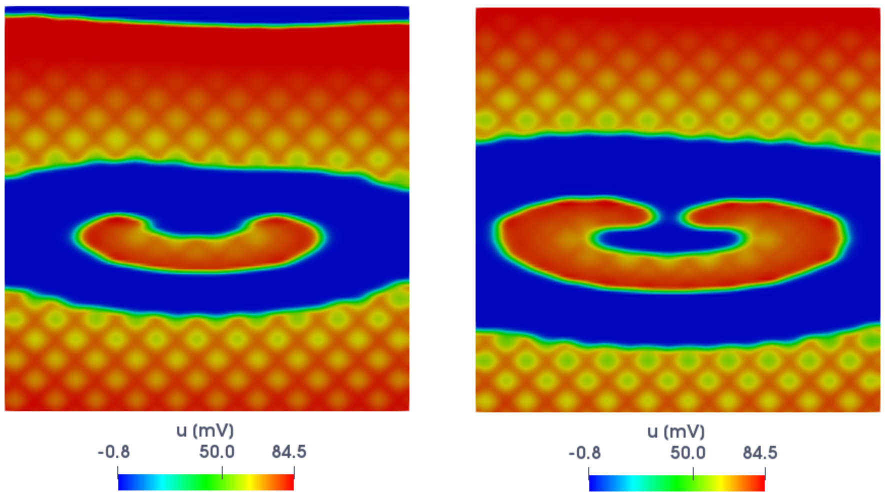


Figure 1.25: Figure of eight re-entry with dispersion of refractoriness at $t = 320$ ms (left) and $t = 340$ ms (right).

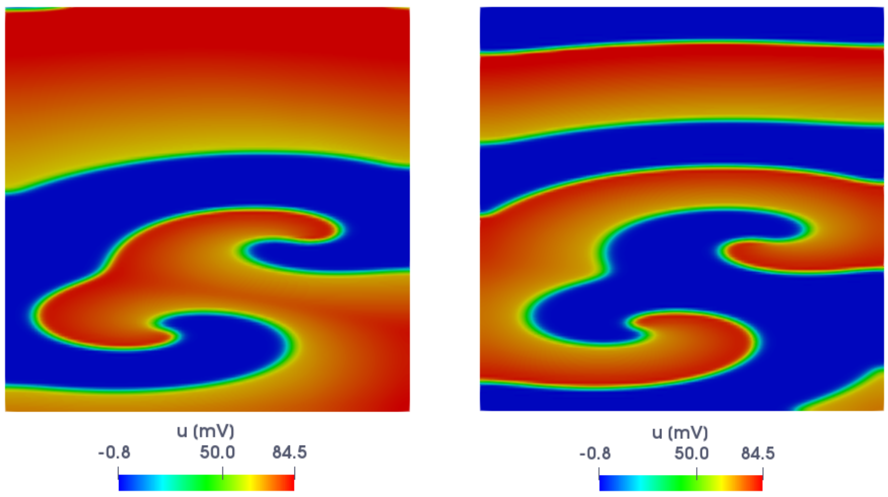


Figure 1.26: Figure of eight re-entry induced by the S1-S2-S3 protocol at $t = 310$ ms (left) and $t = 430$ ms (right).

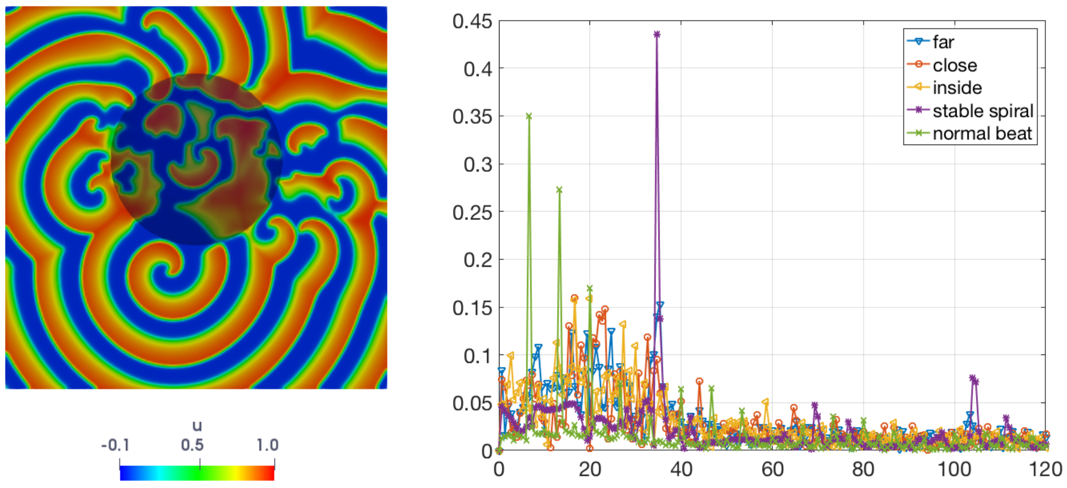


Figure 1.27: Left: Spiral wave with different APD restitution properties in the domain at $t = 1500$ (the shaded circular region is the portion of the domain in which a steep APD restitution is present). Right: Spectrum of the AP in different conditions.

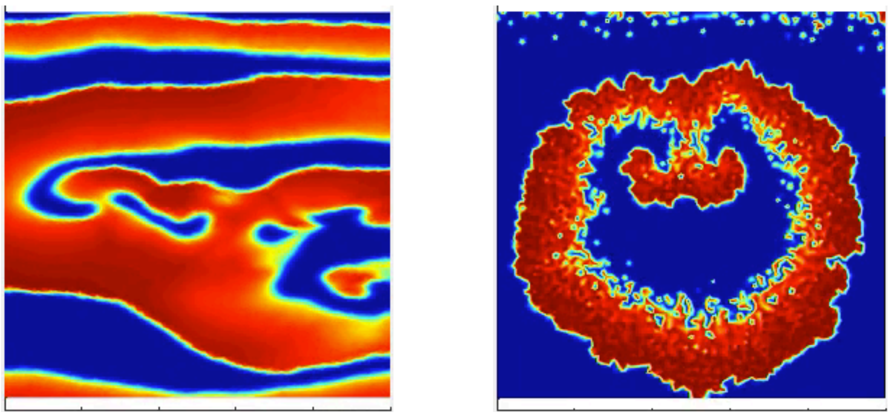


Figure 1.28: Re-entry break-up in a square slab of cardiac tissue without (left) and with fibrosis (right) for $c_1 = 0.12$ at $t = 380$ ms.

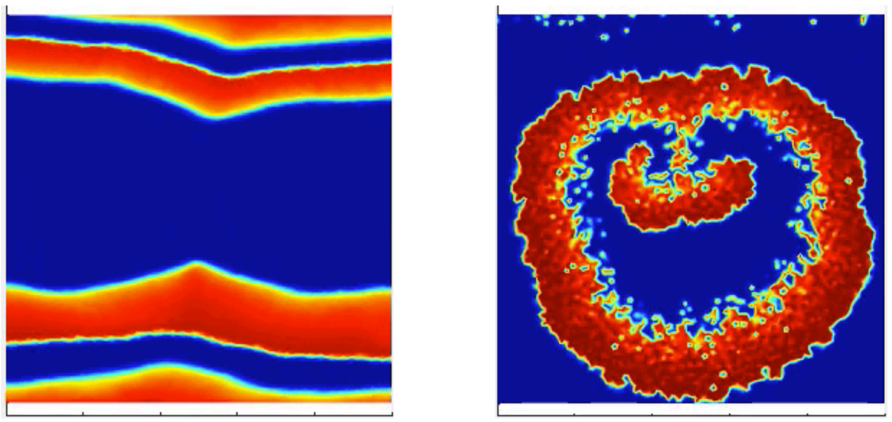


Figure 1.29: Re-entry break-up in a square slab of cardiac tissue without (left) and with fibrosis (right) for $c_1 = 0.14$ at $t = 380$ ms.

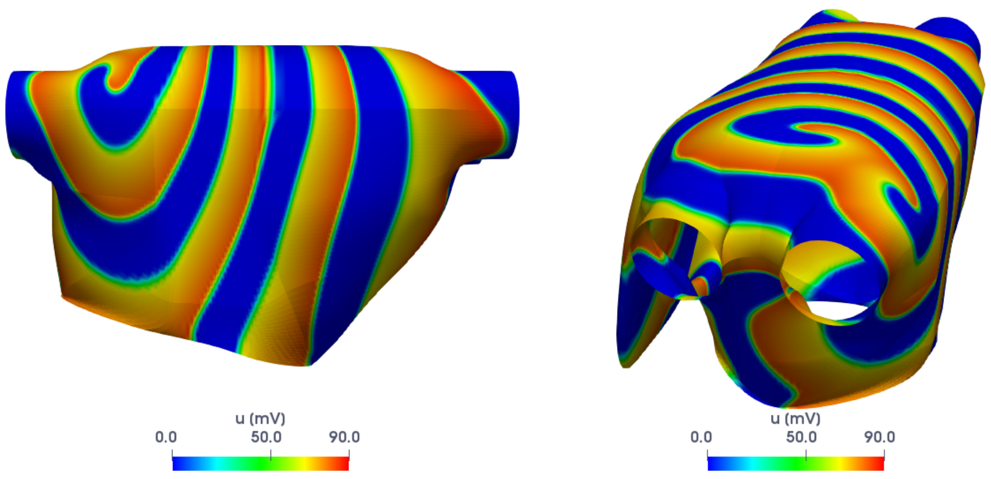


Figure 1.30: Figure of eight re-entry with inverted stimuli on an idealized LA geometry at $t = 800$ ms.

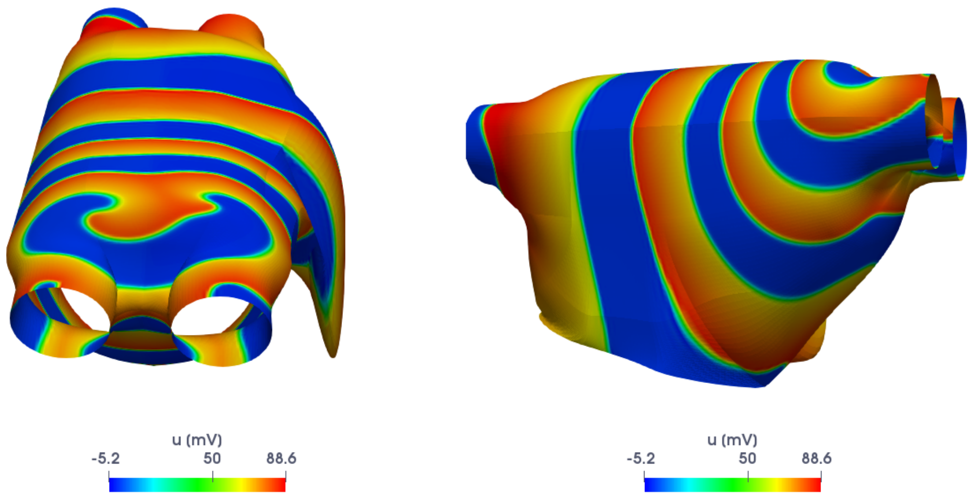


Figure 1.31: Figure of eight re-entry with both stimuli on PVs on an idealized LA geometry at $t = 564$ ms.

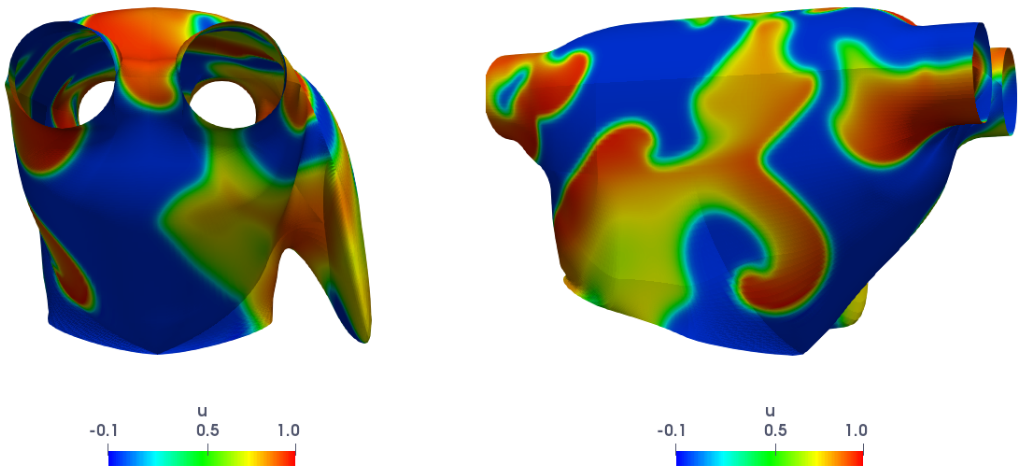


Figure 1.32: Re-entry break-up with $c_1 = 0.03$ on an idealized LA geometry at $t = 1200$.

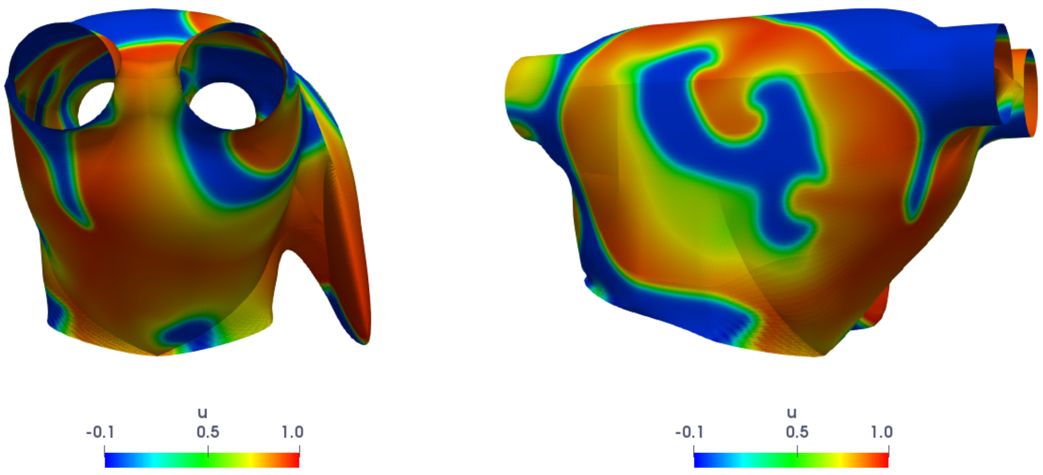


Figure 1.33: Re-entry break-up on an idealized LA geometry at $T = 10000$.

Chapter 2

# Reduced order modeling and deep learning

The aim of this Chapter is to provide the bases of reduced order modeling and deep learning (DL), cores of the methodologies developed in the following, for nonlinear time-dependent parametrized PDEs, such as cardiac electrophysiology (EP), in order to to achieve computational efficiency in the solution of systems (1.3) and (1.1). In particular, starting from the well-known setting of linear (projection-based) reduced order models (ROMs), we generalize this task to the case of nonlinear ROMs. We provide a quick overview of DL and of useful facts about neural networks. Moreover, an overview of those DL models, which the techniques proposed in Chapter 3 rely on, is provided together with an example of application consisting in the classification of atrial fibrillation (AF) starting from electrocardiograms (ECGs). The latter represents one of the most meaningful and successful examples of application of DL techniques in the cardiac EP field.

## 2.1 Reduced order modeling for parametrized PDEs

The solution of a parametrized system of PDEs by means of a *full order model* (FOM), such as the FE method and IGA, whenever dealing with real-time or multi-query scenarios, may entail prohibitive computational costs if the FOM is high-dimensional, as anticipated in Chapter 1. In the real-time context, the FOM solution must be computed in a very limited amount of time; in the multi-query one, the FOM must be solved for a huge number of parameter instances sampled from the parameter space. Cardiac EP problems detailed in Chapter 1 fit into real-time and multi-query contexts, due to the fact that, in view of the integration within the clinical practice, outputs of interest must be computed in short times, and, as outlined in Section 1.5, repetitive solutions of systems (1.3) and (1.1) are required in order to analyze different pathological scenarios.

Reduced order modeling techniques aim at replacing the FOM by a *reduced order model* (ROM), featuring a much lower dimension, however still able to express the physical features of the problem described by the FOM, thus being able to efficiently approximate the global map $(t, \boldsymbol{\mu}) \mapsto \mathbf{u}_h(t, \boldsymbol{\mu})$, where $t \in (0, T)$ denotes time, $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{n_\mu}$ a vector of input parameters and $\mathbf{u}_h(t, \boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ the solution of a dynamical system arising from the space discretization of a time-dependent (non)linear parametrized PDE, such as the one of systems (1.3) and (1.1). The basic assumption underlying the construction of such a ROM is that the solution of the parametrized PDE, belonging a priori to a high-dimensional (discrete) space, lies on a low-dimensional manifold embedded in this space. The goal of a ROM is then to approximate the *solution manifold* – that is, the set of all PDE solutions when the parameters vary in the parameter space – through a suitable, approximated *trial manifold.*

A widespread family of reduced order modeling techniques relies on the assumption that the reduced order approximation can be expressed by a linear combination of basis functions, built starting from a set of FOM solutions, called snapshots. Among these techniques, proper orthogonal decomposition (POD) – equivalent to principal component analysis in statistics [Hastie et al., 2001], or Karhunen-Loève expansion in stochastic applications – exploits the singular value decomposition (SVD) of a suitable snapshot matrix (or the eigen-decomposition of the corresponding snapshot correlation matrix), and the greedy algorithm [Prud'homme et al., 2001, Buffa et al., 2012] consists in an iterative procedure based on the evaluation of efficient estimates of the error between the FOM and the ROM, thus yielding *linear* ROMs, in which the ROM approximation is given by the linear superimposition of POD modes. In this case, the solution manifold is approximated through a *linear* trial manifold, that is, the ROM approximation is sought in a low-dimensional linear trial subspace.

Projection-based methods are linear ROMs in which the ROM approximation of the PDE solution, for any new parameter value, results from the solution of a low-dimensional (nonlinear, dynamical) system, whose unknowns are the ROM degrees of freedom (or generalized coordinates). Despite the PDE (and thus the FOM) being linear or not, the operators appearing in the ROM are obtained by imposing that the projection of the FOM residual evaluated on the ROM trial solution is orthogonal to a low-dimensional, linear test subspace, which might coincide with the trial subspace. Hence, the resulting ROM manifold is linear, that is, the ROM approximation is expressed as the linear combination of a set of basis functions. In particular, in projection-based ROMs, the reduced dynamics is obtained through a projection process onto a linear subspace [Benner et al., 2017, Benner et al., 2015, Quarteroni et al., 2016]. POD-Galerkin ROMs and greedy-Galerkin ROMs have been successfully applied in several contexts [Grepl and Patera, 2010, Veroy and Patera, 2005, Rozza et al., 2008] and to a broad range of applications, such as structural dynamics and elasticity [Willcox and Peraire, 2002, Amsallem et al., 2009, Bonomi et al., 2017], aerodynamics [Carlberg et al., 2013, Bui-Thanh et al., 2008, Bui-Thanh et al., 2004], cardiovascular fluid-dynamics [Manzoni et al., 2012, Colciago et al., 2014, Ballarin et al., 2016], and many other fields.

However, linear ROMs might experience computational bottlenecks at different extents when dealing with parametrized problems featuring coherent structures (possibly dependent on parameters) that propagate over time, namely in transport and wave-type phenomena, or convection-dominated flows, as soon as the physical behavior under analysis is strongly affected by parametric dependence. For larger parametric variations, or stronger dependence of coherent structures on parameters, the dimension of the linear trial manifold can easily become extremely large (if compared to the intrinsic dimension of the solution manifold for the sake of accuracy) thus compromising the ROM efficiency. The same difficulty may also affect (often expensive) hyper-reduction techniques, such as the (discrete) empirical interpolation (DEIM) [Barrault et al., 2004, Chaturantabut and Sorensen, 2010]. Such hyper-reduction techniques are essential to assemble the operators appearing in the ROM in order to not rely on expensive $N_h$-dimensional arrays, see, e.g., [Farhat et al., 2020] for further details. To overcome this issue, *ad-hoc* extensions of the POD strategy may bee considered, see, e.g., [Ohlberger and Rave, 2016, Pagani et al., 2018, Reiss et al., 2018]. Projections-based methods, such as, e.g, POD-Galerkin ROMs, are usually intrusive. In several applications the FOM is not directly accessible or deriving the reduced equations starting from the FOM entails high computational costs. For this reason, several recent works focus on non-intrusive, purely data-driven ROMs built by means of DL models. For example, in [González and Balajewicz, 2018] an autoencoder is used to compress the state and is followed by a recurrent neural network, [Carlberg et al., 2019] applies hierarchical dimensionality reduction comprising autoencoders and PCA followed by dynamics learning to recover missing CFD data and in [Takeishi et al., 2017, Lusch et al., 2018] autoencoders are applied to learn approximate invariant subspaces of the Koopman operator.

Our goal is to set up non-intrusive DL-based *nonlinear* ROMs, i.e. the solution manifold is approximated through a *nonlinear* trial manifold, whose dimension is nearly equal (if not equal) to the intrinsic dimension of the solution manifold that we aim at approximating. In this way, we want to overcome the critical issues of linear ROMs, e.g. projection-based ROMs, arising when applied to problems featuring wave-type phenomena and problems characterized by remarkable variability of the solution (with respect to the problem parameters), such as physiological and pathological cardiac EP.

## 2.2 Problem formulation

We formulate the construction of ROMs in algebraic terms, for the sake of generality, starting from the high-fidelity (spatial) approximation of nonlinear, time-dependent, parametrized PDEs; note that cardiac EP problems fall into this class. By introducing suitable space discretizations techniques, such as, e.g., the FE method or IGA (see Chapter 1 for further details), the high-fidelity, FOM can be expressed as a nonlinear parametrized dynamical system. Given $\boldsymbol{\mu} \in \mathcal{P}$, a vector of input parameters, we aim at solving the initial value problem

$$\begin{cases} \dot{\mathbf{u}}_h(t;\boldsymbol{\mu}) = \mathbf{f}(t, \mathbf{u}_h(t;\boldsymbol{\mu});\boldsymbol{\mu}) \quad t \in (0,T), \\ \mathbf{u}_h(0;\boldsymbol{\mu}) = \mathbf{u}_0(\boldsymbol{\mu}), \end{cases} \tag{2.1}$$

where the parameter space $\mathcal{P} \subset \mathbb{R}^{n_\mu}$ is a bounded and closed set, $\mathbf{u}_h : [0,T) \times \mathcal{P} \to \mathbb{R}^{N_h}$ is the parametrized solution of (2.1), $\mathbf{u}_0 : \mathcal{P} \to \mathbb{R}^{N_h}$ is the initial datum and $\mathbf{f} : (0,T) \times \mathbb{R}^{N_h} \times \mathcal{P} \to \mathbb{R}^{N_h}$ is a (nonlinear) function, encoding the system dynamics. The FOM dimension $N_h$ is related with the finite-dimensional subspace $X_h$ introduced in Section 1.3 and it can be extremely high whenever the PDE problem shows complex physical behaviors and/or high degrees of accuracy are required to its solution. The parameter $\boldsymbol{\mu} \in \mathcal{P}$ may represent physical or geometrical properties of the system, like, e.g., material properties, initial and boundary conditions, or the shape of the domain.

Our goal is the efficient numerical approximation of the whole set

$$\mathcal{S}_h = \{\mathbf{u}_h(t;\boldsymbol{\mu}) \mid t \in [0,T) \text{ and } \boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{n_\mu}\} \subset \mathbb{R}^{N_h}, \tag{2.2}$$

of solutions to problem (2.1) when $(t;\boldsymbol{\mu})$ varies in $[0,T) \times \mathcal{P}$, also referred to as solution manifold (a sketch is provided in Figure 2.2). Assuming that, for any given parameter $\boldsymbol{\mu} \in \mathcal{P}$,

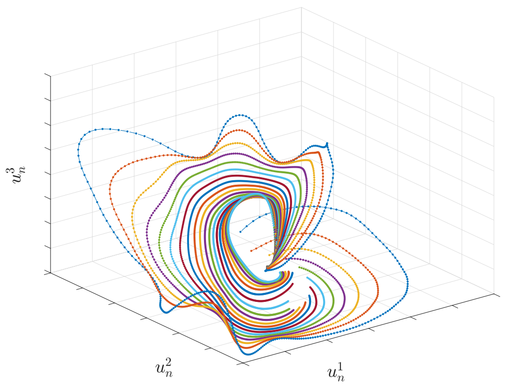


Figure 2.1: Example of a two-dimensional manifold embedded in $\mathbb{R}^3$. Each curve represents the time-evolution of the first three components of the solution of a (nonlinear) parametrized PDE for a fixed parameter value $\boldsymbol{\mu}$.

problem (2.1) admits a unique solution, for each $t \in (0,T)$, the intrinsic dimension of the solution manifold is at most $n_{\boldsymbol{\mu}}+1 \ll N_h$, where $n_{\boldsymbol{\mu}}$ is the number of parameters (time plays the role of an additional coordinate). This means that each point $\mathbf{u}_h(t;\boldsymbol{\mu})$ belonging to $\mathcal{S}_h$ is completely defined in terms of at most $n_{\boldsymbol{\mu}}+1$ intrinsic coordinates, or equivalently, the tangent space to the manifold at any given $\mathbf{u}_h(t;\boldsymbol{\mu})$ is spanned by $n_{\boldsymbol{\mu}}+1$ basis vectors.

## 2.3 Projection-based reduced order models

The most common way to build a ROM for the efficient approximation of problem (2.1) relies on the introduction of a *reduced linear trial manifold*, that is, a subspace $\tilde{\mathcal{S}}_n = \mathrm{Col}(\mathbf{V}) \subset \mathbb{R}^{N_h}$ of dimension $n \ll N_h$, spanned by the $n$ columns of a matrix $\mathbf{V} \in \mathbb{R}^{N_h \times n}$. Hence, what we can refer to as a linear ROM looks for an approximation $\tilde{\mathbf{u}}_h(t;\boldsymbol{\mu}) \approx \mathbf{u}_h(t;\boldsymbol{\mu})$ in the form

$$\tilde{\mathbf{u}}_h(t;\boldsymbol{\mu}) = \mathbf{V}\mathbf{u}_n(t;\boldsymbol{\mu}), \tag{2.3}$$

where $\tilde{\mathbf{u}}_h : [0,T) \times \mathcal{P} \to \tilde{\mathcal{S}}_n$. Here $\mathbf{u}_n(t;\boldsymbol{\mu}) \in \mathbb{R}^n$ for each $t \in [0,T)$, $\boldsymbol{\mu} \in \mathcal{P}$, denotes the vector of intrinsic coordinates (or degrees of freedom) of the ROM approximation; note that the map

$$\boldsymbol{\Psi}_h : \mathbb{R}^n \to \mathbb{R}^{N_h}, \qquad \mathbf{s}_n \mapsto \tilde{\mathbf{s}}_h = \mathbf{V}\mathbf{s}_n,$$

that, given the (low-dimensional) intrinsic coordinates, returns the (high-dimensional) approximation of the FOM solution $\mathbf{u}_h(t;\boldsymbol{\mu})$, is linear. The ROM approximation (2.3) is sought as a linear superimposition of modes, comparable to superposition of effects.

Proper orthogonal decomposition (POD) is one of the most widely employed techniques to generate the linear trial manifold [Fink and Rheinboldt, 1984]. Considering a set of $N_{train}$ instances of the parameter $\boldsymbol{\mu} \in \mathcal{P}$, we introduce the snapshot matrix $\mathbf{S} \in \mathbb{R}^{N_h \times N_s}$ defined as

$$\mathbf{S} = \left[\mathbf{u}_h(t^1;\boldsymbol{\mu}_1) \mid \ldots \mid \mathbf{u}_h(t^{N_t};\boldsymbol{\mu}_1) \mid \ldots \mid \mathbf{u}_h(t^1;\boldsymbol{\mu}_{N_{train}}) \mid \ldots \mid \mathbf{u}_h(t^{N_t};\boldsymbol{\mu}_{N_{train}})\right], \tag{2.4}$$

where we have introduced a partition of the time interval $[0,T]$ in $N_t$ time steps $\{t^k\}_{k=1}^{N_t}$, $t^k = k\Delta t$, of size $\Delta t = T/N_t$ and $N_s = N_{train}N_t$. Moreover, let us introduce a symmetric and positive definite matrix $\mathbf{X}_h \in \mathbb{R}^{N_h \times N_h}$ encoding a suitable norm (e.g., the energy norm $||\cdot||_{\mathbf{X}_h} = ||\mathbf{X}_h^{1/2}\cdot||_2$) on the high-dimensional space and admitting a Cholesky factorization $\mathbf{X}_h = \mathbf{H}^T\mathbf{H}$. POD computes the SVD of $\mathbf{HS}$,

$$\mathbf{HS} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{Z}^T,$$

where $\mathbf{U} = [\boldsymbol{\zeta}_1|\ldots|\boldsymbol{\zeta}_{N_h}] \in \mathbb{R}^{N_h \times N_h}$, $\mathbf{Z} = [\boldsymbol{\psi}_1|\ldots|\boldsymbol{\psi}_{N_s}] \in \mathbb{R}^{N_s \times N_s}$ and $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1,\ldots,\sigma_r) \in \mathbb{R}^{N_h \times N_s}$ with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$, and $r \leq \min(N_h, N_s)$, and sets the columns of $\mathbf{V}$ in terms of the first $n$ left singular vectors of $\mathbf{S}$ that is, $\mathbf{V} = [\mathbf{H}^{-1}\boldsymbol{\zeta}_1|\ldots|\mathbf{H}^{-1}\boldsymbol{\zeta}_n]$. By construction, the columns of $\mathbf{V}$ are orthonormal (with respect to the scalar product $(\cdot,\cdot)_{X_h}$) and among all possible $n$-dimensional subspaces spanned by the column of a matrix $\mathbf{W} \in \mathbb{R}^{N_h \times n}$, $\mathbf{V}$ provides the best reconstruction of the snapshots, that is,

$$\begin{aligned}&\sum_{i=1}^{N_{train}}\sum_{k=1}^{N_t} \|\mathbf{u}_h(t^k;\boldsymbol{\mu}_i) - \mathbf{V}\mathbf{V}^T\mathbf{X}_h\mathbf{u}_h(t^k;\boldsymbol{\mu}_i)\|^2_{\mathbf{X}_h} = \\ &\min_{W \in \mathcal{V}_n} \sum_{i=1}^{N_{train}}\sum_{k=1}^{N_t} \|\mathbf{u}_h(t^k;\boldsymbol{\mu}_i) - \mathbf{W}\mathbf{W}^T\mathbf{X}_h\mathbf{u}_h(t^k;\boldsymbol{\mu}_i)\|^2_{\mathbf{X}_h},\end{aligned} \tag{2.5}$$

where $\mathcal{V}_n = \{\mathbf{W} \in \mathbb{R}^{N_h \times n} : \mathbf{W}^T\mathbf{X}_h\mathbf{W} = \mathbf{I}\}$. For this reason, we refer to $\mathbf{V}\mathbf{V}^T\mathbf{X}_h\mathbf{u}_h(t;\boldsymbol{\mu})$ as to the optimal-POD reconstruction, that is, the projection of $\mathbf{u}_h(t;\boldsymbol{\mu})$ onto the reduced subspace $\tilde{\mathcal{S}}_n$ of dimension $n < N_h$.

In order to model the reduced dynamics of the system, that is, the time evolution of the generalized coordinates $\mathbf{u}_n(t;\boldsymbol{\mu})$, we can replace $\mathbf{u}_h(t;\boldsymbol{\mu})$ by (2.3) in system (2.1)

$$\begin{cases} \mathbf{V}\dot{\mathbf{u}}_n(t;\boldsymbol{\mu}) = \mathbf{f}(t, \mathbf{V}\mathbf{u}_n(t;\boldsymbol{\mu});\boldsymbol{\mu}) & t \in (0,T), \\ \mathbf{V}\mathbf{u}_n(0;\boldsymbol{\mu}) = \mathbf{u}_0(\boldsymbol{\mu}), \end{cases} \tag{2.6}$$

and impose that the residual

$$\mathbf{r}_h(\mathbf{V}\mathbf{u}_n(t;\boldsymbol{\mu})) = \mathbf{V}\dot{\mathbf{u}}_n(t;\boldsymbol{\mu}) - \mathbf{f}(t, \mathbf{V}\mathbf{u}_n(t;\boldsymbol{\mu});\boldsymbol{\mu}) \tag{2.7}$$

associated to the first equation of (2.6) is orthogonal to an $n$-dimensional subspace spanned by the column of a matrix $\mathbf{Y} \in \mathbb{R}^{N_h \times n}$, that is

$$\mathbf{Y}^T \mathbf{r}_h(\mathbf{V}\mathbf{u}_n) = \mathbf{0}.$$

This condition yields the following ROM

$$\begin{cases} \mathbf{Y}^T\mathbf{V}\dot{\mathbf{u}}_n(t;\boldsymbol{\mu}) = \mathbf{Y}^T\mathbf{f}(t, \mathbf{V}\mathbf{u}_n(t;\boldsymbol{\mu});\boldsymbol{\mu}) & t \in (0,T), \\ \mathbf{u}_n(0;\boldsymbol{\mu}) = (\mathbf{Y}^T\mathbf{V})^{-1}\mathbf{Y}^T\mathbf{u}_0(\boldsymbol{\mu}). \end{cases} \tag{2.8}$$

In the case $\mathbf{Y} = \mathbf{V}$, a Galerkin projection is performed, while the case $\mathbf{Y} \neq \mathbf{V}$ yields a more general Petrov-Galerkin projection. Note that choosing $\mathbf{Y}$ such that $\mathbf{Y}^T\mathbf{V} = \mathbf{I} \in \mathbb{R}^{N_h \times N_h}$ does not automatically ensure ROM stability on long time intervals.

The RB method under the form of either POD-Galerkin or POD-Petrov-Galerkin methods has been successfully applied to a broad range of parametrized time-dependent (non)linear problems (see, e.g., [Pagani et al., 2018, Manzoni et al., 2016]) however it provides low-dimensional subspaces of dimension $n \gg n_\mu + 1$ much larger than the intrinsic dimension of the solution manifold – relying on a linear, global trial manifold thus represent a major bottleneck to computational efficiency [Ohlberger and Rave, 2016, Pagani et al., 2018]. This is the case, for instance, of hyperbolic problems, for which the RB method is not able to significantly decrease the dimensionality of the problem. In the following we refer to ROMs built by means of the RB method in the POD-Galerkin form as POD-Galerkin ROMs.

### 2.3.1 Hyper-reduction techniques

The efficient evaluation of the ROM arrays appearing in (2.8) as time and parameters vary is still a challenging task in order to achieve an efficient online, i.e. testing, evaluation of a ROM when dealing with nonlinear and/or complex nonaffine terms, i.e. terms depending nonlinearly on the input parameters of the problem. The function $\mathbf{f} : (0,T) \times \mathbb{R}^{N_h} \times \mathcal{P} \to \mathbb{R}^{N_h}$ in (2.1) can be seen as the sum of linear and nonlinear operators, that is

$$\mathbf{f}(t, \mathbf{u}_h(t;\boldsymbol{\mu});\boldsymbol{\mu}) = \mathbf{L}(t;\boldsymbol{\mu})\mathbf{u}_h(t;\boldsymbol{\mu}) + \mathbf{N}(t, \mathbf{u}_h(t;\boldsymbol{\mu});\boldsymbol{\mu}). \tag{2.9}$$

When dealing with nonlinear operators $\mathbf{N}(\cdot,\cdot;\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$, evaluating $\mathbf{V}^T\mathbf{N}(\cdot,\cdot;\boldsymbol{\mu})$ depends on the FOM dimension $N_h$; indeed, at each time step, its assembly relies on high-dimensional arrays and this would be a very expensive task compromising the overall efficiency of the ROM. To overcome this problem, the (discrete) empirical interpolation method (DEIM) can be exploited at each time step to handle the nonaffine $\boldsymbol{\mu}$-dependent and nonlinear terms efficiently, as proposed in [Chaturantabut and Sorensen, 2010, Maday et al., 2008]. It allows to speed up the evaluation of nonlinear and nonaffine arrays, avoiding to access the FOM structures and ensuring the overall ROM efficiency. In particular, the DEIM approximation of a nonlinear operator is computed through the following steps, see, e.g., [Farhat et al., 2020]:

- compute the snapshot matrix of the nonlinear term $\mathbf{N}$

$$\mathbf{S}_N = \left[\mathbf{N}(t^1, \mathbf{u}_h(t^1;\boldsymbol{\mu}_1);\boldsymbol{\mu}_1)| \ \dots \ |\mathbf{N}(t^{N_t}, \mathbf{u}_h(t^{N_t};\boldsymbol{\mu}_{N_{train}}), \boldsymbol{\mu}_{N_{train}})\right], \tag{2.10}$$

- compute the matrix of basis functions $\boldsymbol{\Phi} = [\phi_1, \dots, \phi_m] \in \mathbb{R}^{N_h \times m}$ by applying POD to $\mathbf{S}_N$,
- select iteratively $m$ interpolation indexes $\mathcal{I} \subset \{1, \dots, N_h\}$, with $|\mathcal{I}| = m$, from the basis $\boldsymbol{\Phi}$ according to a suitable greedy procedure, which minimizes at each step the interpolation error over the snapshots set measured in the maximum norm (see [Maday et al., 2008] for further details); in particular, the index matrix is built

$$\mathbf{P} = [\mathbf{e}_1, \dots, \mathbf{e}_m] \qquad (\mathbf{e}_i)_j = \delta_{ij}.$$

Given a new $\boldsymbol{\mu}$ and the associated intrinsic coordinates $\mathbf{u}_n(t^k;\boldsymbol{\mu})$ at a given time instant, during the online phase, we can compute $\overline{\mathbf{N}}(t^k, \mathbf{u}_h(t^k;\boldsymbol{\mu});\boldsymbol{\mu})$ as

$$\overline{\mathbf{N}}(t^k, \mathbf{u}_h(t^k;\boldsymbol{\mu});\boldsymbol{\mu}) = \boldsymbol{\Phi}\boldsymbol{\theta}(t^k;\boldsymbol{\mu}) \qquad \text{with} \qquad \boldsymbol{\Phi}_{\mathcal{I}}\boldsymbol{\theta}(t^k;\boldsymbol{\mu}) = \mathbf{N}_{\mathcal{I}}(t^k, \mathbf{V}\mathbf{u}_n(t^k;\boldsymbol{\mu});\boldsymbol{\mu}), \tag{2.11}$$

where $\boldsymbol{\Phi}_{\mathcal{I}}$ and $\mathbf{N}_{\mathcal{I}}(t^k, \mathbf{V}\mathbf{u}_n(t^k;\boldsymbol{\mu});\boldsymbol{\mu})$ denote the matrix formed by the $\mathcal{I}$ rows of $\boldsymbol{\Phi}$ and the vector $\mathbf{N}(t^k, \mathbf{V}\mathbf{u}_n(t^k;\boldsymbol{\mu});\boldsymbol{\mu})$ evaluated at the $\mathcal{I}$ entries, respectively. The vector $\boldsymbol{\theta}(t^k;\boldsymbol{\mu}) = [\theta_1(t^k;\boldsymbol{\mu}), \dots, \theta_m(t^k;\boldsymbol{\mu})] \in \mathbb{R}^m$ is evaluated by solving the linear system in (2.11), encoding $m$ interpolation constraints at the mesh points selected in $\mathcal{I}$. We can express

$$\boldsymbol{\Phi}_{\mathcal{I}} = \mathbf{P}^T\boldsymbol{\Phi}, \qquad \mathbf{N}_{\mathcal{I}}(t^k, \mathbf{V}\mathbf{u}_n(t^k;\boldsymbol{\mu});\boldsymbol{\mu}) = \mathbf{N}(t^k, \mathbf{P}^T\mathbf{V}\mathbf{u}_n(t^k;\boldsymbol{\mu});\boldsymbol{\mu}).$$

In conclusion, the DEIM approximation reads

$$\mathbf{V}^T\mathbf{N}(t^k, \mathbf{V}\mathbf{u}_n(t^k;\boldsymbol{\mu});\boldsymbol{\mu}) \approx \underbrace{\mathbf{V}^T\boldsymbol{\Phi}(\mathbf{P}^T\boldsymbol{\Phi})^{-1}}_{n\times m}\underbrace{\mathbf{N}(t^k, \mathbf{P}^T\mathbf{V}\mathbf{u}_n(t^k;\boldsymbol{\mu});\boldsymbol{\mu})}_{m\times 1}.$$

We emphasize that $\mathbf{P}$ selects only a subset of indexes from the FOM solution; this means that we do not need to assemble nonlinear operators on the entire mesh, but only on the elements related to the degrees of freedom selected by the DEIM algorithm.

In those cases where the linear operators $\mathbf{L}(\cdot;\boldsymbol{\mu})$ $\boldsymbol{\mu}$-dependence is nonaffine, it is possible to rely on DEIM or matrix DEIM (MDEIM) [Negri et al., 2015, Bonomi et al., 2017] to get an approximated affine expansion.

We remark that hyper-reduction techniques entail intrusive changes to the FOM implementation of the problem and linear subspaces whose dimension $m$ may become very large, in order to provide an approximation to FOM arrays sufficiently accurate, thus leading to inefficient ROMs.

### 2.3.2 Local POD-Galerkin ROMs

Problems featuring a travelling wave behavior, such as cardiac EP, might easily yield solutions showing a remarkable variability over the parameter space. This might causes the solution manifold (2.2) to be highly nonlinear; as a matter of fact, its (linear) approximation by means of a single linear subspace yields accurate approximation only at the price of considering very large dimensions $n$ and $m$ of the POD expansion and of the DEIM approximation, respectively, thus preventing the ROM from ensuring a considerable speed-up compared to the FOM.

A first attempt to overcome the computational bottleneck entailed by the use of a linear, global trial manifold is to build a piecewise linear trial manifold, using local reduced bases whose dimension is smaller than the one of the global linear trial manifold. Clustering algorithms applied on a set of snapshots can be employed to partition them into $N_c$ clusters from which POD can extract a subspace of reduced dimension; the ROM is then obtained by following the strategy described above on each cluster separately, see, e.g. [Amsallem et al., 2012, Amsallem et al., 2015]. In particular, multiple local subspaces can be generated when performing the POD-Galerkin ROM approximation of the PDE solution, and the DEIM approximation can be used for terms depending nonlinearly on either the solution or the input parameters of the problem. Approximating the manifold by a series of subspaces of smaller dimension results in a more efficient approach than building a single subspace of larger dimension. With this aim, we employ a four-step procedure as proposed in [Amsallem et al., 2012]:

1. we compute the snapshot matrix $\mathbf{S} \in \mathbb{R}^{N_h \times N_s}$ by solving the FOM over time for suitably chosen training-parameter instances;

2. for a given $N_c$ we group snapshots into clusters $\mathbf{S}^k$, $k = 1, \ldots, N_c$; each column of $\mathbf{S}$ is thus assigned to a cluster accordingly to the *k-means* clustering algorithm [Likas et al., 2003];

3. we construct a local reduced basis for each cluster through POD;

4. we construct a ROM for each cluster by projecting the original FOM onto each reduced subspace $\mathbf{V}_k$, $k = 1, \ldots, N_c$, as in the classical POD-Galerkin ROM.

As soon as the local ROMs for the solution of the PDE have been built, snapshots of the nonlinear terms are computed to form the matrix $\mathbf{S}_N$ defined in (2.10). Then, also the columns of $\mathbf{S}_N$ are partitioned into $N_c$ clusters $\mathbf{S}_N^k$, $k = 1, \ldots, N_c$, and the DEIM procedure of Subsection 2.3.1 is applied to each cluster, yielding a set of $N_c$ bases $\mathbf{\Phi}_k$, $k = 1, \ldots, N_c$, to be used to approximate nonlinear terms efficiently. The same number $N_c$ of clusters is chosen for both approximations, although in principle a different number of clusters could be selected to partition both column sets $\mathbf{S}$ and $\mathbf{S}_N$.

The online query to the local POD-Galerkin ROM is then performed by exploiting the reduced matrices and vectors, as well as the local DEIM approximation associated to the reduced subspace selected at step 2; note that switching from a local ROM to a new one must be done inexpensively during the online stage. Neighboring snapshots can be either added or not to each cluster to obtain overlapping clusters; we do not consider any overlap among clusters, although in principle this can also be done [Amsallem et al., 2012, Amsallem and Haasdonk, 2016].

The approach described above was firstly proposed in [Amsallem et al., 2012] to address the construction of local ROMs in the state space – although without constructing a ROM to be systematically queried over the parameter space – and further extended in [Amsallem et al., 2015, Amsallem and Haasdonk, 2016]. It was also applied in [Peherstorfer et al., 2014] for the sake of approximating nonlinear quantities by DEIM.

An alternative approach is based on classification binary trees and has been introduced in [Amsallem and Haasdonk, 2016]. It has been employed (and compared) in [Pagani et al., 2018] in order to solve parametrized problems in cardiac EP, showing that the approach based on the k-means clustering algorithm improves the approximation accuracy of the ROM.

## 2.4 Nonlinear dimensionality reduction

Using a piecewise linear trial manifold partially overcomes the limitation of a linear dimensionality reduction technique as POD, yet employing local bases of dimension much higher than the intrinsic dimension of the solution manifold $\mathcal{S}_h$. An approach based on a dictionary of solutions, computed offline, has been developed in [Abgrall et al., 2016] as an alternative to using a truncated reduced basis based on POD, together with an online $L^1$-norm minimization of the residual.

Other possible options involving nonlinear transformations of modes might rely on a reconstruction of the POD modes at each time step using Lax pairs [Gerbeau and Lombardi, 2014], on the solution of Monge-Kantorovich optimal transport problems [Iollo and Lombardi, 2014], on a problem-dependent change of coordinates requiring the solution of an optimization problem repeatedly [Cagniart et al., 2019], on shifted POD modes [Reiss et al., 2018] after multiple transport velocities have been identified and separated, or again basis updates are derived from querying the full model at a few selected spatial coordinates [Peherstorfer, 2018].

Despite providing remarkable improvements compared to the classic (Petrov-)Galerkin-POD approach, all these strategies exhibit some drawbacks, such as: *(i)* the high computational costs entailed during the online testing evaluation stage of the ROM – which is not restricted to the intensive offline training stage; *(ii)* performance and settings are highly dependent on the problem at hand; *(iii)* the need to deal only with a linear superimposition of modes (which characterizes linear ROMs), yielding low-dimensional spaces whose dimension is still (much) higher than the intrinsic dimension of the solution manifold.

Motivated by the need of avoiding the drawbacks of linear ROMs and setting a general paradigm for the construction of efficient, extremely low-dimensional ROMs, we resort to nonlinear dimensionality reduction techniques.

We build a nonlinear ROM to approximate $\mathbf{u}_h(t;\boldsymbol{\mu}) \approx \tilde{\mathbf{u}}_h(t;\boldsymbol{\mu})$ by

$$\tilde{\mathbf{u}}_h(t;\boldsymbol{\mu}) = \boldsymbol{\Psi}_h(\mathbf{u}_n(t;\boldsymbol{\mu})), \tag{2.12}$$

where $\boldsymbol{\Psi}_h : \mathbb{R}^n \to \mathbb{R}^{N_h}$, $\boldsymbol{\Psi}_h : \mathbf{s}_n \mapsto \boldsymbol{\Psi}_h(\mathbf{s}_n)$, $n \ll N_h$, is a nonlinear, differentiable function; similar approaches can be found in [González and Balajewicz, 2018, Lee and Carlberg, 2020]. As a matter of fact, the solution manifold $\mathcal{S}_h$ is approximated by a *reduced nonlinear trial manifold*

$$\tilde{\mathcal{S}}_n = \{\boldsymbol{\Psi}_h(\mathbf{u}_n(t;\boldsymbol{\mu})) \mid \mathbf{u}_n(t;\boldsymbol{\mu}) \in \mathbb{R}^n,\ t \in [0,T) \text{ and } \boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{n_\mu}\} \subset \mathbb{R}^{N_h} \tag{2.13}$$

so that $\tilde{\mathbf{u}}_h : [0,T) \times \mathcal{P} \to \tilde{\mathcal{S}}_n$. As before, $\mathbf{u}_n : [0,T) \times \mathcal{P} \to \mathbb{R}^n$ denotes the vector-valued function of two arguments representing the intrinsic coordinates of the ROM approximation. Our goal is to set a ROM whose dimension $n$ is as close as possible to the intrinsic dimension $n_{\boldsymbol{\mu}}+1$ of the solution manifold $\mathcal{S}_h$, i.e. $n \geq n_{\boldsymbol{\mu}}+1$, in order to correctly capture the solution of the dynamical system by containing the size of the approximation spaces [Lee and Carlberg, 2020].

To model the relationship between each pair $(t,\boldsymbol{\mu}) \to \mathbf{u}_n(t,\boldsymbol{\mu})$, and to describe the system dynamics on the reduced nonlinear trial manifold $\tilde{\mathcal{S}}_n$ in terms of the intrinsic coordinates, we consider a nonlinear map under the form

$$\mathbf{u}_n(t;\boldsymbol{\mu}) = \boldsymbol{\Phi}_n(t,\boldsymbol{\mu}), \tag{2.14}$$

where $\boldsymbol{\Phi}_n : [0,T) \times \mathbb{R}^{n_\mu} \to \mathbb{R}^n$ is a differentiable, nonlinear function. No additional assumptions such as, e.g., the (exact, or approximate) affine $\boldsymbol{\mu}$-dependence as in the POD-Galerkin ROM, are needed, thus avoiding (intrusive and often expensive) hyper-reduction techniques.

## 2.5 Basic concepts of deep learning

Artificial intelligence (AI) refers to the ability of computers to imitate typical human intelligence behaviors, to carry out tasks requiring human intelligence [Russell and Norvig, 2009]. This may be achieved by explicitly programming a machine to solve a given problem described by means of formal instructions. Sometimes it is not possible to formally describe the task to carry out. The AI systems ability to acquire their own knowledge, by extracting patterns from raw data, is called machine learning (ML). In this case computers are not explicitly programmed to solve a specific task but they learn from experience. A formal definition of ML is provided in [Mitchell, 1997]: a computer program is said to learn from experience $E$, with respect to some class of task $T$ and a performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves because of experience $E$.

Deep learning (DL) is a particular kind of ML in which the computer also learns a data representation expressed as a hierarchy of nested concepts more and more complex and abstract. For example, if we are interested in solving a classification problem, given the two categories person and animal, starting from images, the computer has to learn the map associating to a set of pixel one identity. A DL system represents the concept of image of a person by combining simpler concepts as corners and contours, defined in terms of edges, identified starting from the pixel of the image, the raw data (see Figure 2.2 for the illustration of such DL system).

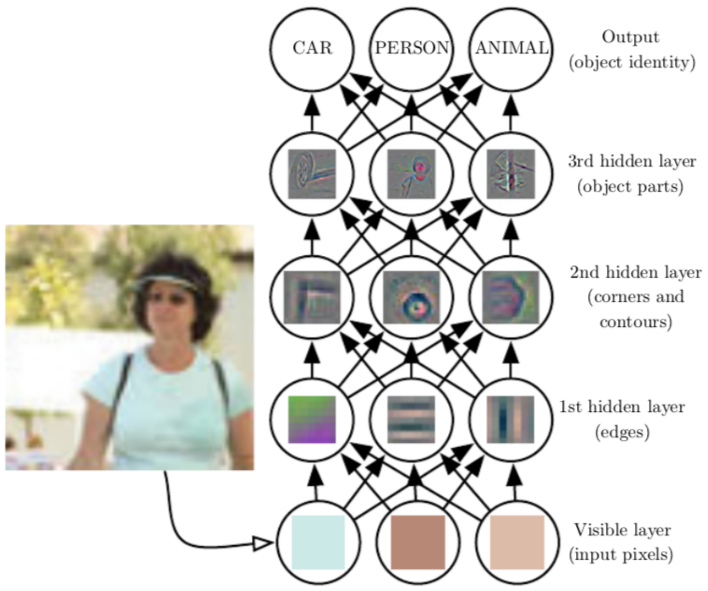


Figure 2.2: Illustration of a DL model [Goodfellow et al., 2016].

DL techniques have gained great attention in recent years in several areas like computer vision [Krizhevsky et al., 2012, Antipov et al., 2017], natural language processing [Sutskever et al., 2014, Devlin et al., 2018] and speech recognition [Bourlard and Wellekens, 1989, Chung et al., 2017], due to their ability to discover pattern and extract features from massive datasets, in order to make predictions without providing hand-crafted features.

Many kinds of tasks can be solved with ML and DL. The most common are [Goodfellow et al., 2016]

- *classification*: the computer program is asked to specify which of $k$ categories some input belongs to. To solve this task, the learning algorithm is usually asked to produce a function $f : \mathbb{R}^n \to \{1, \dots, k\}$. When $y = f(\mathbf{x})$, the model assigns an input described by vector $\mathbf{x}$ to a category identified by a numeric code $y$, or the probability distribution over classes;

- *regression*: the computer program is asked to predict a numerical value given some input. To solve this task, the learning algorithm is asked to output a function $f : \mathbb{R}^n \to \mathbb{R}$.

ML and DL algorithms can be broadly categorized following three paradigms. Provided a certain experience $E$, i.e. a dataset $\{\mathbf{x}^1, \dots, \mathbf{x}^M\}$, the kind of experience they are allowed to have during the learning process can be divided in

- *supervised learning*: given the desired outputs $y^1, \ldots, y^M$, learn to produce the correct outputs given a new set of inputs;
- *unsupervised learning*: exploit regularities in $\{\mathbf{x}^1, \ldots, \mathbf{x}^M\}$ to build a representation to be used for reasoning or prediction;
- *reinforcement learning*: producing actions $a^1, \ldots, a^M$, which affect the environment, and receiving rewards $r^1, \ldots, r^M$, learn to act in order to maximize rewards in the long term.

In supervised learning problems, three datasets are commonly used in the different stages of the definition of the model

- *training set*: data used to fit the parameters of the model,
- *validation set*: data used to estimate the generalization error during training, allowing for the hyperparameters to be updated accordingly,
- *testing set*: data not seen during the training phase and used to provide an unbiased evaluation of the performance of the model.

In this work, we focus on supervised and unsupervised learning, by carrying out regression tasks, and, in the rest of this Section, we provide an overview of those DL models which the techniques proposed in the following rely on.

### 2.5.1 Deep feedforward neural networks

A remarkable example of DL model is the deep feedforward neural network (DFNN). A DFNN is a mathematical function modeling the relationship between a set of input values and some output values [Goodfellow et al., 2016]. This mathematical function is obtained through composition of simpler (nonlinear) functions, or layers, and allows to learn complex hierarchies of features. More formally, provided an input $\mathbf{x} \in \mathbb{R}^{N_0}$ a DFNN with $L$ layers takes the form

$$\boldsymbol{\phi}^{DF} : (\mathbf{x}; \boldsymbol{\theta}_{DF}) \mapsto \boldsymbol{\phi}_L(\cdot; \boldsymbol{\theta}_L) \circ \boldsymbol{\phi}_{L-1}(\cdot; \boldsymbol{\theta}_{L-1}) \circ \ldots \circ \boldsymbol{\phi}_1(\mathbf{x}; \boldsymbol{\theta}_1), \tag{2.15}$$

where $\boldsymbol{\phi}_i(\cdot; \boldsymbol{\theta}_i) : \mathbb{R}^{N_i - 1} \mapsto \mathbb{R}^{N_i}$, $i = 1, \ldots, L$, refers to the activation function applied at layer $i$ of the DFNN and $\boldsymbol{\theta}_i = (\mathbf{W}_i, \mathbf{b}_i)$, with $\mathbf{W}_i \in \mathbb{R}^{N_i \times N_{i-1}}$ and $\mathbf{b}_i \in \mathbb{R}^{N_i}$, $i = 1, \ldots, L$, are the weights and the bias of layer $i$ such that $\boldsymbol{\theta}_{DF} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L)$. We usually refer to the collection of all weights and biases as to the *parameters vector*. Each layer of the network corresponds to a matrix whose values are computed by applying a linear transformation to the previous layer followed by the application of a nonlinear activation function. In particular, referring to Figure 2.3, $\mathbf{y}_0 = \mathbf{x} \in \mathbb{R}^{N_0}$ is the input layer, $\mathbf{y}_L = \boldsymbol{\phi}^{DF}(\mathbf{x}; \boldsymbol{\theta}_{DF}) \in \mathbb{R}^{N_L}$ is the output layer, and each hidden layer $\mathbf{y}_i \in \mathbb{R}^{N_i}$, $i = 1, \ldots, L-1$, takes the form

$$\mathbf{y}_i = \boldsymbol{\phi}_i(\mathbf{W}_i \mathbf{y}_{i-1} + \mathbf{b}_i). \tag{2.16}$$

Given a set of $M$ input-output pair observations $\{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^M$ and considering a supervised learning paradigm [Goodfellow et al., 2016], the learning task consists in finding the optimal parameters vector $\boldsymbol{\theta}^*_{DF}$ by solving the optimization problem

$$\min_{\boldsymbol{\theta}_{DF}} \mathcal{J}(\boldsymbol{\theta}_{DF}) = \min_{\boldsymbol{\theta}_{DF}} \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}(\mathbf{y}^i, \mathbf{y}_L^i; \boldsymbol{\theta}_{DF}), \tag{2.17}$$

where $\mathcal{J}$ is the loss (or cost) function, and $\mathcal{L}$ is the per-example loss function, measuring the mismatch between the desired observed output $\mathbf{y}^i$ and the approximated one $\mathbf{y}_L^i$.

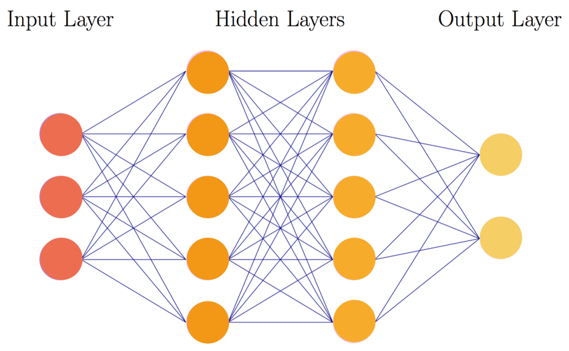


Figure 2.3: Feedforward neural network.

Problem (2.17) is usually solved by means of the gradient descent method exploiting the back-propagation algorithm [Rumelhart et al., 1986] to compute the derivatives of the loss function with respect to parameters. In particular, the gradient descent method requires to evaluate

$$\nabla_{\boldsymbol{\theta}_{DF}} \mathcal{J}(\boldsymbol{\theta}_{DF}) = \frac{1}{M} \sum_{i=1}^{M} \nabla_{\boldsymbol{\theta}_{DF}} \mathcal{L}(\mathbf{y}^i, \mathbf{y}_L^i; \boldsymbol{\theta}_{DF}), \tag{2.18}$$

a task which might easily become prohibitive when the size $M$ of the training dataset is very large, thus causing a single step of the gradient descent method to require a huge amount of time. The stochastic gradient descent (SGD) method allows to reduce the computational cost associated to the computation of the gradient of the loss function, by exploiting the fact that (2.18) can be considered as an expectation over the entire training dataset. Such an expectation can be approximated using a small set (or *minibatch*) of samples; hence, at each iteration the SGD method samples a minibatch of $m < M$ data points, drawn (e.g., uniformly) from the training dataset [Goodfellow et al., 2016], and approximates the gradient (2.18) of the loss function by

$$\widehat{\nabla}_{\boldsymbol{\theta}_{DF}} \mathcal{J}(\boldsymbol{\theta}_{DF}) = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}_{DF}} \mathcal{L}(\mathbf{y}^i, \mathbf{y}_L^i; \boldsymbol{\theta}_{DF}).$$

### 2.5.2 Convolutional neural networks

Convolutional neural networks (CNNs) [LeCun et al., 1998] are the standard neural network architecture in computer vision tasks, since they are well-suited to high-dimensional and spatially distributed data like images. This is due to the local approach of convolutional layers which enables them to exploit spatial correlations among pixels in order to extract low-level features of the input to carry out the task. The main ingredients of a convolutional layer are convolutional kernels, or filters, which consist in tensors of smaller dimensions with respect to the input. Each element of a feature map is obtained by sliding the kernel over the image and by computing the discrete convolution, as shown in Figure 2.4, where we assume striding equal to 1 and valid padding.

Considering a 3-dimensional input $\mathbf{Y}_0 = \mathbf{X} \in \mathbb{R}^{N_0^1 \times N_0^2 \times N_0^3}$ and a bank of $K_i$ convolutional filters in layer $i$ denoted as $\mathbf{W}_i^k \in \mathbb{R}^{n_i^1 \times n_i^2 \times n_i^3}$, $i = 1, \dots, L$ and $k = 1, \dots, K_i$, the $k$-th feature map is computed as

$$\mathbf{Y}_i^k = \phi_i(\mathbf{W}_i^k * \mathbf{Y}_{i-1} + b_i^k).$$

where $\mathbf{Y}_i \in \mathbb{R}^{N_i^1 \times N_i^2 \times N_i^3}$ (or, equivalently, $\mathbf{Y}_i^k \in \mathbb{R}^{N_i^1 \times N_i^2}$) with $N_i^1$ and $N_i^2$ depending on $n_i^1$ and $n_i^2$, respectively, the padding and the striding strategies, and $N_i^3 = K_i$.

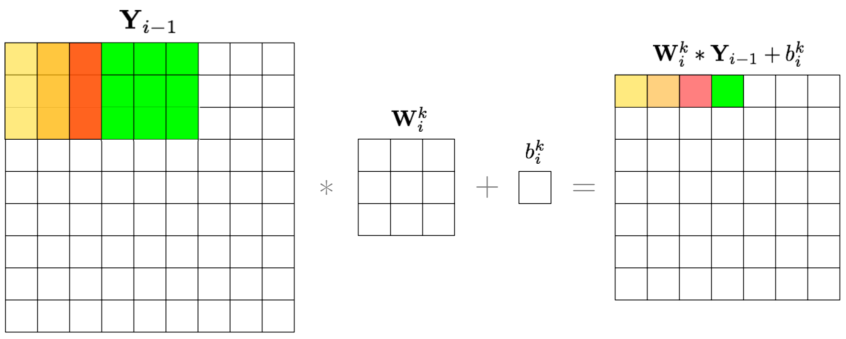


Figure 2.4: Computation of the elements of a feature map in a convolutional layer.

Convolutional layers are characterized by *shared parameters*, that is, weights are shared by all the elements (neurons) in a particular feature map, and *local connectivity*, that is, each neuron in a feature map is connected only to a local region of the input. Parameter sharing allows convolutional layers to enjoy another property: translation invariance or, more precisely, translation equivariance. This means that if the input varies, the output changes accordingly [Goodfellow et al., 2016]. In particular, if we apply a transformation to the input $Y_0$ and then compute the convolution, the result is the same we would obtain by computing the convolution and then applying the transformation to the output. The two properties above increase efficiency of CNNs, both in terms of memory and computational costs, with respect to DFNNs, thus making them preferable to the latter when dealing with extremely high-dimensional data.

### 2.5.3 Autoencoder neural networks

Autoencoders (AEs) [Bourlard and Kamp, 1998, Hinton and Zemel, 1994] are a particular type of feedforward neural networks aiming at reproducing, under suitable constraints, the identity function by generating the following map

$$\mathbf{f}^{AE}(\cdot;\boldsymbol{\theta}_E,\boldsymbol{\theta}_D) : \mathbf{x}_h \mapsto \tilde{\mathbf{x}}_h \quad \text{with} \quad \tilde{\mathbf{x}}_h \simeq \mathbf{x}_h, \tag{2.19}$$

where, of course, $\tilde{\mathbf{x}}_h$ is just an approximation of $\mathbf{x}_h$ due to the unavoidable error associated to the learning process. Internally, an AE has a hidden layer consisting in a *code* used to represent the input. We focus on undercomplete AEs [Goodfellow et al., 2016], where the constraint imposed is the reduction of the dimension of the code with respect to the input and output dimensions.

By considering the input $\mathbf{y}_0 = \mathbf{x}_h \in \mathbb{R}^{N_h}$ and the output $\mathbf{y}_L = \tilde{\mathbf{x}}_h \in \mathbb{R}^{N_h}$, an AE is composed by two main parts (see Figure 2.5)

- the *encoder* function $\mathbf{f}_n^E(\cdot;\boldsymbol{\theta}_E) : \mathbf{x}_h \mapsto \tilde{\mathbf{x}}_n = \mathbf{f}_n^E(\mathbf{x}_h;\boldsymbol{\theta}_E)$, where $\mathbf{f}_n^E(\cdot;\boldsymbol{\theta}_E) : \mathbb{R}^{N_h} \to \mathbb{R}^n$ and $n \ll N_h$, mapping the high-dimensional input $\mathbf{x}_h$ onto the low-dimensional code $\tilde{\mathbf{x}}_n$. The encoder function depends on a vector of parameters $\boldsymbol{\theta}_E \in \mathbb{R}^{N_E}$ collecting all the weights and biases specifying the function itself;

- the *decoder* function $\mathbf{f}_h^D(\cdot;\boldsymbol{\theta}_D) : \tilde{\mathbf{x}}_n \mapsto \tilde{\mathbf{x}}_h = \mathbf{f}_h^D(\tilde{\mathbf{x}}_n;\boldsymbol{\theta}_D)$, where $\mathbf{f}_h^D(\cdot;\boldsymbol{\theta}_D) : \mathbb{R}^n \to \mathbb{R}^{N_h}$, mapping the code $\tilde{\mathbf{x}}_n$ to an approximation of the original high-dimensional input $\tilde{\mathbf{x}}_h$. Similarly to the encoder function, the decoder function depends on a vector of parameters $\boldsymbol{\theta}_D \in \mathbb{R}^{N_D}$ collecting all the weights and biases specifying the function itself.

The AE is then defined as

$$\mathbf{f}^{AE}(\cdot;\boldsymbol{\theta}_E,\boldsymbol{\theta}_D) : \mathbf{x}_h \mapsto \tilde{\mathbf{x}}_h = \mathbf{f}_h^D(\mathbf{f}_n^E(\mathbf{x}_h;\boldsymbol{\theta}_E);\boldsymbol{\theta}_D).$$

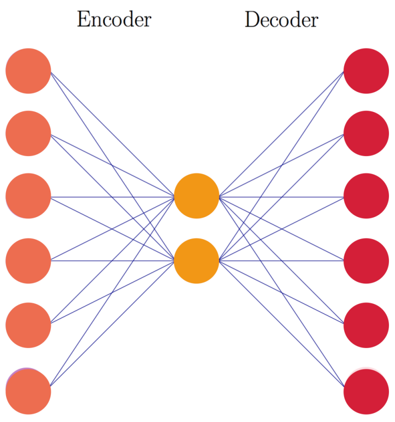


Figure 2.5: Autoencoder neural network.

AE learning lays within the *unsupervised learning* paradigm [Goodfellow et al., 2016] since its goal is to reconstruct the input being the target output an approximation of the input. An AE not only learns a low-dimensional representation of the high-dimensional input but also learns how to reconstruct the input from the code through the encoder and the decoder functions.

When dealing with large inputs, as the ones arising from the discretization of system (2.1), the use of a feedforward AE may become prohibitive as the number of parameters (weights and biases) required may be very large. As pointed out in Subsection 2.5.2, parameter sharing and local connectivity allow to reduce the numbers of parameters of the network and the number of associated computations, both in the forward and in the backward pass, thus training and testing computational times, hence the idea of relying on convolutional AEs for the sake of building our deep learning-based ROM technique.

**Remark 1** *By choosing single-layer linear encoder and decoder functions, i.e. by setting the activation function in (2.16) equal to the identity function, that is*

$$\tilde{\mathbf{x}}_n = \mathbf{f}_n^E(\mathbf{x}_h; \boldsymbol{\theta}_E) = \mathbf{W}_E \mathbf{x}_h + \mathbf{b}_E \qquad \text{and} \qquad \tilde{\mathbf{x}}_h = \mathbf{f}_h^D(\tilde{\mathbf{x}}_n; \boldsymbol{\theta}_D) = \mathbf{W}_D \tilde{\mathbf{x}}_n + \mathbf{b}_D,$$

*where* $\mathbf{W}_E \in \mathbb{R}^{n \times N_h}$, $\mathbf{b}_E \in \mathbb{R}^n$, $\mathbf{W}_D \in \mathbb{R}^{N_h \times n}$, $\mathbf{b}_D \in \mathbb{R}^{N_h}$, *and by setting the per-example loss function in (2.17) equal to*

$$\mathcal{L}(\mathbf{x}_h, \tilde{\mathbf{x}}_h; \boldsymbol{\theta}_E, \boldsymbol{\theta}_D) = ||\mathbf{x}_h - \tilde{\mathbf{x}}_h||^2 = ||\mathbf{x}_h - \mathbf{W}\mathbf{W}^T \mathbf{x}_h||^2, \tag{2.20}$$

*with* $\mathbf{W} = \mathbf{W}_D = \mathbf{W}_E^T$, *the AE will learn the same subspace as the one spanned by the first* $n$ *POD modes. However, without additional constraints on* $\mathbf{W}$, *the columns of* $\mathbf{W}$ *do not form an orthonormal basis or do not have any hierarchical ordering.*

## 2.6 Example: classification of atrial fibrillation by CNN

In the cardiac EP context, ML and DL are currently areas of intense exploration [Itchhaporia et al., 1996]. The use of this kind of algorithms, in cardiac EP research, has expanded exponentially in recent years, with emphasis on disease detection and diagnosis, prediction of patient outcomes, automatic segmentation of medical images and novel characterization of disease (see, e.g, [Feeny et al., 2020] for a complete review). Disease detection, starting from electrocardiograms (ECGs) or electrograms (EGMs) time series, results to be one of the areas in which the use of neural networks is more advanced [Rajpurkar et al., 2017, Xia et al., 2018].

In this Section we provide an example of application of neural networks to a classification problem, in a supervised learning paradigm (see Section 2.5), to detect AF from ECGs. This example is inspired by the PhysioNet Computing in Cardiology Challenge 2017 [Clifford et al., 2017].

The task we want to perform is classification (see Section 2.5 for further details) of short single lead ECG recordings in four classes: normal sinus rhythm, AF, alternative rhythm or noise (see Figure 2.6 (left)). Despite the great interest shown by the scientific community in detecting heart arrhythmias [Rajpurkar et al., 2017, Xia et al., 2018], AF detection remains problematic primarily because it may be episodic, as in paroxysmal cases (see Subsection 1.1.1). Previous studies concerning AF classification are generally limited in applicability because: *(i)* only classification of normal and AF rhythms are performed; *(ii)* good performance is shown on carefully selected clean data; *(iii)* a testing dataset is not used; *(iv)* only a small number of patients is considered. Reliable detection of AF, from a single short lead ECG, is a challenging task, and the broad taxonomy of rhythms makes this particularly difficult. In particular, many non AF rhythms exhibit irregular features that may be similar to AF.

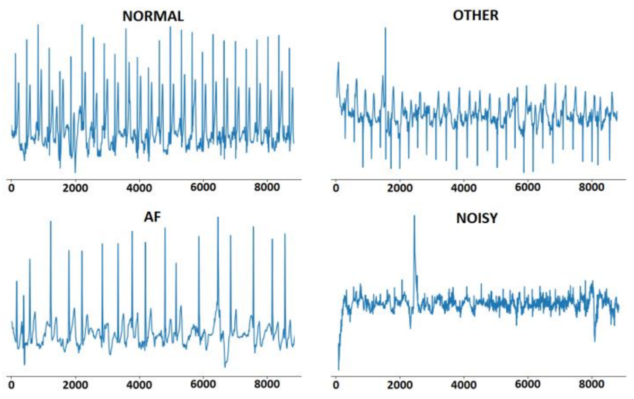
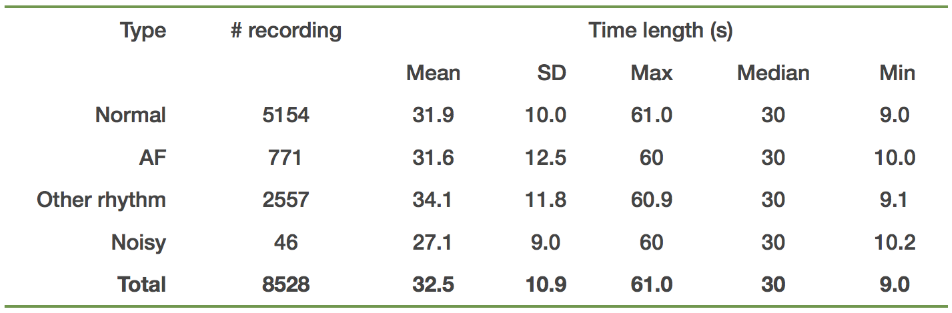


Figure 2.6: Left: Typical recordings for each of the four classes in the dataset [Clifford et al., 2017]. Right: AliveCor device [Clifford et al., 2017].

In the task considered, ECG recordings, collected using the AliveCor device (see Figure 2.6 (right)), are sampled at 300 Hz and band-pass filtered by the device. The training dataset is composed by 8528 signals of variable lengths and they are not uniformly distributed among the four classes, as shown in Figure 2.7. Indeed, the normal class represents almost the 60% of the entire dataset whereas the noise class only the 3%. Due to the very low class representation and to the fact that, from an electrophysiological point of view, the noise class is not meaningful we decide not to consider this class. Moreover, motivated by the fact that the testing dataset is still not available we perform only validation, not testing. Although we considered to reserve a small set of data at our disposal for the final testing purpose, we believe that is better to provide by our side the best DL model we can, using all the data for training and validation purposes. Two metrics are used to evaluate the performance of the neural network

- accuracy:
$$A = \frac{\#\text{correct predictions}}{\#\text{total predictions}},$$
- F1 score:
$$\bar{F_1} = \frac{F_{1n} + F_{1a} + F_{1o}}{3},$$



| Type | # recording | Time length (s) | | | | |
|---|---|---|---|---|---|---|
| | | Mean | SD | Max | Median | Min |
| Normal | 5154 | 31.9 | 10.0 | 61.0 | 30 | 9.0 |
| AF | 771 | 31.6 | 12.5 | 60 | 30 | 10.0 |
| Other rhythm | 2557 | 34.1 | 11.8 | 60.9 | 30 | 9.1 |
| Noisy | 46 | 27.1 | 9.0 | 60 | 30 | 10.2 |
| Total | 8528 | 32.5 | 10.9 | 61.0 | 30 | 9.0 |

Figure 2.7: Data profile for the training set [Clifford et al., 2017].

where $F_{1n} = \dfrac{2Nn}{\sum N + \sum n}$, $F_{1a} = \dfrac{2Aa}{\sum A + \sum a}$ and $F_{1o} = \dfrac{2Oo}{\sum O + \sum o}$ in accord with Figure 2.8. This is the scoring method provided by the organizers of the Physionet Challenge 2017.

| | Predicted Classification | | | | |
|---|---|---|---|---|---|
| | Normal | AF | Other | Noisy | Total |
| Normal | $Nn$ | $Na$ | $No$ | $Np$ | $\sum N$ |
| AF | $An$ | $Aa$ | $Ao$ | $Ap$ | $\sum A$ |
| Other | $On$ | $Oa$ | $Oo$ | $Op$ | $\sum O$ |
| Noisy | $Pn$ | $Pa$ | $Po$ | $Pp$ | $\sum P$ |
| Total | $\sum n$ | $\sum a$ | $\sum o$ | $\sum p$ | |

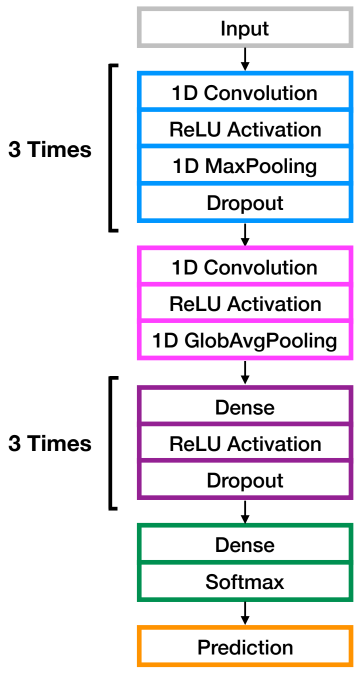
Figure 2.8: Scoring notation used for the confusion matrix [Clifford et al., 2017].

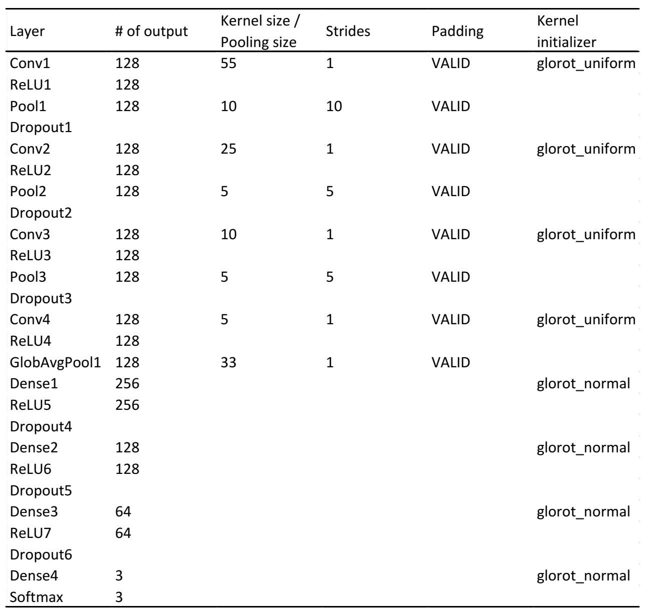
Data are splitted cyclically into training and validation sets, according to the proportion of 9 to 1, with the 10-fold cross-validation approach [Goodfellow et al., 2016], used to reduce the variance of the generalization error. We end up with 10 different networks which global performance are evaluated considering the average of the performance of the 10 networks. Moreover, data are normalized by using the mean and the standard deviation of all training samples. Each sample is truncated to a length of reference equal to 10100 elements. When the time series is shorter the signal is extended by repeating the last elements.

We decided to implement a 1D deep CNN, taking as a baseline one of the highest ranked projects in the challenge [Pyakillya et al., 2017, Xiong et al., 2017]. Moreover, a 34-layer 1D CNN has shown to exceed the average cardiologists performance in classifying ECGs among 14 classes [Rajpurkar et al., 2017]. The architecture of the neural network used to solve the classification task is shown in Figure 2.9. The neural network presents a structure of convolutional and dense blocks, followed by a final dense layer. We rely on the dropout technique [Srivastava et al., 2014], which limits overfitting by randomly setting outputs activation to 0 during training, according to the dropout rate $d_p$. The weights and biases initializations follow the Glorot and Bengio uniform law, in the case of convolutional layers, and the Glorot and Bengio normal law, in the case of dense layers [Glorot and Bengio, 2010]. As nonlinear activation function we employ the ReLU function [Agarap, 2018] defined as

$$\sigma(z) = \begin{cases} z & z \geq 0 \\ 0 & z < 0. \end{cases}$$

We solve the optimization problem (2.17) by means of the ADAM algorithm [Kingma and Ba, 2015] with a starting learning rate $\eta = 10^{-3}$. The bacth size, dropout rate and maximum





| Layer | # of output | Kernel size / Pooling size | Strides | Padding | Kernel initializer |
|---|---|---|---|---|---|
| Conv1 | 128 | 55 | 1 | VALID | glorot_uniform |
| ReLU1 | 128 | | | | |
| Pool1 | 128 | 10 | 10 | VALID | |
| Dropout1 | | | | | |
| Conv2 | 128 | 25 | 1 | VALID | glorot_uniform |
| ReLU2 | 128 | | | | |
| Pool2 | 128 | 5 | 5 | VALID | |
| Dropout2 | | | | | |
| Conv3 | 128 | 10 | 1 | VALID | glorot_uniform |
| ReLU3 | 128 | | | | |
| Pool3 | 128 | 5 | 5 | VALID | |
| Dropout3 | | | | | |
| Conv4 | 128 | 5 | 1 | VALID | glorot_uniform |
| ReLU4 | 128 | | | | |
| GlobAvgPool1 | 128 | 33 | 1 | VALID | |
| Dense1 | 256 | | | | glorot_normal |
| ReLU5 | 256 | | | | |
| Dropout4 | | | | | |
| Dense2 | 128 | | | | glorot_normal |
| ReLU6 | 128 | | | | |
| Dropout5 | | | | | |
| Dense3 | 64 | | | | glorot_normal |
| ReLU7 | 64 | | | | |
| Dropout6 | | | | | |
| Dense4 | 3 | | | | glorot_normal |
| Softmax | 3 | | | | |

Figure 2.9: Left: CNN architecture. Right: Detailed architecture.

number of epochs are set to $N_b = 276$, $d_p = 0.5$ and $N_{epochs} = 50$, respectively, through hyperparameters tuning. `Tensorflow` is used as DL framework for the implementation of the model [Abadi et al., 2016] and all the computations are performed on a Nvidia GeForce GTX 1070 8 GB GPU.

In Figure 2.10 we show the trend of the accuracy over the validation set versus the epochs for each fold. Both the accuracy $A$ and the F1 score $\bar{F}_1$ are equal to 0.85 (see Table 2.6). The class predicted with lower accuracy is the O, i.e. alternative rhythm, one due to the fact that this class collects a great variety of signals, related to different pathologies, whereas the other classes refer to one precise condition. This is confirmed by the scores obtained by participants to the challenge. Indeed, in Table 2.6 we compare the performance of our model with the one obtained by [Zihlmann et al., 2017, Xiong et al., 2017, Andreotti et al., 2017]. In particular, [Xiong et al., 2017] employs a 1D CNN and has gained the second highest score, in [Zihlmann et al., 2017] the authors implement a 2D CNN which takes as input the spectrograms of the ECGs and [Andreotti et al., 2017] uses a 1D residual CNN inspired by the work of [Rajpurkar et al., 2017]. In Figure 2.11 (left) the mean confusion matrix

| | AF | N | O | $\bar{F}_1$ |
|---|---|---|---|---|
| our model | 0.86 | 0.9 | 0.78 | 0.85 |
| Andreotti et al. | 0.68 | 0.88 | 0.67 | 0.74 |
| Zihlmann et al. | 0.77 | 0.89 | 0.73 | 0.8 |
| Xiong et al. | 0.87 | 0.93 | 0.83 | 0.88 |

Table 2.1: Comparison among our network and existing models.

for the proposed model is shown. By looking at the third line, the loss of accuracy in the prediction of the class O, with respect to the others, is related to the records misclassified normal. Finally, in Figure 2.11 (right), we show the accuracy trend over the validation set versus the number of epochs for the best network of the 10-fold cross-validation by achieving

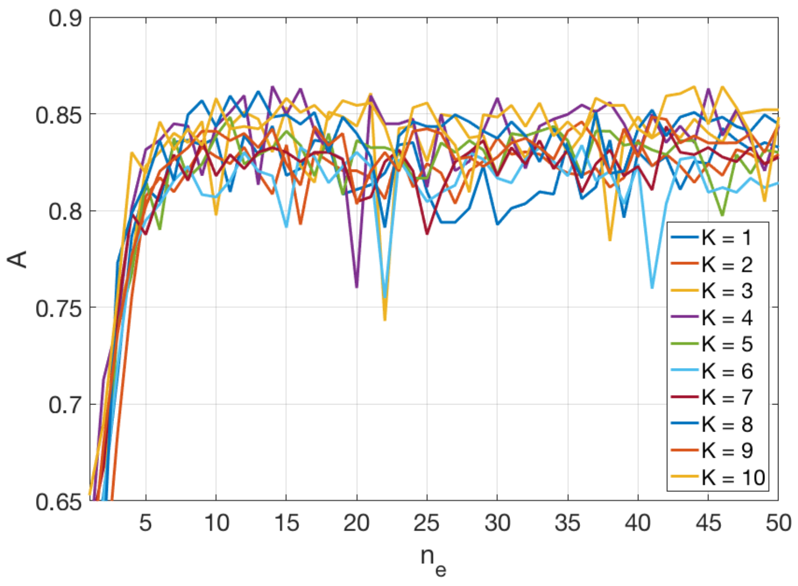
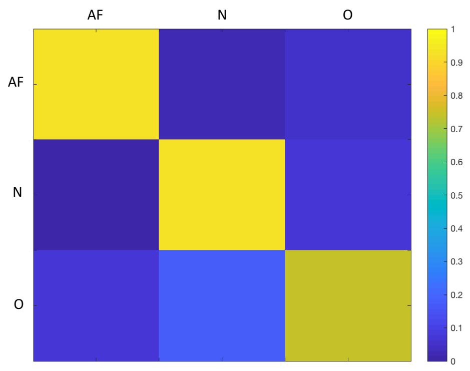


Figure 2.10: Accuracy trend vs. number of epochs with 10-fold cross-validation.

an accuracy equal to 0.87, higher than the 0.85 one obtained in [Pyakillya et al., 2017].



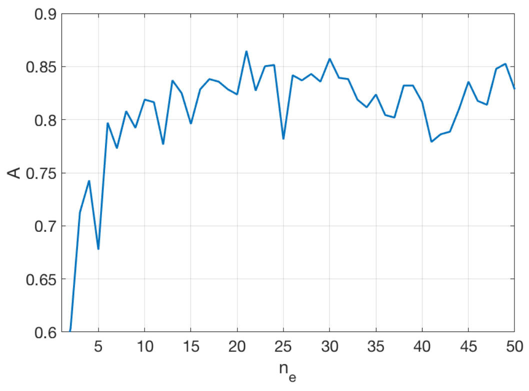


Figure 2.11: Left: Confusion matrix for 10-fold cross-validation. Right: Accuracy trend vs. number of epochs for the best network.

In conclusion, we are able to accurately identify AF short single-lead ECGs by means of the model presented in this Section. The proposed model gives comparable results with the performance obtained by participants to the challenge. Our findings confirm that an approach based on deep neural networks is suitable for this kind of problems.

Neural networks, applied to the kind of task described in this Section, are not able to provide information about the spatial distribution of the electrical signal in the heart. We are interested in using DL algorithms, in the cardiac EP applicative context, over their classification ability, that is we focus on the efficient reconstruction of the potential field, which otherwise would be the output of the expensive solution of the Bidomain and Monodomain equations by suitable numerical methods, such as the FE method and IGA, in order to reproduce all the time and spatial scales of the transmembrane potential. Indeed, we want to approximate the map $(t, \boldsymbol{\mu}) \mapsto \mathbf{u}_h(t, \boldsymbol{\mu})$, where $t \in (0, T)$ denotes time, $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{n_\mu}$ a vector of input parameters and $\mathbf{u}_h(t, \boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ the solution of cardiac EP systems, in order to retain more information than the one associated to an ECG, as in this case. In particular, by accurately characterizing the solutions of equations (1.1) and (1.3) in terms of their time and spatial scales, inferring a specific arrhythmia is a straightforward task.

Chapter 3

# A deep learning-based reduced order model for time-dependent nonlinear parametrized PDEs

In this Chapter[1], we propose a computational, non-intrusive approach based on deep learning (DL) algorithms to deal with the construction of efficient reduced order models (ROMs) (which we refer to as DL-ROMs) in order to tackle parameter-dependent PDEs; in particular, we consider PDEs that feature wave-type phenomena. A comprehensive framework is presented for the global approximation of the map $(t, \boldsymbol{\mu}) \mapsto \mathbf{u}_h(t, \boldsymbol{\mu})$, where $t \in (0, T)$ denotes time, $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{n_\mu}$ a vector of input parameters and $\mathbf{u}_h(t, \boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ the solution of a dynamical system arising from the space discretization of a time-dependent (non)linear parametrized PDE. Moreover, we assess the DL-ROM numerical accuracy on three different test cases of increasing complexity (with respect to the parametric dependence and the nature of the PDE). We show that the proposed DL-ROM framework can efficiently provide solutions to parametrized cardiac electrophysiology (EP) problems in Chapter 4.

## 3.1 Parametrized PDEs and deep learning

As outlined in Chapter 2, conventional ROMs, such as POD-Galerkin ROMs, show severe limitations when dealing with nonlinear time-dependent parametrized PDEs, such as cardiac EP problems. These might be related to *(i)* the need to deal with projections onto high dimensional linear approximating trial manifolds, *(ii)* expensive hyper-reduction strategies, or *(iii)* the intrinsic difficulty to handle physical complexity with linear superimpositions of modes. Several recent works have shown possible applications of DL techniques to parametrized PDEs – thanks to their approximation capabilities, their extremely favorable computational performances during online testing phases, and their relative easiness of implementation – both from a theoretical [Kutyniok et al., 2019] and a computational standpoint. Regarding this latter aspect, artificial neural networks (ANN), such as feedforward neural networks, have been employed to model the reduced dynamics in a data-driven and less intrusive way (avoiding, e.g., the costs entailed by projection-based ROMs), but still relying on a linear trial manifold built, e.g., through POD. For instance, in [Guo and Hesthaven, 2018, Guo and Hesthaven, 2019, Hesthaven and Ubbiali, 2018, San and Maulik, 2018, Kast et al., 2020] the solution of a (nonlinear, time-dependent) ROM for any new parameter value has been

[1]This Chapter is mainly based on the paper [Fresca et al., 2020a].

replaced by the evaluation of ANN-based regression models; similar ideas can be found, e.g., in [Kani and Elsheikh, 2017, Mohan and Gaitonde, 2018, Wan et al., 2018, Pulch and Youssef, 2020, Bērzinš et al., 2020].

A first effort to include physics laws in the definition of the ROM is provided in [Chen et al., 2020] where the intrinsic coordinates of the linear trial manifold, generated by POD, are approximated by means of a feedforward neural network trained by minimizing the mean squared residual error of the ROM on a set of points in the parameter space. The limitations of this approach are related to the fact that it has been applied only to steady state PDEs and it is not capable of handling efficiently terms that depend nonlinearly on either the solution or the input parameters of the problem. Few attempts have been made in order to describe the reduced trial manifold where the approximation is sought (avoiding, e.g., the linear superimposition of POD modes) through ANNs, see, e.g., [González and Balajewicz, 2018, Lee and Carlberg, 2020]. In particular:

- a projection-based ROM technique has been introduced in [Lee and Carlberg, 2020], in which the FOM system is projected onto a nonlinear trial manifold identified by means of the decoder function of a convolutional AE. However, the ROM is derived by minimizing a residual formulation, for which the quasi-Newton method herein employed requires the computation of an approximated Jacobian of the residual at each time step,
- a ROM technique based on a deep convolutional recurrent AE has been proposed in [González and Balajewicz, 2018], where a reduced trial manifold is generated through a convolutional AE; the latter is then used to train a Long Short-Term Memory (LSTM) neural network modeling the reduced dynamics. However, explicit parameter dependence in the PDE problem is not considered, apart from $\boldsymbol{\mu}$-dependent initial data, and the LSTM is trained on reduced approximations obtained through the encoder function of the AE.

Another promising application of machine learning techniques within a ROM framework deals with the efficient evaluation of ROM errors, see, e.g., [Freno and Carlberg, 2018, Pagani et al., 2019, Parish and Carlberg, 2019, Trehan et al., 2017].

We set up nonlinear ROMs whose dimension is nearly equal (if not equal) to the intrinsic dimension of the solution manifold that we aim at approximating (see Section 2.2). Our DL-ROM approach combines and improves the techniques introduced in [González and Balajewicz, 2018, Lee and Carlberg, 2020] by shaping an all-inclusive DL-based ROM technique, where we both

- construct the reduced trial manifold,
- model the reduced dynamics on it employing ANNs.

The former task is achieved by using the decoder function of a convolutional AE; the latter task is instead carried out by considering a feedforward neural network and the encoder function of a convolutional AE. Moreover, we set up a computational procedure performing the training of both network architectures simultaneously, by minimizing a loss function that weights two terms, one dedicated to each single task.

In this respect, we are able to design a flexible framework capable to handle parameters affecting both PDE operators and data, which avoids both the expensive projection stage of [Lee and Carlberg, 2020] and the training of a more expensive LSTM network. In the DL-ROM, the intrusive construction of a ROM is replaced by the evaluation of the ROM generalized coordinates through a deep feedforward neural network taking only $(t, \boldsymbol{\mu})$ as inputs. The proposed technique is purely data-driven, non-intrusive, that is, it only relies on the computation of a set of FOM snapshots – in this respect, DL does not replace the high-fidelity FOM as, e.g., in the works by Karniadakis and coauthors [Raissi and Karniadakis, 2018, Raissi et al., 2017a, Raissi et al., 2017b, Raissi et al., 2019, Raissi, 2018]; rather, DL techniques are built upon it, to enhance the repeated evaluation of the FOM for different values of the parameters.

## 3.2 Deep learning-based reduced order models (DL-ROMs)

In this Section we detail the construction of the proposed nonlinear ROM. In this respect, we define the functions $\boldsymbol{\Psi}_h$ and $\boldsymbol{\Phi}_n$ in (2.12) and (2.14) by means of DL algorithms, exploiting neural network architectures. This choice is motivated by their ability of effectively approximating nonlinear maps, and by their ability to learn from data and generalize to unseen data. On the other hand, DL models enable us to build non-intrusive, completely data-driven, ROMs, since their construction only requires to access the dataset, the parameter values and the snapshot matrix, but not the FOM arrays appearing in (2.1). The DL-ROM technique that we develop is composed by two main blocks responsible, respectively, for the *reduced dynamics learning* and the *reduced trial manifold learning* (see Figure 3.1). Hereon, we denote by $N_{train}$, $N_{test}$ and $N_t$ the number of training-parameter instances, of testing-parameter instances and time instances, respectively, and we set $N_s = N_{train}N_t$. The dimension of both the FOM solution and the ROM approximation is $N_h$, while $n \ll N_h$ denotes the number of intrinsic coordinates.

For the description of the system dynamics on the reduced nonlinear trial manifold (which we refer to as *reduced dynamics learning*), we employ a *deep feedforward neural network* (DFNN) with $L$ layers, that is, we define the function $\boldsymbol{\Phi}_n$ in definition (2.14) as

$$\boldsymbol{\Phi}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF}) = \boldsymbol{\phi}_n^{DF}(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF}), \tag{3.1}$$

thus yielding the map

$$(t,\boldsymbol{\mu}) \mapsto \mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF}) = \boldsymbol{\phi}_n^{DF}(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF}),$$

where $\boldsymbol{\phi}_n^{DF}$ takes the form (2.15), with $t \in [0,T)$, and results from the subsequent composition of a nonlinear activation function, with a linear transformation of the input, $L$ times. Here $\boldsymbol{\theta}_{DF}$ denotes the vector of parameters of the DFNN.

Regarding instead the description of the reduced nonlinear trial manifold $\tilde{\mathcal{S}}_n$ defined in (2.13) (which we refer to as *reduced trial manifold learning*), we employ the *decoder function of a convolutional autoencoder* (AE), that is, we define the function $\boldsymbol{\Psi}_h$ appearing in (2.12) and (2.13) as

$$\boldsymbol{\Psi}_h(\mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF});\boldsymbol{\theta}_D) = \mathbf{f}_h^D(\mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF});\boldsymbol{\theta}_D), \tag{3.2}$$

thus yielding the map

$$\mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF}) \mapsto \tilde{\mathbf{u}}_h(t;\boldsymbol{\mu},\boldsymbol{\theta}) = \mathbf{f}_h^D(\mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF});\boldsymbol{\theta}_D),$$

where $\mathbf{f}_h^D$ results from the composition of several layers, some of which of convolutional type, overall depending on the vector $\boldsymbol{\theta}_D$ of parameters of the decoder function.

Combining the two former stages, the DL-ROM approximation is given by

$$\tilde{\mathbf{u}}_h(t;\boldsymbol{\mu},\boldsymbol{\theta}) = \mathbf{f}_h^D(\boldsymbol{\phi}_n^{DF}(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF});\boldsymbol{\theta}_D), \tag{3.3}$$

where $\boldsymbol{\phi}_n^{DF}(\cdot;\cdot,\boldsymbol{\theta}_{DF}) : [0,T) \times \mathbb{R}^{n_\mu} \to \mathbb{R}^n$ and $\mathbf{f}_h^D(\cdot;\boldsymbol{\theta}_D) : \mathbb{R}^n \to \mathbb{R}^{N_h}$ are defined as in (3.1) and (3.2), respectively, and $\boldsymbol{\theta} = (\boldsymbol{\theta}_{DF},\boldsymbol{\theta}_D)$ are the parameters defining the neural network. The architecture of DL-ROM is shown in Figure 3.1.

Computing the ROM approximation (3.3) for any new value of $\boldsymbol{\mu} \in \mathcal{P}$, at any given time, requires evaluation of the map $(t,\boldsymbol{\mu}) \to \tilde{\mathbf{u}}_h(t;\boldsymbol{\mu},\boldsymbol{\theta})$ at the testing stage, once the parameters

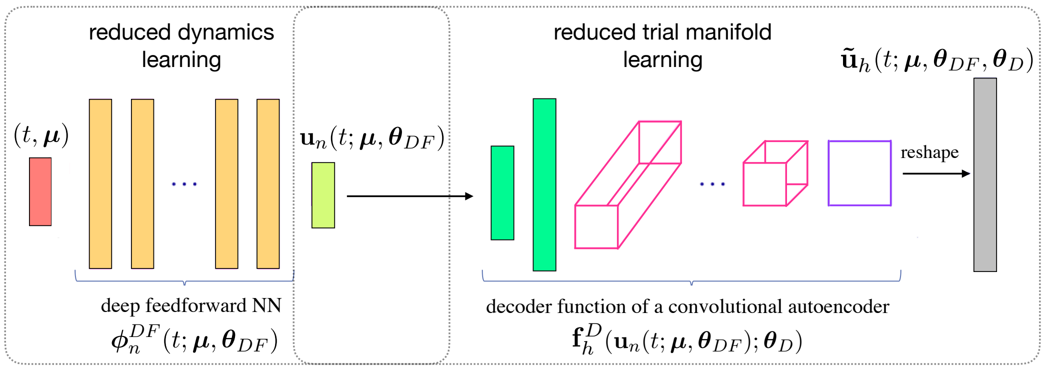

reduced dynamics learning
$(t, \boldsymbol{\mu})$
deep feedforward NN
$\boldsymbol{\phi}_n^{DF}(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF})$
$\mathbf{u}_n(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF})$
reduced trial manifold learning
$\tilde{\mathbf{u}}_h(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF}, \boldsymbol{\theta}_D)$
reshape
decoder function of a convolutional autoencoder
$\mathbf{f}_h^D(\mathbf{u}_n(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF}); \boldsymbol{\theta}_D)$

Figure 3.1: DL-ROM architecture (online stage, testing): the DL-ROM to be queried for any new selected couple $(t, \boldsymbol{\mu})$ during the testing phase.

$\boldsymbol{\theta} = (\boldsymbol{\theta}_{DF}, \boldsymbol{\theta}_D)$ have been determined, once and for all, during the training stage. The training stage consists in solving an optimization problem (in the variable $\boldsymbol{\theta}$) after a set of snapshots of the FOM have been computed. More precisely, provided the parameter matrix $\mathbf{M} \in \mathbb{R}^{(n_{\boldsymbol{\mu}}+1)\times N_s}$ defined as

$$\mathbf{M} = [(t^1, \boldsymbol{\mu}_1)|\dots|(t^{N_t}, \boldsymbol{\mu}_1)|\dots|(t^1, \boldsymbol{\mu}_{N_{train}})|\dots|(t^{N_t}, \boldsymbol{\mu}_{N_{train}})], \tag{3.4}$$

and the snapshot matrix $\mathbf{S}$, we find the optimal parameters $\boldsymbol{\theta}^*$ solution of

$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{N_s} \sum_{i=1}^{N_{train}} \sum_{k=1}^{N_t} \mathcal{L}(t^k, \boldsymbol{\mu}_i; \boldsymbol{\theta}) \to \min_{\boldsymbol{\theta}} \tag{3.5}$$

where

$$\begin{aligned} \mathcal{L}(t^k, \boldsymbol{\mu}_i; \boldsymbol{\theta}) &= \frac{1}{2}\|\mathbf{u}_h(t^k; \boldsymbol{\mu}_i) - \tilde{\mathbf{u}}_h(t^k; \boldsymbol{\mu}_i, \boldsymbol{\theta})\|^2 \\ &= \frac{1}{2}\|\mathbf{u}_h(t^k; \boldsymbol{\mu}_i) - \mathbf{f}_h^D(\boldsymbol{\phi}_n^{DF}(t^k; \boldsymbol{\mu}_i, \boldsymbol{\theta}_{DF}); \boldsymbol{\theta}_D)\|^2. \end{aligned} \tag{3.6}$$

To solve the optimization problem (3.5)-(3.6) we use the ADAM algorithm [Kingma and Ba, 2015] which is a stochastic gradient descent method [Robbins and Monro, 1951] computing an adaptive approximation of the first and second momentum of the gradients of the loss function. In particular, it computes exponentially weighted moving averages of the gradients and of the squared gradients. We set the starting learning rate to $\eta = 10^{-4}$, the batch size to $N_b = 20$ and the maximum number of epochs to $N_{epochs} = 10000$. We perform cross-validation, in order to tune the hyperparameters of the DL-ROM, by splitting the data in training and validation sets, with a proportion 8:2. Moreover, we implement an early-stopping regularization technique to reduce overfitting [Goodfellow et al., 2016], arresting the training if the loss, over the validation set, does not decrease over 500 epochs. As nonlinear activation function we employ the ELU function [Clevert et al., 2015] defined as

$$\sigma(z) = \begin{cases} z & z \geq 0 \\ \exp(z) - 1 & z < 0. \end{cases}$$

No activation function is applied at the last convolutional layer of the decoder neural network, as usually done when dealing with AEs. The parameters, weights and biases, are initialized through the He uniform initialization [He et al., 2015].

As we rely on a convolutional autoencoder to define the function $\boldsymbol{\Psi}_h$, we also exploit the encoder function

$$\tilde{\mathbf{u}}_n(t; \boldsymbol{\mu}, \boldsymbol{\theta}_E) = \mathbf{f}_n^E(\mathbf{u}_h(t; \boldsymbol{\mu}); \boldsymbol{\theta}_E), \tag{3.7}$$

which maps each FOM solution associated to $(t; \boldsymbol{\mu}) \in \text{Col}(\mathbf{M})$ provided as inputs to the feed-forward neural network (3.1), onto a low-dimensional representation $\tilde{\mathbf{u}}_n(t; \boldsymbol{\mu}, \boldsymbol{\theta}_E)$ depending on the parameters vector $\boldsymbol{\theta}_E$ defining the encoder function.

Indeed, the actual architecture of DL-ROM used only during the training and the validation phases, but not during testing, is the one shown in Figure 3.2. In practice, we add to

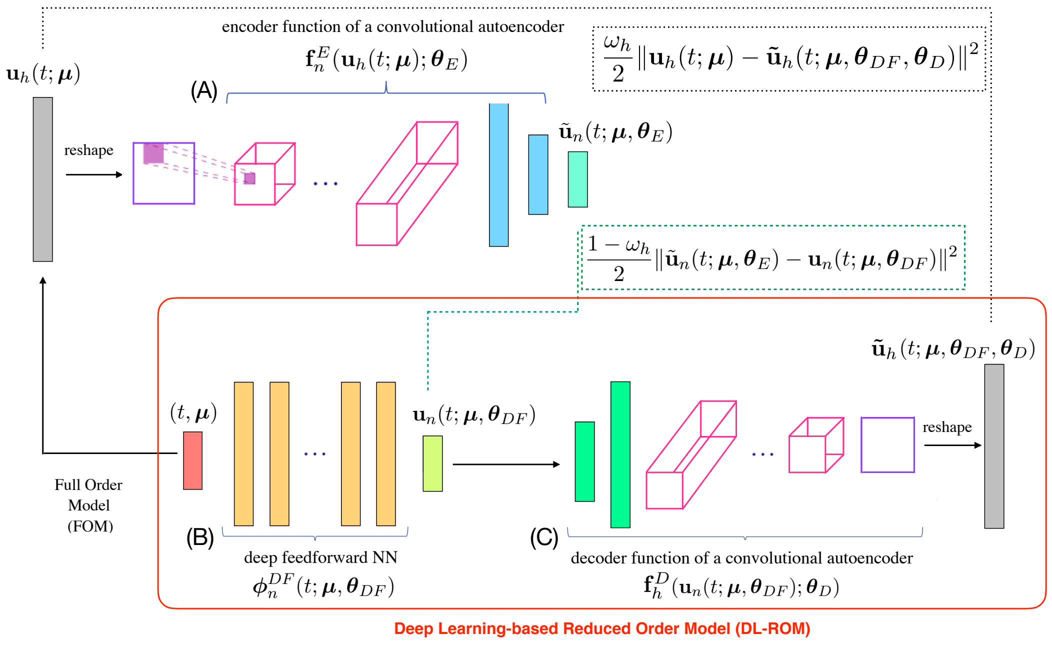

encoder function of a convolutional autoencoder
$\mathbf{f}_n^E(\mathbf{u}_h(t;\boldsymbol{\mu});\boldsymbol{\theta}_E)$
$\frac{\omega_h}{2}\|\mathbf{u}_h(t;\boldsymbol{\mu}) - \tilde{\mathbf{u}}_h(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF},\boldsymbol{\theta}_D)\|^2$
$\mathbf{u}_h(t;\boldsymbol{\mu})$
(A)
reshape
$\tilde{\mathbf{u}}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_E)$
$\frac{1-\omega_h}{2}\|\tilde{\mathbf{u}}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_E) - \mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF})\|^2$
$\tilde{\mathbf{u}}_h(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF},\boldsymbol{\theta}_D)$
$(t,\boldsymbol{\mu})$
$\mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF})$
reshape
Full Order Model (FOM)
(B)
(C)
deep feedforward NN
$\boldsymbol{\phi}_n^{DF}(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF})$
decoder function of a convolutional autoencoder
$\mathbf{f}_h^D(\mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF});\boldsymbol{\theta}_D)$
Deep Learning-based Reduced Order Model (DL-ROM)

Figure 3.2: DL-ROM architecture (offline stage, training and validation): DL-ROM architecture used during the training phase. The FOM solution $\mathbf{u}_h(t; \boldsymbol{\mu})$ is provided as input to block (A) which outputs $\tilde{\mathbf{u}}_n(t; \boldsymbol{\mu})$. The same parameter instance associated to the FOM, i.e. $(t; \boldsymbol{\mu})$, enters block (B) which provides as output $\mathbf{u}_n(t; \boldsymbol{\mu})$ and the error between the low-dimensional vectors (dashed green box) is accumulated. The intrinsic coordinates $\mathbf{u}_n(t; \boldsymbol{\mu})$ are given as input to block (C) returning the ROM approximation $\tilde{\mathbf{u}}_h(t; \boldsymbol{\mu})$. Then the reconstruction error (dashed black box) is computed.

the DL-ROM architecture introduced above the encoder function of the convolutional AE. This produces an additional term in the *per-example* loss function (3.6), thus yielding the following optimization problem to be solved:

$$\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{N_s} \sum_{i=1}^{N_{train}} \sum_{k=1}^{N_t} \mathcal{L}(t^k, \boldsymbol{\mu}_i; \boldsymbol{\theta}), \tag{3.8}$$

where

$$\begin{aligned} \mathcal{L}(t^k, \boldsymbol{\mu}_i; \boldsymbol{\theta}) = \; & \frac{\omega_h}{2}\|\mathbf{u}_h(t^k; \boldsymbol{\mu}_i) - \tilde{\mathbf{u}}_h(t^k; \boldsymbol{\mu}_i, \boldsymbol{\theta}_{DF}, \boldsymbol{\theta}_D)\|^2 \\ & + \frac{1-\omega_h}{2}\|\tilde{\mathbf{u}}_n(t^k; \boldsymbol{\mu}_i, \boldsymbol{\theta}_E) - \mathbf{u}_n(t^k; \boldsymbol{\mu}_i, \boldsymbol{\theta}_{DF})\|^2 \end{aligned} \tag{3.9}$$

and $\boldsymbol{\theta} = (\boldsymbol{\theta}_E, \boldsymbol{\theta}_{DF}, \boldsymbol{\theta}_D)$, with $\omega_h \in [0, 1]$. The *per-example* loss function (3.9) combines the reconstruction error (that is, the error between the FOM solution and the DL-ROM approximation) and the error between the intrinsic coordinates and the output of the encoder. This further term allows to enhance the performance of the DL-ROM, as shown in Test 3 of Section 3.4.

## 3.3 Training and testing algorithms

We now detail the algorithms through which the training and testing phases of the networks are performed. First of all, data normalization and standardization enhance the training phase of the network by rescaling all the values contained in the dataset to a common frame. For this reason, the inputs and the output of DL-ROM are normalized by applying an affine transformation in order to rescale them in the range $[0,1]$. In particular, provided the training parameter matrix $\mathbf{M}^{train} \in \mathbb{R}^{(n_{\boldsymbol{\mu}}+1)\times N_s}$, we define

$$M^i_{max} = \max_{j=1,\dots,N_s} M^{train}_{ij}, \qquad M^i_{min} = \min_{j=1,\dots,N_s} M^{train}_{ij}, \tag{3.10}$$

so that data are normalized by applying the following transformation

$$M^{train}_{ij} \mapsto \frac{M^{train}_{ij} - M^i_{max}}{M^i_{max} - M^i_{min}}, \qquad i = 1,\dots,n_{\boldsymbol{\mu}}+1,\ j = 1,\dots,N_s. \tag{3.11}$$

Each feature of the training parameter matrix is rescaled according to its maximum and minimum values. Regarding instead the training snapshot matrix $\mathbf{S}^{train} \in \mathbb{R}^{N_h\times N_s}$, we define

$$S_{max} = \max_{i=1,\dots,N_h}\max_{j=1,\dots,N_s} S^{train}_{ij}, \qquad S_{min} = \min_{i=1,\dots,N_h}\min_{j=1,\dots,N_s} S^{train}_{ij}, \tag{3.12}$$

and apply transformation (3.11) by replacing $M^i_{max}, M^i_{min}$ with $S_{max}, S_{min} \in \mathbb{R}$, respectively, that is. we use the same maximum and minimum values for all the features of the snapshot matrix, as in [Lee and Carlberg, 2020, González and Balajewicz, 2018]. Using the latter approach or employing each feature's maximum and minimum values, for the matrix $\mathbf{S}^{train}$, does not lead to remarkable changes in the DL-ROM performance. Transformation (3.11) is applied also to the validation and testing sets, but considering as as maximum and minimum the values computed over the training set. In order to rescale the reconstructed solution to the original values, we apply the inverse transformation of (3.11). We point out that the input of the encoder function, the FOM solution $\mathbf{u}_h = \mathbf{u}_h(t^k;\boldsymbol{\mu}_i)$ for a given (time, parameter) instance $(t^k,\boldsymbol{\mu}_i)$, is reshaped in a matrix. In particular, starting from $\mathbf{u}_h \in \mathbb{R}^{N_h}$ we apply the transformation $\mathbf{u}^R_h$=reshape($\mathbf{u}_h$) where $\mathbf{u}^R_h \in \mathbb{R}^{N_h^{1/2}\times N_h^{1/2}}$. If $N_h$ is not a square, the input $\mathbf{u}_h$ is zero-padded [Goodfellow et al., 2016]. For the sake of simplicity, we continue to refer to the reshaped FOM solution to as $\mathbf{u}_h$. The inverse reshaping transformation is applied to the output of the last convolutional layer in the decoder function, the ROM approximation. Moreover, we highlight that applying one of the functions (3.1)-(3.2)-(3.7) to a matrix $\mathbf{X} \in \mathbb{R}^{m\times N_s}$ means applying it column-wise. The reduced dimension is chosen through hyperparameters tuning, i.e. we start from $n$ equal to the dimension of the solution manifold $(n_\mu + 1)$ and select a different value for $n$ only if it leads to a significant increase of the performance of the neural network.

The training algorithm referring to the architecture of DL-ROM depicted in Figure 3.2 is reported in Algorithm 1. During the training phase, the optimal parameters of the DL-ROM neural network are found by solving the optimization problem (3.8)-(3.9) through the back-propagation and ADAM algorithms. At testing time, the encoder function is instead discarded (the DL-ROM architecture is the one shown in Figure 3.1) and the testing algorithm is provided by Algorithm 2. The testing phase corresponds to a forward step of the DL-ROM neural network in Figure 3.1.

We implement the DL-ROM neural network by means of the `Tensorflow` DL framework [Abadi et al., 2016]; numerical simulations are performed on a workstation equipped with an Nvidia GeForce GTX 1070 8 GB GPU. The code developed for the following tests (Section 3.4) is freely available at `https://github.com/stefaniafresca/DL-ROM-Meth`.

**Algorithm 1** DL-ROM training

```
Input: Parameter matrix $\mathbf{M} \in \mathbb{R}^{(n_{\mu}+1)\times N_s}$, snapshot matrix $\mathbf{S} \in \mathbb{R}^{N_h \times N_s}$, training-validation splitting fraction $\alpha$, starting learning rate $\eta$, batch size $N_b$, maximum number of epochs $N_{epochs}$, early stopping criterion, number of minibatches $N_{mb} = (1-\alpha)N_s/N_b$.
Output: Optimal model parameters $\boldsymbol{\theta}^* = (\boldsymbol{\theta}^*_E, \boldsymbol{\theta}^*_{DF}, \boldsymbol{\theta}^*_D)$.
1: Randomly shuffle $\mathbf{M}$ and $\mathbf{S}$
2: Split data in $\mathbf{M} = [\mathbf{M}^{train}, \mathbf{M}^{val}]$ and $\mathbf{S} = [\mathbf{S}^{train}, \mathbf{S}^{val}]$ $(\mathbf{M}^{val}, \mathbf{S}^{val} \in \mathbb{R}^{N_h \times \alpha N_s})$
3: Normalize data in $\mathbf{M}$ and $\mathbf{S}$ according to (3.11)
4: Randomly initialize $\boldsymbol{\theta}^0 = (\boldsymbol{\theta}^0_E, \boldsymbol{\theta}^0_{DF}, \boldsymbol{\theta}^0_D)$
5: $n_e = 0$
6: while ($\neg$early-stopping and $n_e \leq N_{epochs}$) do
7:     for $k = 1 : N_{mb}$ do
8:         Sample a minibatch $(\mathbf{M}^{batch}, \mathbf{S}^{batch}) \subseteq (\mathbf{M}^{train}, \mathbf{S}^{train})$
9:         $\mathbf{S}^{batch} = \text{reshape}(\mathbf{S}^{batch})$
10:        $\widetilde{\mathbf{S}}^{batch}_n(\boldsymbol{\theta}_E^{N_{mb}n_e+k}) = \mathbf{f}^E_n(\mathbf{S}^{batch}; \boldsymbol{\theta}_E^{N_{mb}n_e+k})$
11:        $\mathbf{S}^{batch}_n(\boldsymbol{\theta}_{DF}^{N_{mb}n_e+k}) = \boldsymbol{\phi}^{DF}_n(\mathbf{M}^{batch}; \boldsymbol{\theta}_{DF}^{N_{mb}n_e+k})$
12:        $\widetilde{\mathbf{S}}^{batch}_h(\boldsymbol{\theta}_{DF}^{N_{mb}n_e+k}, \boldsymbol{\theta}_D^{N_{mb}n_e+k}) = \mathbf{f}^D_h(\mathbf{S}^{batch}_n(\boldsymbol{\theta}_{DF}^{N_{mb}n_e+k}); \boldsymbol{\theta}_D^{N_{mb}n_e+k})$
13:        $\widetilde{\mathbf{S}}^{batch}_h = \text{reshape}(\widetilde{\mathbf{S}}^{batch}_h)$
14:        Accumulate loss (3.9) on $(\mathbf{M}^{batch}, \mathbf{S}^{batch})$ and compute $\widehat{\nabla}_\theta \mathcal{J}$
15:        $\boldsymbol{\theta}^{N_{mb}n_e+k+1} = \text{ADAM}(\eta, \widehat{\nabla}_\theta \mathcal{J}, \boldsymbol{\theta}^{N_{mb}n_e+k})$
16:    end for
17:    Repeat instructions 9-13 on $(\mathbf{M}^{val}, \mathbf{S}^{val})$ with the updated weights $\boldsymbol{\theta}^{N_{mb}n_e+k+1}$
18:    Accumulate loss (3.9) on $(\mathbf{M}^{val}, \mathbf{S}^{val})$ to evaluate early-stopping criterion
19:    $n_e = n_e + 1$
20: end while
```

**Algorithm 2** DL-ROM testing

```
Input: Testing parameter matrix $\mathbf{M}^{test} \in \mathbb{R}^{(n_{\mu}+1)\times(N_{test}N_t)}$, optimal parameters $(\boldsymbol{\theta}^*_{DF}, \boldsymbol{\theta}^*_D)$.
Output: ROM approximation matrix $\widetilde{\mathbf{S}}_h \in \mathbb{R}^{N_h \times (N_{test}N_t)}$.
1: Load $\boldsymbol{\theta}^*_{DF}$ and $\boldsymbol{\theta}^*_D$
2: $\mathbf{S}_n(\boldsymbol{\theta}^*_{DF}) = \boldsymbol{\phi}^{DF}_n(M^{test}; \boldsymbol{\theta}^*_{DF})$
3: $\widetilde{\mathbf{S}}_h(\boldsymbol{\theta}^*_{DF}, \boldsymbol{\theta}^*_D) = \mathbf{f}^D_h(S_n(\boldsymbol{\theta}^*_{DF}); \boldsymbol{\theta}^*_D)$
4: $\widetilde{\mathbf{S}}_h = \text{reshape}(\widetilde{\mathbf{S}}_h)$
```

## 3.4 Numerical results: benchmark problems

In this Section, we report the numerical results obtained by applying the proposed DL-ROM technique to three parametrized, time-dependent PDE problems, namely *(i)* Burgers equation, *(ii)* a linear transport equation, and *(iii)* the Monodomain equation (1.3). In particular, on this latter test case, we highlight the limitation of POD-Galerkin ROMs associated to the linear superimposition of modes (see Section 2.1) and the ability of the DL-ROM in accurately reconstructing the space and time scales of the transmembrane potential (see Section 2.6). We deal with problems set in $d = 1$ (spatial) dimensions. In these one-dimensional test cases we aim at assessing the numerical accuracy of the DL-ROM approximation, comparing it to the solution provided by a POD-Galerkin ROM, which features linear (possibly, piecewise linear) trial manifolds. In the following Chapters, by considering two and three-dimensional test cases, we instead focus on computational efficiency, by comparing the computational times of DL-ROM to the ones entailed by a POD-Galerkin method.

To evaluate the performance of DL-ROM we rely on the loss function (3.9) and on the following error indicators

- the error indicator $\epsilon_{rel} \in \mathbb{R}$ given by

$$\epsilon_{rel} = \epsilon_{rel}(\mathbf{u}_h, \tilde{\mathbf{u}}_h) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left( \frac{\sqrt{\sum_{k=1}^{N_t} ||\mathbf{u}_h^k(\boldsymbol{\mu}_{test,i}) - \tilde{\mathbf{u}}_h^k(\boldsymbol{\mu}_{test,i})||^2}}{\sqrt{\sum_{k=1}^{N_t} ||\mathbf{u}_h^k(\boldsymbol{\mu}_{test,i})||^2}} \right), \qquad (3.13)$$

- the relative error $\boldsymbol{\epsilon}_k \in \mathbb{R}^{N_h}$, for $k = 1, \dots, N_t$, defined as

$$\boldsymbol{\epsilon}_k = \boldsymbol{\epsilon}_k(\mathbf{u}_h, \tilde{\mathbf{u}}_h) = \frac{|\mathbf{u}_h^k(\boldsymbol{\mu}_{test}) - \tilde{\mathbf{u}}_h^k(\boldsymbol{\mu}_{test})|}{\sqrt{\frac{1}{N_t}\sum_{k=1}^{N_t} ||\mathbf{u}_h^k(\boldsymbol{\mu}_{test})||^2}}. \qquad (3.14)$$

While (3.13) is a synthetic indicator, the quantity defined in (3.14) is instead a function of the space independent variable

### 3.4.1 Test 1: Burgers equation

Let us consider the parametrized one-dimensional nonlinear Burgers equation

$$\begin{cases} \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} - \dfrac{1}{\mu}, \dfrac{\partial^2 u}{\partial x^2} = 0 & (x,t) \in (0,L) \times (0,T), \\ u(0,t) = 0 & t \in (0,T), \\ u(L,t) = 0 & t \in (0,T), \\ u(x,0) = u_0(x) & x \in (0,L), \end{cases} \qquad (3.15)$$

where

$$u_0(x) = \frac{x}{1 + \sqrt{1/A_0}\exp(\mu x^2/4)},$$

with $A_0 = \exp(\mu/8)$, $L = 1$ and $T = 2$. System (3.15) has been discretized in space by means of linear finite elements, with $N_h = 256$ grid points, and in time by means of the Backward Euler scheme, with $N_t = 100$ time instances. The parameter space, to which belongs the single ($n_\mu = 1$) parameter, is given by $\mathcal{P} = [100, 1000]$. We consider $N_{train} = 20$ training-parameter instances uniformly distributed over $\mathcal{P}$ and $N_{test} = 19$ testing-parameter instances, each of them corresponding to the midpoint between two consecutive training-parameter instances.

The configuration of the DL-ROM neural network used for this test case is the following. We choose a 12-layers DFNN equipped with 50 neurons per hidden layer and $n$ neurons in the output layer, where $n$ corresponds to the dimension of the reduced trial manifold. The architectures of the encoder and decoder functions are instead reported in Tables 3.1 and 3.2, and are similar to the ones used in [Lee and Carlberg, 2020].

| layer | input dimension | output dimension | kernel size | #of filters | stride | padding |
|---|---|---|---|---|---|---|
| 1 | [16, 16, 1] | [16, 16, 8] | [5, 5] | 8 | 1 | SAME |
| 2 | [16, 16, 8] | [8, 8, 16] | [5, 5] | 16 | 2 | SAME |
| 3 | [8, 8, 16] | [4, 4, 32] | [5, 5] | 32 | 2 | SAME |
| 4 | [4, 4, 32] | [2, 2, 64] | [5, 5] | 64 | 2 | SAME |
| 5 | $N_h$ | 256 | | | | |
| 6 | 256 | $n$ | | | | |

Table 3.1: Attributes of convolutional and dense layers in the encoder $\mathbf{f}_n^E$.

Problem (3.15) does not represent a remarkably challenging task for linear ROMs, such as the POD-Galerkin method. Indeed, by using the POD method on the snapshot matrix (the

| layer | input dimension | output dimension | kernel size | #of filters | stride | padding |
|---|---|---|---|---|---|---|
| 1 | $n$ | 256 | | | | |
| 2 | 256 | $N_h$ | | | | |
| 3 | [2, 2, 64] | [4, 4, 64] | [5, 5] | 64 | 2 | SAME |
| 4 | [4, 4, 64] | [8, 8, 32] | [5, 5] | 32 | 2 | SAME |
| 5 | [8, 8, 32] | [16, 16, 16] | [5, 5] | 16 | 2 | SAME |
| 6 | [16, 16, 16] | [16, 16, 1] | [5, 5] | 1 | 1 | SAME |

Table 3.2: Attributes of dense and transposed convolutional layers in the decoder $\mathbf{f}_h^D$.

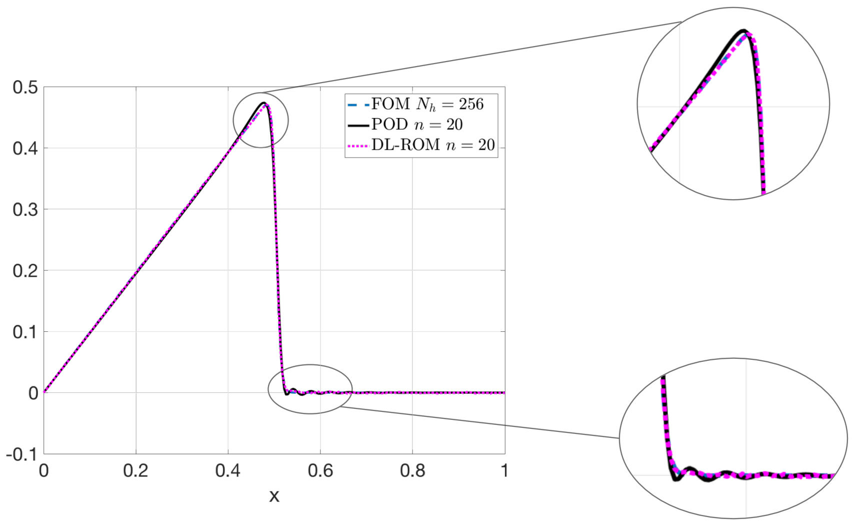


Figure 3.3: *Test 1*: FOM, optimal-POD and DL-ROM solutions for the testing-parameter instance $\mu_{test} = 976.32$ at $t = 0.02$, with $n = 20$.

latter built by collecting the solution of (3.15) $N_{train}$ training-parameter instances), we find that a linear trial manifold of dimension 20 is enough to capture more than the 99.99% of the energy of the system [San and Maulik, 2018, Quarteroni et al., 2016]. In order to assess the DL-ROM performance, we compute the DL-ROM solution by fixing the dimension of the nonlinear trial manifold to $n = 20$. In Figure 3.3 we compare the DL-ROM and the FOM solutions, with the optimal-POD reconstruction (that is, the projection of the FOM solution onto the POD linear trial manifold of dimension 20), for $t = 0.02$ and the testing-parameter instance $\mu_{test} = 976.32$.

The latter testing value has been selected as the instance of $\mu$ for which the reconstruction task results to be the most difficult both for POD and DL-ROM, being the diffusion term in (3.15) smaller and the solution closer to the one of a purely hyperbolic system. In particular, for $\mu_{test} = 976.32$, employing the DL-ROM technique allows us to halve the error indicator $\epsilon_{rel}$ associated to the optimal-POD reconstruction. Referring to Figure 3.3, the DL-ROM approximation is more accurate than the optimal POD reconstruction, indeed it mostly fits the FOM solution, even in correspondence of its maximum, as shown in Figure 3.3. Moreover, it does not introduce oscillations where a large gradient of the FOM solution is observed, as it happens instead by employing POD. The same comparison of Figure 3.3, but with a reduced dimension $n = 10$, is shown in Figure 3.4, where the difference in terms of accuracy provided by the two approaches is even more remarkable.

Finally, in Figure 3.5 we highlight the accuracy properties of both the DL-ROM and

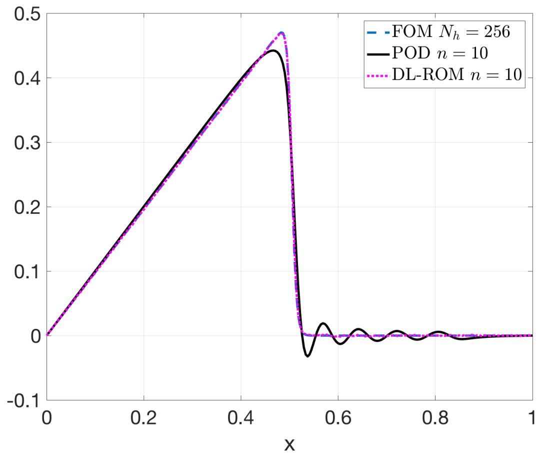


Figure 3.4: *Test 1*: FOM, optimal-POD and DL-ROM solutions for the testing-parameter instance $\mu_{test} = 976.32$ at $t = 0.02$, with $n = 10$.

POD techniques by displaying the behavior of the error indicator $\epsilon_{rel}$, defined in (3.13), with respect to the dimension $n$ of the corresponding reduced trial manifold. For $n < 20$ the DL-ROM approximation is more accurate than the one provided by POD, and only for $n = 20$ the two techniques provide almost the same accuracy.

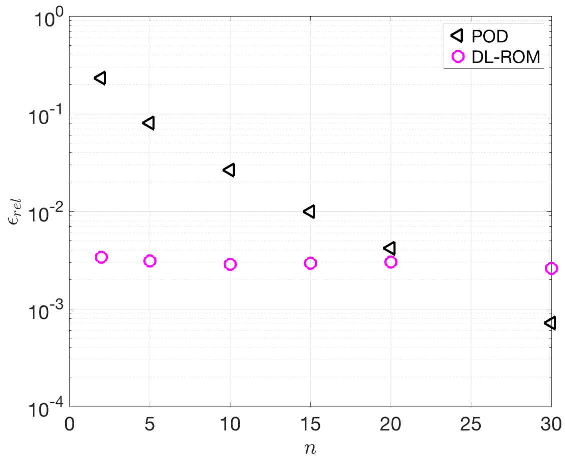


Figure 3.5: *Test 1*: Error indicator $\epsilon_{rel}$ vs. $n$ on the testing set.

### 3.4.2 Test 2: Linear transport equation

#### Test 2.1: $n_\mu = 1$ input parameter

First, we consider the parametrized one-dimensional linear transport equation

$$\begin{cases} \dfrac{\partial u}{\partial t} + \mu \dfrac{\partial u}{\partial x} = 0 & (x,t) \in \mathbb{R} \times (0,T), \\ u(x,0) = u_0(x) & x \in \mathbb{R}, \end{cases} \tag{3.16}$$

whose solution is $u(x,t) = u_0(x - \mu t)$; here $u_0(x) = (1/\sqrt{2\pi\sigma})e^{-x^2/2\sigma}$ and $T = 1$.

The parameter represents the velocity of the travelling wave, varying in the parameter space $\mathcal{P} = [0.775, 1.25]$; we set $\sigma = 10^{-4}$. The dataset is built by uniformly sampling the exact solution in the domain $(0, L) \times (0, T)$, with $L = 1$, considering $N_h = 256$ degrees of freedom in the space discretization and $N_t = 200$ time instances. We consider $N_{train} = 20$ training-parameter instances uniformly distributed over $\mathcal{P}$ and $N_{test} = 19$ testing-parameter instances such that $\mu_{test,i} = (\mu_{train,i} + \mu_{train,i+1})/2$, for $i = 1, \ldots, N_{test}$. This test case, and more in general hyperbolic problems, are examples in which the use of a linear approach to ROM might yield a loss of accuracy. Indeed, the dimension of the linear trial manifold must be very large, if compared to the dimension of the solution manifold, in order to capture the variability of the FOM solution over the parameter space $\mathcal{P}$.

Figure 3.6 shows the exact solution and the DL-ROM approximation for the testing-parameter instance $\mu_{test} = 0.8625$; here, we set the dimension of the nonlinear trial manifold to $n = 2$, equal to the dimension $n_\mu + 1$ of the solution manifold. Moreover, in Figure 3.6 we also report the relative error $\boldsymbol{\epsilon}_k \in \mathbb{R}^{N_h}$ (3.14), for $k = 1, \ldots, N_t$, associated to the selected $\mu_{test} \in \mathcal{P}$, whose largest values are found in proximity of the largest variations of the solution.

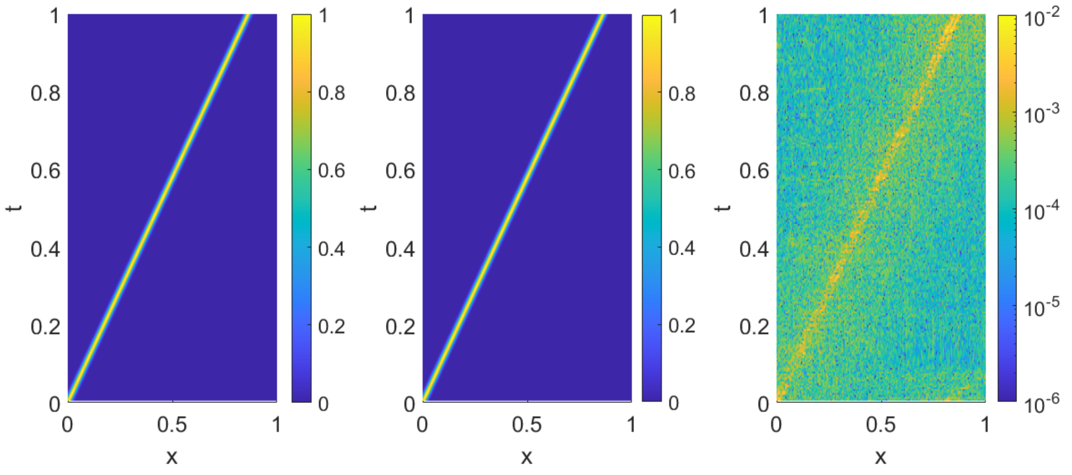


Figure 3.6: *Test 2.1*: Exact solution (left), DL-ROM solution with $n = 2$ (center) and relative error $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instance $\mu_{test} = 0.8625$ in the space-time domain.

In Figure 3.7 we report the exact solution and the DL-ROM approximation, with $n = 2$, at three particular time instances. To compare the performance of DL-ROM with a linear ROM, we performed POD on the snapshot matrix and report, for the same testing-parameter instance, the optimal-POD reconstruction (that is, the projection of the exact solution onto the POD linear trial manifold). Still with $n = 50$ POD modes, the optimal-POD reconstruction is affected by spurious oscillations. On the other hand, the DL-ROM approximation with $n = 2$ yields an error indicator $\epsilon_{rel} = 8.74 \cdot 10^{-3}$; to achieve the same accuracy obtained through DL-ROM over the testing set, a linear trial manifold should have dimension $n = 90$.

Figure 3.8 shows the behavior of the error indicator (3.13) with respect to the reduced dimension $n$. By increasing the dimension $n$ of the nonlinear trial manifold there is a mild improvement of the DL-ROM performance, i.e. the error indicator slight decreases; however, such an improvement is not significant, in general: in this range of $n$, indeed, the number of parameters (i.e., weights and biases) of the DL-ROM neural network slight increases, thus implying almost the same approximation capability of the neural network.

**Remark 2** *(Hyperparameters tuning). The hyperparameters of the DL-ROM neural network are tuned by evaluating the loss function over the validation set and by setting each of them equal to the value minimizing the generalization error on the validation set. In particular, we show the tests performed to choose the size of the (transposed) convolutional kernels in the*

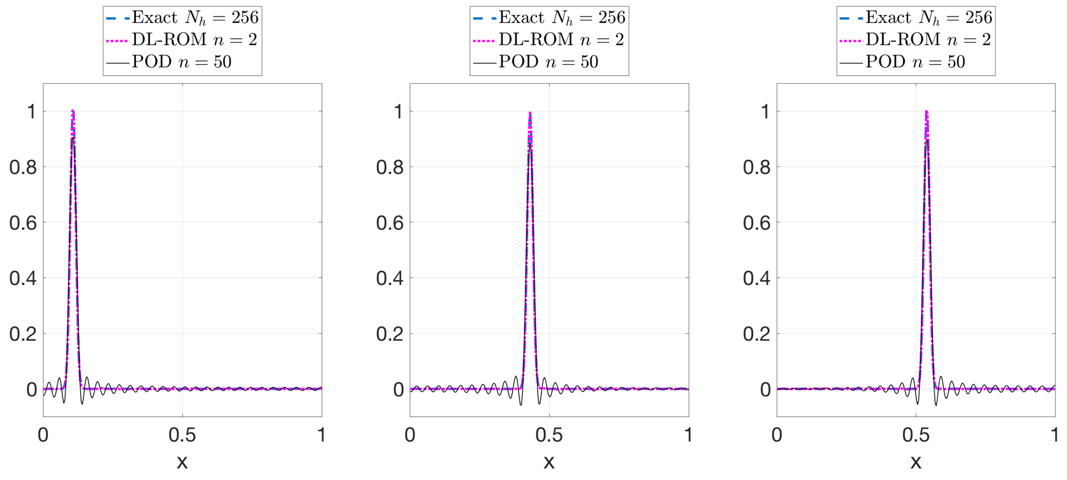


Figure 3.7: *Test 2.1*: Exact solution, DL-ROM approximation and optimal-POD reconstruction for the testing-parameter instance $\mu_{test} = 0.8625$ at $t = 0.125, 0.5$ and $0.625$.

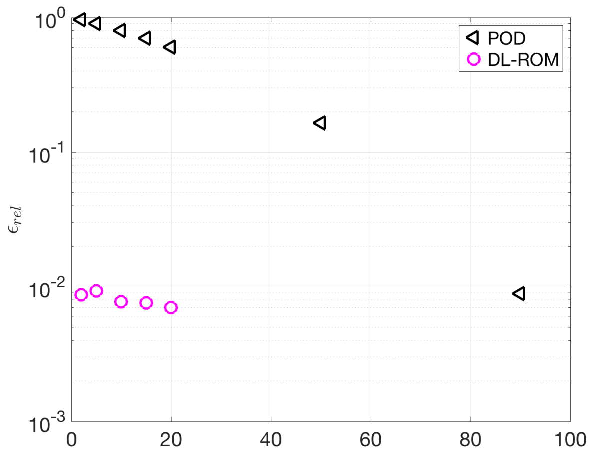


Figure 3.8: *Test 2.1*: Error indicator $\epsilon_{rel}$ vs. $n$ on the testing set.

*(decoder) encoder function, the number of hidden layers in the feedforward neural network and the number of neurons for each hidden layer. The hyperparameters evaluation starts from the default configuration in Table 2.*

| kernel size | #hidden layers | #neurons |
|---|---|---|
| [3, 3] | 1 | 50 |

Table 3.3: *Test 2.1*: Starting configuration of DL-ROM.

*Then, the best values are found iteratively by inspecting the impact of the variation of a single hyperparameter at a time on the validation loss. Once the best value of each hyperparameter is found, it replaces the default value from that point on. For each hyperparameter the tuning is performed in a range of values for which the training of the network is computationally affordable.*

*In Figure 3.9, we show the impact of the size of the convolutional kernels on the loss over the validation and testing sets, the number of hidden layers in the FDNN and the number of* *neurons in each hidden layer by varying the reduced dimension in order to find the best value of such hyperparameter over n. The final configuration of the DL-ROM neural network is the one provided in Table 2.*

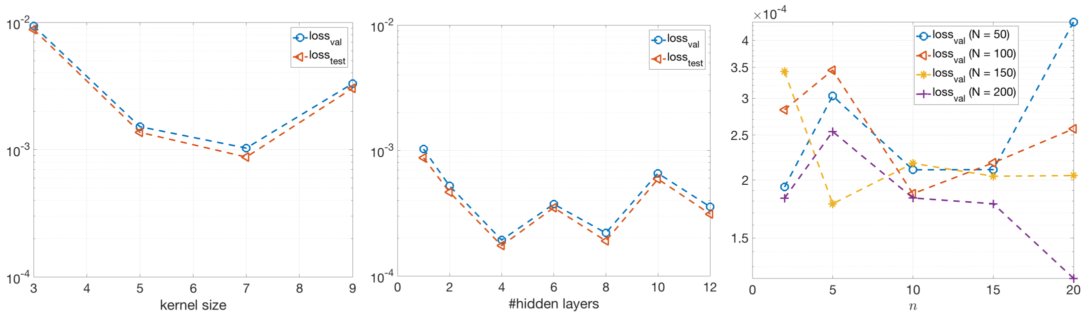


Figure 3.9: *Test 2.1*: Impact of the kernel size (left), the number of hidden layers (center) and the number of neurons (right) on the validation and testing losses.

| kernel size | #hidden layers | #neurons |
|---|---|---|
| [7, 7] | 4 | 200 |

Table 3.4: *Test 2.1*: Final configuration of DL-ROM.

### Test 2.2: $n_\mu = 2$ input parameters

Here we consider again the parametrized one-dimensional transport equation

$$\begin{cases} \dfrac{\partial u}{\partial t} + \dfrac{\partial u}{\partial x} = 0 & (x,t) \in \mathbb{R} \times (0,T), \\ u(x,0) = u_0(x) & x \in \mathbb{R}, \end{cases} \tag{3.17}$$

whose exact solution is $u(x,t) = u_0(x-t;\boldsymbol{\mu})$; however, we now take

$$u_0(x;\boldsymbol{\mu}) = \begin{cases} 0 & \text{if } x < \mu_1, \\ \mu_2 & \text{if } x \geq \mu_1, \end{cases} \tag{3.18}$$

as initial datum, where $\boldsymbol{\mu} = [\mu_1, \mu_2]^T$. The $n_\mu = 2$ parameters belong to the parameter space $\mathcal{P} = \mathcal{P}_{\mu_1} \times \mathcal{P}_{\mu_2} = [0.025, 0.25] \times [0.5, 1]$. We build the dataset by uniformly sampling the exact solution in the domain $(0,L) \times (0,T)$, with $L = 1$ and $T = 1$, and by considering $N_h = 256$ grid points for the space discretization and $N_t = 100$ time instances. We collect, both for $\mu_1$ and $\mu_2$, $N_{train} = 21$ training-parameter instances uniformly distributed in the parameter space $\mathcal{P}$ and $N_{test} = 20$ testing-parameter instances, selected as in the other test cases. Equation (3.17), completed with the initial datum (3.18), represents a challenging test bed for linear ROMs because of the difficulty to accurately reconstruct the jump discontinuity of the exact solution as a linear combination of basis functions computed from the snapshots, for a testing-parameter instance. The architecture of the DL-ROM neural network used here is the one presented in the Test 2.1.

In Figure 3.10 we show the exact solution and the DL-ROM approximation obtained by setting $n = 3$ (thus equal to the dimension of the solution manifold $n_\mu + 1$) for the testing-parameter instance $\boldsymbol{\mu}_{test} = (0.154375, 0.6375)$, along with the relative error $\boldsymbol{\epsilon}_k$, defined in (3.14). Also in this case, the relative error is larger close to the solution discontinuity.

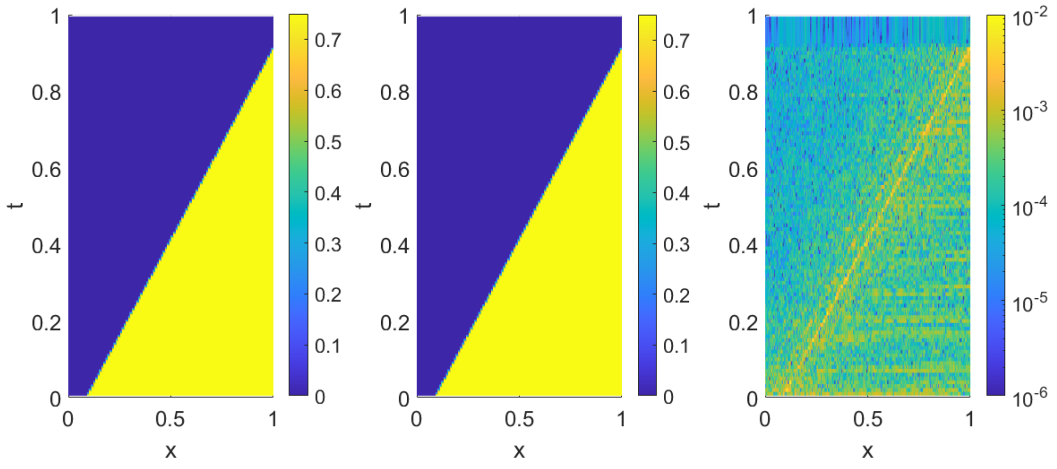


Figure 3.10: *Test 2.2*: Exact solution (left), DL-ROM solution with $n = 3$ (center) and relative error $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instance $\boldsymbol{\mu}_{test} = (0.154375, 0.6375)$ in the space-time domain.

In Figure 3.11 we report the DL-ROM approximation, the optimal-POD reconstruction and the exact solution, for the time instances $t = 0.245, 0.495$ and $0.745$, and the testing-parameter instance $\boldsymbol{\mu}_{test} = (0.154375, 0.6375)$. The dimension of the reduced manifolds are $n = 3$ and $n = 50$ for the DL-ROM and the POD techniques, respectively. Also in this case, even by setting the dimension of the linear manifold equal to $n = 50$, the reconstructed solution presents spurious oscillations. Moreover, the optimal-POD reconstruction is not able to fit the discontinuity of the exact solution in a sharp way. These oscillations are significantly mitigated by the use of our DL-ROM technique, which is able to fit the jump discontinuity accurately, as shown in Figure 3.11.

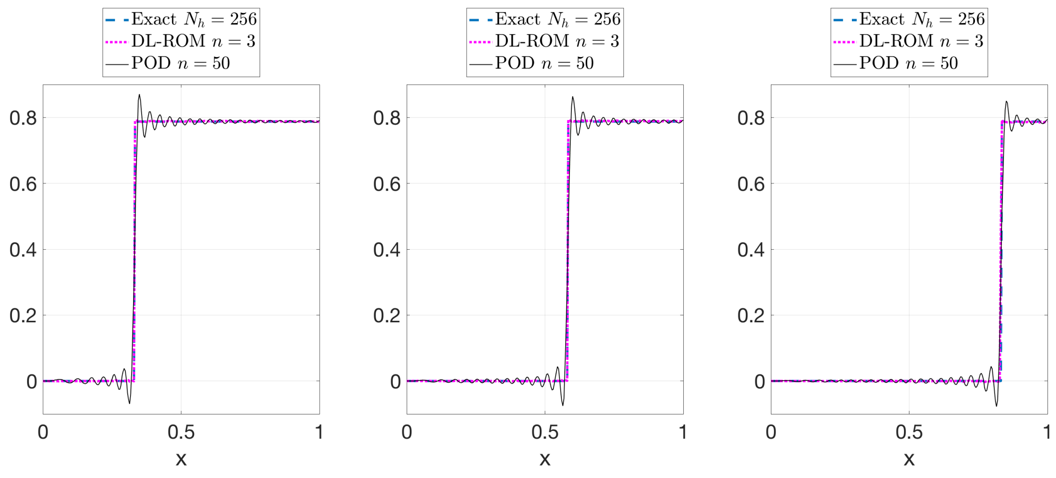


Figure 3.11: *Test 2.2*: Exact, DL-ROM and optimal-POD solutions for the testing-parameter instance $\boldsymbol{\mu}_{test} = (0.154375, 0.6375)$ at $t = 0.245, 0.495$ and $0.745$.

Finally, we can remark (see Figure 3.12) the same behavior of the relative error with respect to the reduced dimension $n$ as in the previous test case. The DL-ROM approximation yields an error indicator $\epsilon_{rel} = 2.85 \cdot 10^{-2}$ with $n = 3$; a similar accuracy would be achieved by POD only through a linear trial manifold of dimension $n = 165$.

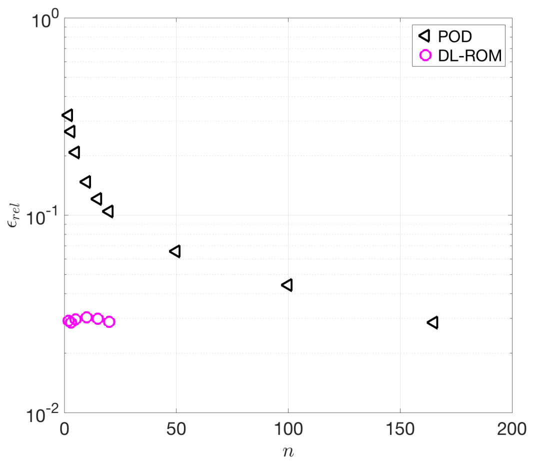


Figure 3.12: *Test 2.2*: Error indicator $\epsilon_{rel}$ vs. $n$ on the testing set.

### 3.4.3 Test 3: Monodomain equation

In the first stage of the assessment of the perfomance of the DL-ROM, we refer to a benchmark one-dimensional test case in cardiac EP. Further exploitations of the developed technique in the context of cardiac EP will come in Chpater 4. In particular, we consider the one-dimensional Monodomain equation (1.3) coupled with the FitzHugh-Nagumo cellular model [FitzHugh, 1961, Nagumo et al., 1962] (see Section 1.2)

$$\begin{cases} \mu \dfrac{\partial u}{\partial t} - \mu^2 \dfrac{\partial^2 u}{\partial x^2} + u(u-0.1)(u-1) + w = 0 & (x,t) \in (0,L) \times (0,T), \\ \dfrac{dw}{dt} + (\gamma w - \beta u) = 0 & (x,t) \in (0,L) \times (0,T), \\ \dfrac{\partial u}{\partial x}(0,t) = 50000 t^3 e^{-15t} & t \in (0,T), \\ \dfrac{\partial u}{\partial x}(L,t) = 0 & t \in (0,T), \\ u(x,0) = 0, \; w(x,0) = 0 & x \in (0,L), \end{cases} \tag{3.19}$$

where $L = 1$, $T = 2$, $\gamma = 2$ and $\beta = 0.5$; the parameter $\mu$ belongs to the parameter space $\mathcal{P} = 5 \cdot [10^{-3}, 10^{-2}]$. System (3.19) has been discretized in space through linear finite elements, by considering $N_h = 256$, i.e. $\dim(X_h) = 256$, and using a one-step, semi-implicit, first order scheme for time discretization; see Section 1.3 for further details[2]. The solution of the former problem consists in a parameter-dependent travelling wave, which exhibits sharper and sharper fronts as the parameter $\mu$ gets smaller (see Figure 3.13).

We consider $N_{train} = 20$ training-parameter instances uniformly distributed in the parameter space $\mathcal{P}$ and $N_{test} = 19$ testing-parameter instances, each of them corresponding to the midpoint between two consecutive training-parameter instances.

Figure 3.14 shows the FOM solution and the DL-ROM one obtained by setting $n = 2$, the dimension of the solution manifold, for the testing-parameter instance $\mu_{test} = 0.0062$. We also report in Figure 3.14 the relative error $\boldsymbol{\epsilon}_k$ (3.14), which takes larger values close to the points where the FOM solution shows steeper gradients. The accuracy obtained by our DL-ROM technique with $n = 2$, and measured by the error indicator on the testing set, is $\epsilon_{rel} = 3.42 \cdot 10^{-3}$.

[2]The `Matlab` library used to compute snapshots and to implement the (local) POD-Galerkin method for problem (3.19) is available at `https://github.com/StefanoPagani/LocalROM`

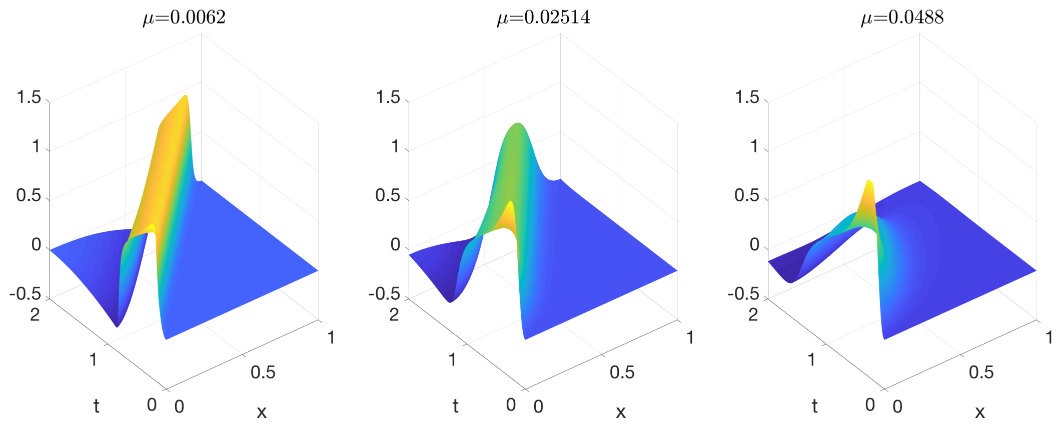


Figure 3.13: *Test 3*: FOM solutions for different testing-parameter instances.

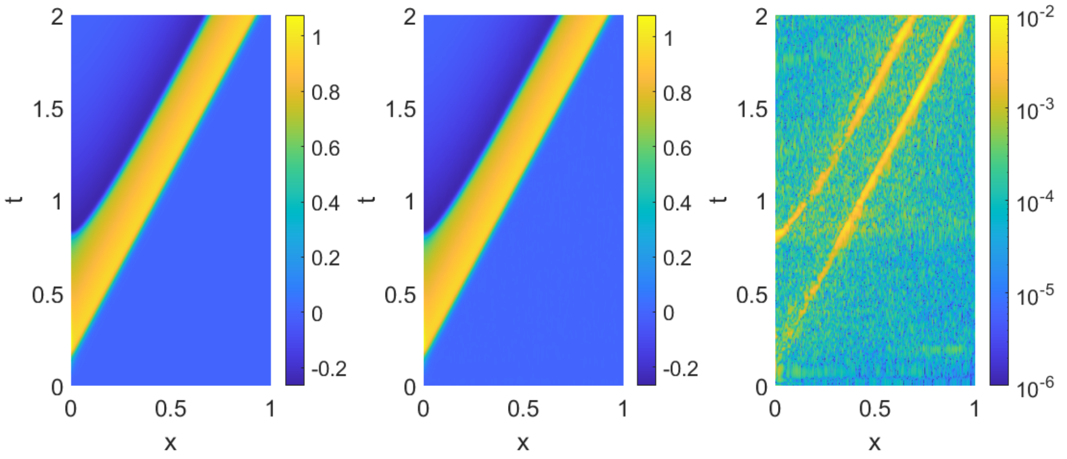


Figure 3.14: *Test 3*: FOM solution (left), DL-ROM solution with $n = 2$ (center) and relative error $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instance $\mu_{test} = 0.0062$ in the space-time domain.

In order to assess the performance of the DL-ROM against a linear ROM, we consider a POD-Galerkin ROM exploiting local reduced bases; these latter are obtained by applying POD to a set of clusters which partition the original snapshot set. In particular, we employ the *k-means* clustering algorithm [Likas et al., 2003], an unsupervised statistical learning technique for finding clusters and cluster centers in an unlabelled dataset, to partition into $N_c$ clusters the snapshots, i.e. the columns of $S$, such that those within each cluster are more closely related to one another than elements assigned to different clusters. In Table 3.5 we report the maximum number of basis functions among all the clusters, i.e. the dimension of the largest linear trial manifold, required by the (local) POD-Galerkin ROM, in order to achieve the same accuracy obtained through a DL-ROM. By increasing the number $N_c$ of clusters, the dimension of the largest linear trial subspace decreases; this does not hold as long as the number of clusters is larger than $N_c = 32$. Indeed, the dimension of some linear subspaces become so small that the error might increase compared to the one obtained with fewer clusters.

In particular, in Figure 3.15 the POD-Galerkin ROM approximations obtained by considering $n = 2$ and $n = 66$ basis functions are shown. In Figure 3.16 we compare the FOM solution for $\mu_{test} = 0.0157$ at $t = 0.4962, 0.9975$ and $1.4987$, with the DL-ROM approximation obtained for $n = 2$, and the POD-Galerkin approximation with a global ($N_c = 1$) linear

| $N_c = 1$ | $N_c = 2$ | $N_c = 4$ | $N_c = 8$ | $N_c = 16$ | $N_c = 32$ |
|---|---|---|---|---|---|
| 66 | 68 | 55 | 34 | 26 | 20 |

Table 3.5: *Test 3*: Maximum number of basis functions for the POD-Galerkin ROM.

trial manifold, of dimension $n = 2, 20$ and 66, respectively.

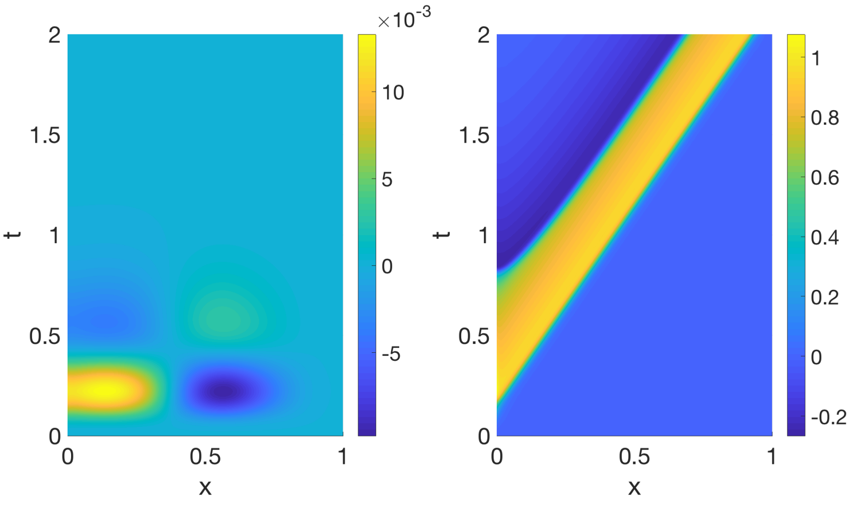


Figure 3.15: *Test 3*: POD-Galerkin ROM solutions for the testing parameter instance $\mu_{test} = 0.0062$ with $n = 2$ (left) and $n = 66$ (right).

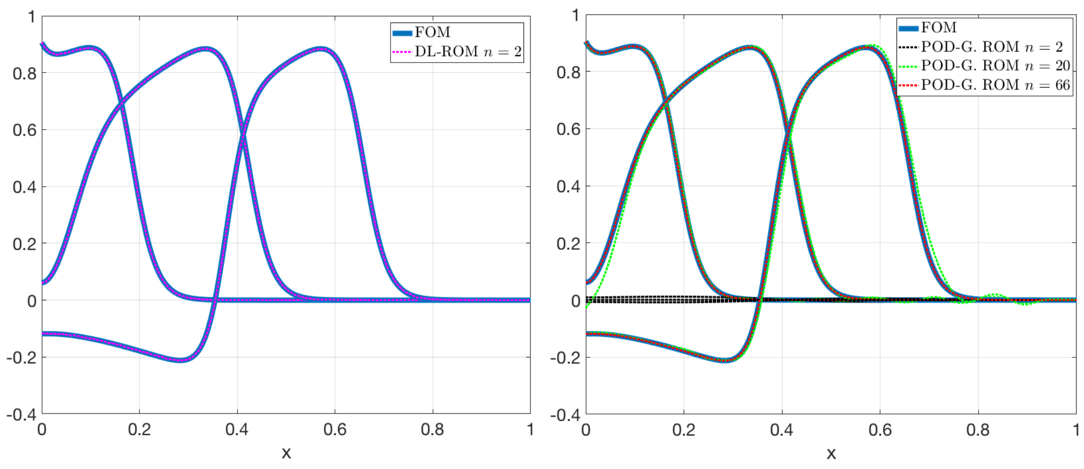


Figure 3.16: *Test 3*: FOM and DL-ROM solutions (left) and FOM and POD-Galerkin ROM solutions (right) for the testing-parameter instance $\mu_{test} = 0.0157$ at $t = 0.4962, 0.9975$ and 1.4987.

The convergence of the error indicator (3.13) as a function of the reduced dimension $n$ is shown in Figure 3.17. For the (local) POD-Galerkin ROM, by increasing the dimension of the largest linear trial manifold, the error indicator decreases; this also occurs for the DL-ROM technique for $n \leq 20$, although the error decay in this latter case is almost negligible, for the same reason pointed out in Test 2.1. If we consider larger values of $n$, e.g. $n = 40$, overfitting might then occur, meaning that the neural network model is too complex with respect to the amount of data provided to it during the training phase. This might explain the slight increase of the error indicator $\epsilon_{rel}$ for $n = 40$. Similar trends are found in [Bhattacharya et al., 2020].

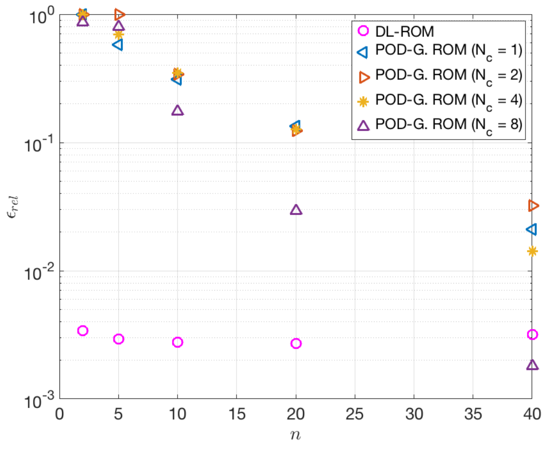


Figure 3.17: *Test 3*: Error indicator $\epsilon_{rel}$ vs. $n$ on the testing set.

Finally, in Figure 3.18 we report the behavior of the loss function and of the error indicator $\epsilon_{rel}$ with respect to the number of training-parameter instances, i.e. the size of the training dataset. By providing more data to the DL-ROM neural network, its approximation capability increases, thus yielding a decrease in the generalization error and the error indicator. In particular, the decay of the loss function with respect to the number of training-parameter instances $N_{train}$ is of about order $1/N^3_{train}$, while the decay of the error indicator (3.13) is of about order $1/N^2_{train}$.

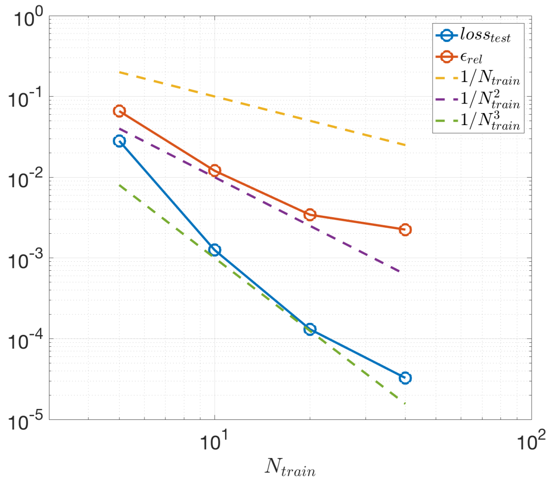


Figure 3.18: *Test 3*: Loss and error indicator $\epsilon_{rel}$ on the testing set vs. number of training-parameter instances of the parameter $\mu$.

**Remark 3** *(Hyperparameters tuning). In order to perform hyperparameters tuning we follow the same procedure used for Test 2.1. We start from the default configuration and we tune the size of the (transposed) convolutional kernels in the (decoder) encoder function, the number of hidden layers in the DFNN and the number of neurons for each hidden layer. In Figure 3.19* *we show the impact of different values of hyperparameters on the validation and testing losses. The final configuration of the DL-ROM neural network is the one provided in Table 3.6.*

| kernel size | #hidden layers | #neurons |
|---|---|---|
| [7, 7] | 1 | 200 |

Table 3.6: *Test 3*: Final configuration of DL-ROM.

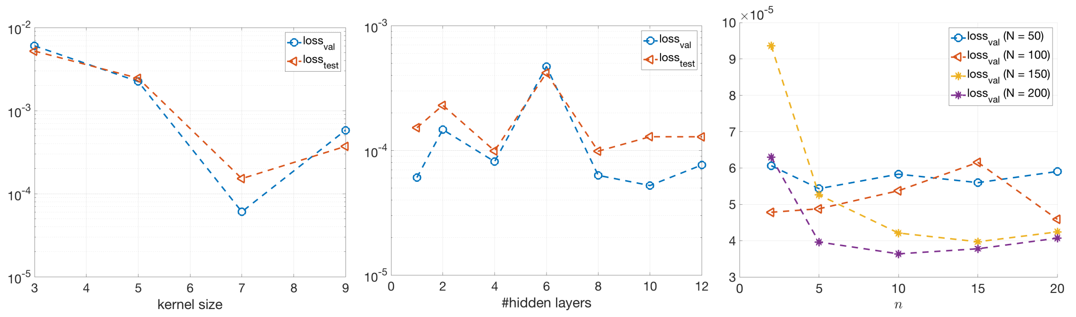


Figure 3.19: *Test 3*: Impact of the kernel size (left), the number of hidden layers (center) and the number of neurons (right) on the validation and testing losses.

**Remark 4** *(Sensitivity with respect to the weight $\omega_h$). For all the test cases, we set the parameter $\omega_h$ in the loss function (3.9) equal to $\omega_h = 1/2$. To justify this choice, we performed a sensitivity analysis for problem (3.19) as shown in Figure 3.20. For extreme values of $\omega_h$, the error indicator (3.13) worsens of about one order of magnitude. In particular, the case $\omega_h = 1$ (that is, not considering the contribution of the encoder function $\mathbf{f}_n^E$ in the loss) yields worse DL-ROM performance; similarly, the case $\omega_h = 0$ would neglect the reconstruction error (that is, the first term in the per-example loss function 3.9) – this is why the error indicator is large for $\omega_h = 0.1$. All the values of $\omega_h$ in the range $[0.2, 0.9]$ do not yield significant differences in terms of error indicator, so we decided to set $\omega_h = 1/2$.*

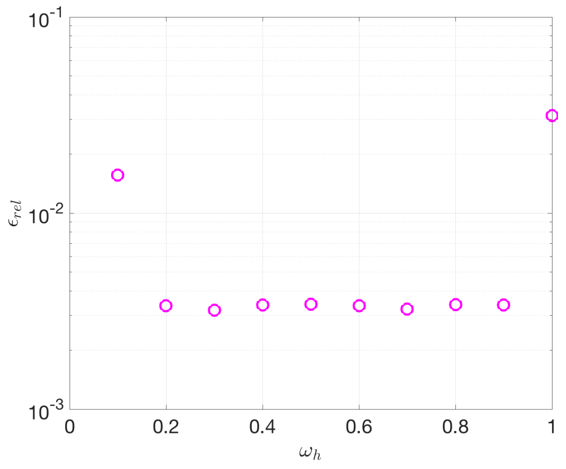


Figure 3.20: *Test 3*: Error indicator $\epsilon_{rel}$ vs. $\omega_h$.

## 3.5 Discussion

In this Chapter we proposed a novel technique to build low-dimensional ROMs by exploiting DL algorithms, which we refer to as DL-ROM, to overcome typical computational bottlenecks shown by classical, linear projection-based ROM techniques (such as POD-Galerkin ROMs) when dealing with problems featuring coherent structures propagating over time. The numerical results obtained for three different test cases show that the proposed DL-ROM technique provides sufficiently accurate solutions to the parametrized PDEs involving a low-dimensional solution manifold whose dimension is equal to (or slightly larger than) the solution manifold $n_\mu + 1$. The proposed DL-ROM outperforms linear ROMs such as the RB method (relying on a global POD basis), as well as nonlinear approaches exploiting local POD bases, when applied both to *(i)* problems which are challenging for linear ROMs, such as the linear transport equation or the Monodomain equation, and *(ii)* problems which are more tractable using a linear ROM, like Burgers equation, however featuring POD bases with much higher dimension. The proposed DL-ROM technique provides approximations that are orders of magnitude more accurate than the ones provided by linear ROMs, when keeping the same dimension. Error decrements are moderate when considering low-dimensional spaces of increasing dimensions, thus making, in the numerical tests considered, the accuracy of both approximations comparable when dealing with $\mathcal{O}(10^2)$ POD basis functions. Furthermore, compared to POD-Galerkin ROMs, our DL-ROM technique completely avoids the use of (very often expensive) hyper-reduction techniques.

Our numerical assessment has shown that employing DL techniques to build ROMs for nonlinear parametrized PDEs is indeed a feasible way, in terms of numerical accuracy; this is a fundamental step toward the application of the DL-ROM technique to cardiac EP problems involving more complex solutions and a larger number $N_h$ of DOFs (see Chapter 4).

Chapter 4

# DL-ROM numerical results in cardiac electrophysiology

In this Chapter[1], we assess the computational performance of the DL-ROM strategy, proposed in Chapter 3, on five relevant test cases in cardiac electrophysiology (EP), by focusing on computational times. We start by detailing the construction of projection-based reduced order models (ROMs), such as POD-Galerkin ROMs, considering the parametric spatial discretization of the Monodomain equation (1.3) coupled with the Aliev-Panfilov (A-P) ionic model (1.5).

## 4.1 Parametrized PDEs in cardiac electrophysiology

Solving systems (1.1) and (1.3) using standard numerical methods such as the FE method and IGA (see Chapter 1), is computationally demanding and far from being able to provide solutions or compute outputs of interest in *real-time* applications. Indeed, the propagation of the electrical signal is characterized by the fast dynamics of very steep fronts, thus requiring very fine space and time discretizations [Sundnes et al., 2007, Colli Franzone and Pavarino, 2004, Colli Franzone et al., 2014]; see also, e.g., [Sundnes et al., 2009, Cervi and Spiteri, 2019] for higher-order and/or more robust numerical methods, [Bendahmane et al., 2010] for time and space adaptivity, and [Sachetto Oliveira et al., 2018] regarding the use of GPU computing in this context.

In Section 1.5 we highlight the need for exploring the parameter space, i.e. solving the equations, modeling the propagation of the electrical signal in the heart, for several values of parameters, in order to investagate different scenarios or intra- and inter-subject variability. Solving the equations may quickly become unaffordable if such a coupled system must be solved for several parameters instances. *Multi-query* analysis is relevant in a variety of situations: when analyzing multiple scenarios, when dealing with sensitivity analysis and uncertainty quantification (UQ) problems in order to account for inter-subject variability [Mirams et al., 2016, Johnstone et al., 2016, Hurtado et al., 2017, Clayton et al., 2020], for parameter estimation or data assimilation, in which some unknown (or unaccessible) quantities characterizing the mathematical model must be inferred from a set of measurements [Dhamala et al., 2018, Quaglino et al., 2018, Johnston et al., 2018, Pathmanathan et al., 2019, Levrero-Florencio et al., 2020]. In all these cases, to achieve computational efficiency, multi-query analysis in cardiac EP must rely on suitable *surrogate* models see, e.g., [Niederer et al., 2020] for a recent review on the topic. Among surrogate models, several options are

[1]This Chapter is mainly based on the paper [Fresca et al., 2020b].

available, such as *(i)* emulators, obtained, e.g., via Polynomial Chaos Expansions, GP regression or neural networks [Coveney et al., 2020, Longobardi et al., 2020, Lei et al., 2020, Ayed et al., 2019], aiming at the approximation of the input-output mapping by fitting a set of training data, including possibily governing laws in its definition [Sahli Costabal et al., 2020]; *(ii)* lower-fidelity models, introducing suitable modeling simplifications – such as, for instance, the Eikonal model in this context [Neic et al., 2017]; and *(iii)* ROMs obtained through a projection process on the equations governing the phenomenon under consideration to reduce the state-space dimensionality, see Chapter 2 for further details. Although typically more intrusive to implement, ROMs often yield more accurate approximations than data fitting and usually generate more significant computational gains than lower-fidelity models.

Conventional projection-based ROMs built, e.g., through the RB method [Quarteroni et al., 2016], yields inefficient ROMs when dealing with nonlinear time-dependent parametrized PDE-ODE system as the one arising from cardiac EP. The three major computational bottlenecks shown by such kind of ROMs for cardiac EP are:

- the linear superimposition of modes, on which they are based, would cause the dimension of the ROM to be excessively large to guarantee an acceptable accuracy;
- evaluating the ROM requires the solution of a dynamical system, which might be unstable unless the size of time-step $\Delta t$ is very small;
- the ROM must also account for the dynamics of the gating variables, even when aiming at computing just the electrical potential. This fact entails an extremely intrusive and costly hyper-reduction stage to reduce the solution of the ODE system to a few, selected mesh nodes [Pagani et al., 2018].

To overcome the limitations of projection-based ROMs, we apply the new, non-intrusive ROM technique based on DL algorithms, which we refer to as DL-ROM, introduced in chapter 3. Combining in a suitable way a convolutional AE and a DFNN, the DL-ROM technique enables the construction of an efficient ROM, whose dimension is as close as possible to the number of parameters upon which the solution of the differential problem depends. A preliminary numerical assessment of our DL-ROM technique has already been presented in Section 3.4, albeit on simpler – yet challenging – test cases.

The proposed DL-ROM technique combines data-driven and physics-based models. Indeed, it exploits snapshots taken from a set of FOM solutions (for selected parameter values and time instances) and deep neural network architectures to learn, in a non-intrusive way, both *(i)* the nonlinear trial manifold where the ROM solution is sought, and *(ii)* the nonlinear reduced dynamics. In a linear ROM built, e.g., through POD, the former quantity is nothing but a set of basis functions, while the latter task corresponds to the projection stage in the subspace spanned by these basis functions. Here, our goal is to show that DL-ROM can be effectively used to handle parametrized problems in cardiac EP, accounting for both physiological and pathological conditions, in order to provide fast and accurate solutions. The proposed DL-ROM is computationally efficient during the testing stage, that is for any new scenario unseen during the training stage. This is particularly useful in view of the evaluation of patient-specific features to enable the integration of computational methods in current clinical platforms.

## 4.2 Projection-based ROMs in cardiac electrophysiology

We start from the general setting of linear (projection-based) ROMs provided in Section 2.3, and formulate the construction of a POD-Galerkin ROM on the specific case of the parametrized Monodomain equation. From an algebraic standpoint and by introducing the parameter vector $\boldsymbol{\mu} \in \mathcal{P}$, the spatial discretization of the parameter-dependent version of system (1.3) through the Galerkin-FE approximation [Quarteroni and Valli, 1994] yields the following nonlinear dynamical system for $\mathbf{u}_h = \mathbf{u}_h(t;\boldsymbol{\mu})$, $\mathbf{w}_h = \mathbf{w}_h(t;\boldsymbol{\mu})$, representing our FOM:

$$\begin{cases} \mathbf{M}(\boldsymbol{\mu})\dfrac{\partial \mathbf{u}_h}{\partial t} + \mathbf{A}(\boldsymbol{\mu})\mathbf{u}_h + \mathbf{I}_{ion}(t, \mathbf{u}_h, \mathbf{w}_h; \boldsymbol{\mu}) = \mathbf{I}_{app}(t;\boldsymbol{\mu}) & t \in (0,T), \\ \dfrac{\partial \mathbf{w}_h}{\partial t}(t;\boldsymbol{\mu}) = \mathbf{g}(t, \mathbf{u}_h, \mathbf{w}_h; \boldsymbol{\mu}) & t \in (0,T), \\ \mathbf{u}_h(0) = \mathbf{0} \qquad \mathbf{w}_h(0) = \mathbf{0}. \end{cases} \tag{4.1}$$

Here $\mathbf{A}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ is a matrix arising from the diffusion operator (thus including the conductivity tensor $\mathbf{D}(\boldsymbol{\mu}) = \mathbf{D}(\mathbf{x};\boldsymbol{\mu})$, which can vary within the myocardium due to fibers orientation and conditions, such as the possible presence of ischemic regions); $\mathbf{M}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ is the mass matrix; $\mathbf{I}_{ion}, \mathbf{g} \in \mathbb{R}^{N_h}$ are vectors arising from the nonlinear terms; finally, $\mathbf{I}_{app} \in \mathbb{R}^{N_h}$ is a vector collecting the applied currents. The dimension $N_h$ is related to the dimension of the FE space $X_h$. Note that the system of ODEs arises from the collocation of the ODE in (1.3) at the nodes used for the numerical integration, see Subsection 1.3.1 for further details.

As outlined in Section 2.3, when using a projection-based ROM, the approximation of $\mathbf{u}_h(t;\boldsymbol{\mu})$ is sought as a linear superimposition of modes, under the form

$$\mathbf{u}_h(t;\boldsymbol{\mu}) \approx \mathbf{V}\mathbf{u}_n(t;\boldsymbol{\mu}), \tag{4.2}$$

thus yielding a linear ROM, in which the columns of the matrix $\mathbf{V} = [\boldsymbol{\zeta}_1, \dots, \boldsymbol{\zeta}_n] \in \mathbb{R}^{N_h \times n}$ form an orthonormal basis of a space $V_n$, an $n$-dimensional subspace of $\mathbb{R}^{N_h}$. In the case of POD, $V_n$ provides the best $n$-rank approximation of $\mathbf{S}$ in the Frobenius norm, that is, $\boldsymbol{\zeta}_1, \dots, \boldsymbol{\zeta}_n$ are the first $n$ (left) singular vectors of $\mathbf{S}$ corresponding to the $n$ largest singular values $\sigma_1, \dots, \sigma_n$ of $\mathbf{S}$, such that the projection error is smaller than a desired tolerance $\varepsilon_{POD}$. To meet this requirement, it is sufficient to choose $n$ as the smallest integer such that

$$\frac{\sum_{i=1}^{N} \sigma_i^2}{\sum_{i=1}^{N_s} \sigma_i^2} > 1 - \varepsilon_{POD}^2,$$

i.e., the energy retained by the last $N_s - n$ POD modes is equal or smaller than $\varepsilon_{POD}^2$.

The approximation of $\mathbf{w}_h$ is given instead by its restriction

$$\mathbf{w}_h(t;\boldsymbol{\mu}) \approx \mathbf{P}\mathbf{w}_{h,\mathcal{I}}(t;\boldsymbol{\mu}),$$

to a (possibly, small) subset $\mathcal{I}$ of $m$ degrees of freedom, where $m \ll N_h$, at which the nonlinear term $\mathbf{I}_{ion}$ is interpolated exploiting a problem-dependent basis, spanned by the columns of a matrix $\boldsymbol{\Phi} \in \mathbb{R}^{N_h \times m}$, which is built according to a suitable hyper-reduction strategy, such as DEIM. By referring to Subsection 2.3.1, the DEIM approximation of the ionic term in the potential equation in (1.3) reads

$$\mathbf{V}^T \mathbf{I}_{ion}(t, \mathbf{V}\mathbf{u}_n, \mathbf{w}_h; \boldsymbol{\mu}) \approx \mathbf{V}^T \boldsymbol{\Phi}(\mathbf{P}^T\boldsymbol{\Phi})^{-1} \mathbf{I}_{ion}(t, \mathbf{P}^T\mathbf{V}\mathbf{u}_n, \mathbf{P}^T\mathbf{w}_h; \boldsymbol{\mu}).$$

A POD-Galerkin ROM for system (1.3) is then obtained by *(i)* first, substituting equation (4.2) into equation (4.1) and projecting it onto $V_n$; then, *(ii)* solving the system of ODEs at $m$ selected degrees of freedom, thus yielding the following nonlinear dynamical system for $\mathbf{u}_n = \mathbf{u}_n(t;\boldsymbol{\mu})$ and the selected components $\mathbf{P}^T\mathbf{w}_h = \mathbf{P}^T\mathbf{w}_h(t;\boldsymbol{\mu})$ of $\mathbf{w}_h$:

$$\begin{cases} \mathbf{V}^T\mathbf{M}(\boldsymbol{\mu})\mathbf{V}\dfrac{\partial \mathbf{u}_n}{\partial t} + \mathbf{V}^T\mathbf{A}(\boldsymbol{\mu})\mathbf{V}^T\mathbf{u}_n & \\ \quad + \mathbf{V}^T\boldsymbol{\Phi}(\mathbf{P}^T\boldsymbol{\Phi})^{-1}\mathbf{I}_{ion}(t, \mathbf{P}^T\mathbf{V}\mathbf{u}_n, \mathbf{P}^T\mathbf{w}_h; \boldsymbol{\mu}) - \mathbf{V}^T\mathbf{I}_{app}(t;\boldsymbol{\mu}) = \mathbf{0} & t \in (0,T), \\ \mathbf{P}^T\dfrac{\partial \mathbf{w}_h}{\partial t} + \mathbf{g}(t, \mathbf{P}^T\mathbf{V}\mathbf{u}_n, \mathbf{P}^T\mathbf{w}_h; \boldsymbol{\mu}) = \mathbf{0} & t \in (0,T), \\ \mathbf{u}_n(0) = \mathbf{0} \qquad \mathbf{P}^T\mathbf{w}_h(0) = \mathbf{0}. \end{cases} \tag{4.3}$$

Using (4.3) as an approximation to (4.1) is known to suffer from several problems. First of all, an extensive hyper-reduction stage, exploiting, e.g., DEIM (see Section 2.3.1 for further details), must be performed in order to be able to evaluate any $\boldsymbol{\mu}$- or $\mathbf{u}_h$-dependent quantities appearing in (4.3), that is, without relying on $N_h$-dimensional arrays. Moreover, whenever the solution of the differential problem features coherent structures that propagate over time, such as steep wavefronts, the dimension $n$ of the projection-based ROM (4.3) might easily become very large (see Test 3 of Chapter 3 for the results on a benchmark one-dimensional Monodomain equation test case), due to the basic linearity assumption, by which the solution is given by a linear superimposition of POD modes, thus severely degrading the computational efficiency of the ROM. A possible way to partially overcome this bottleneck is to rely on local reduced bases, built through POD after the set of snapshots has been split into $N_c > 1$ clusters, according to suitable clustering (or unsupervised learning) algorithms [Pagani et al., 2018].

## 4.3 Numerical results

Our choice of the numerical tests is aimed at highlighting the performance of the DL-ROM technique, proposed in Chapter 3, on challenging EP problems, namely pathological cases in portion of cardiac tissue or physiological scenarios on realistic LV geometries.

The configuration of the DL-ROM neural network, together with the values of the hyper-parameters, used for our numerical tests is the one provided in Subsection 3.4.1. To evaluate the performance of the DL-ROM, we use the loss function (3.9) and the error indicators defined in (3.13) and (3.14).

The DL-ROM neural network has been implemented by means of the `Tensorflow` DL framework [Abadi et al., 2016]. The training phase has been carried out on a workstation equipped with an Nvidia GeForce GTX 1070 8 GB GPU while, in addition to this hardware, the testing phase has also been carried out on a HPC cluster. The code is freely available at `https://github.com/stefaniafresca/DL-ROM`.

As highlighted in Section 1.3, the Monodomain equation coupled with the A-P ionic model has been discretized in space through linear finite elements. For the time discretization and the treatment of nonlinear terms, we use a one-step, semi-implicit, first order scheme.

### 4.3.1 Test 1: Two-dimensional slab

We now focus on the two-dimensional Monodomain equation (1.3) coupled to the A-P ionic model (1.5). Here, we consider a square domain $\Omega = (0, 10 \text{ cm})^2$ and two ($n_\mu = 2$) parameters, consisting in the electric conductivities in the longitudinal and the transversal directions to the fibers, i.e., the conductivity tensor $\mathbf{D}(\mathbf{x}; \boldsymbol{\mu})$ takes the form

$$\mathbf{D}(\mathbf{x}; \boldsymbol{\mu}) = \mu_2 I + (\mu_1 - \mu_2)\mathbf{f}_0 \otimes \mathbf{f}_0, \tag{4.4}$$

where $\mathbf{f}_0 = (1, 0)^T$ and the parameters space is $\mathcal{P} = 12.9 \cdot [0.02, 0.2] \times 12.9 \cdot [0.01, 0.1]\text{cm}^2/\text{ms}$. The applied current is defined as

$$I_{app}(\mathbf{x}, \tilde{t}) = \frac{C}{2\pi\alpha} \exp\left( - \frac{||\mathbf{x}||^2}{2\beta} \right) \mathbf{1}_{[0,\bar{t}]}(\tilde{t}),$$

where $C = 100$ mA, $\alpha = 1$, $\beta = 1$ cm$^2$ and $\bar{t} = 2$ ms. The parameters of the A-P ionic model are set to $K = 8$, $a = 0.01$, $b = 0.15$, $\varepsilon_0 = 0.002$, $c_1 = 0.2$, and $c_2 = 0.3$, see, e.g., [Göktepe et al., 2010]. The equations have been discretized in space by considering $N_h = 64 \times 64 = 4096$. For the time discretization, we consider a time-step $\Delta t = 0.1/12.9$ over the interval $(0, T)$, with $T = 400$ ms.

For the training phase, we uniformly sample $N_t = 1000$ time instances in the interval $(0, T)$ and consider $N_{train} = 25$ training-parameter, i.e. $\boldsymbol{\mu}_{train} = 12.9 \cdot (0.02 + i0.045, 0.01 + j0.0225)$

with $i, j = 0, \dots, 4$. For the testing phase, $N_{test} = 16$ testing-parameter instances have been considered, each of them given by $\boldsymbol{\mu}_{test} = 12.9 \cdot (0.0425 + i0.045, 0.0212 + j0.0225)$ with $i, j = 0, \dots, 3$. The maximum number of epochs is $N_{epochs} = 10000$, the batch size is $N_b = 40$ and, regarding the early-stopping criterion, we stop the training if the loss function does not decrease along 500 epochs.

In Figure 4.1 we show the FOM and the DL-ROM solutions, the latter obtained with $n = 3$, for the testing-parameter instance $\boldsymbol{\mu}_{test} = 12.9 \cdot (0.088, 0.066)$ cm$^2$/ms and $\boldsymbol{\mu}_{test} = 12.9 \cdot (0.178, 0.066)$ cm$^2$/ms, respectively, at $\tilde{t} = 47.7$ ms, together with the relative error $\boldsymbol{\epsilon}_k$. We remark the variability of the solution of (1.3) over $\mathcal{P}$, characterized by the propagation of a sharp front across the domain, depending on the parameters values.

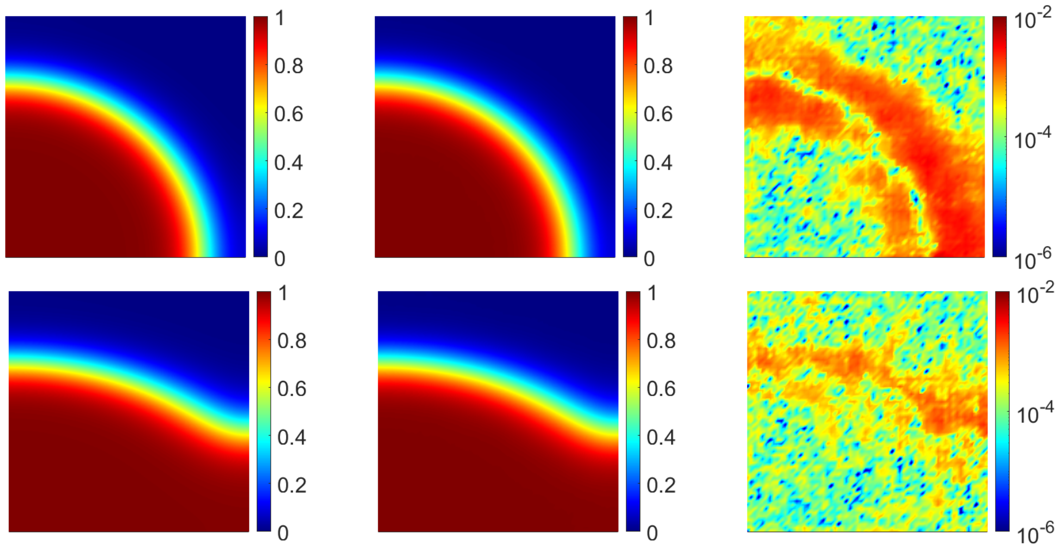


Figure 4.1: *Test 1*: FOM solution (left), DL-ROM solution with $n = 3$ (center) and relative error $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instance $\boldsymbol{\mu}_{test} = 12.9 \cdot (0.088, 0.066)$ cm$^2$/ms (above) and $\boldsymbol{\mu}_{test} = 12.9 \cdot (0.178, 0.066)$ cm$^2$/ms (bottom) at $\tilde{t} = 47.4$ ms.

We now focus on the performance of our DL-ROM technique, in terms of computational efficiency. In Table 4.1 we compare the computational times[2] required to compute the solution for a randomly sampled testing-parameter instance, over the entire time interval $(0, T)$, by the FOM, the (local) POD-Galerkin ROM (for different values of $N_c$) and the DL-ROM, keeping the same degree of accuracy achieved by DL-ROM, i.e. $\epsilon_{rel} = 5.87 \times 10^{-3}$.

| | time [s] | FOM/ROM dimensions |
|---|---|---|
| FOM | 243 | $N_h = 4096$ |
| DL-ROM | 0.45 (1.8) | $n = 3$ |
| POD-Galerkin ROM ($N_c = 1$) | 14 | $n = 87$ |
| POD-Galerkin ROM ($N_c = 2$) | 11 | $n = 58, 49$ |
| POD-Galerkin ROM ($N_c = 4$) | 9 | $n = 44, 33, 31, 29$ |
| POD-Galerkin ROM ($N_c = 6$) | 8 | $n = 38, 33, 27, 26, 21, 6$ |
| POD-Galerkin ROM ($N_c = 8$) | 8 | $n = 30, 25, 24, 22, 21, 20, 19, 6$ |

Table 4.1: *Test 1*: FOM, POD-Galerkin ROM and DL-ROM computational times along with FOM and reduced trial manifold(s) dimensions.

[2]Here we performed our simulations on a full 64 GB node (20 Intel® Xeon® E5-2640 v4 2.4GHz cores) of the HPC cluster available at MOX, Politecnico di Milano.

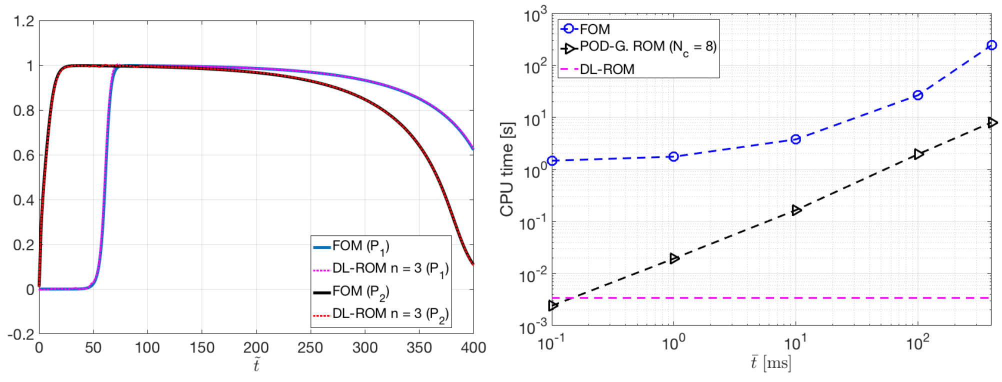


Figure 4.2: *Test 1*: Left: FOM and DL-ROM solutions for the testing-parameter istance $\boldsymbol{\mu}_{test} = 12.9 \cdot (0.088, 0.066)$ cm$^2$/ms at $P_1$ and $P_2$. Right: FOM, (local) POD-Galerkin ROM and DL-ROM computational times to compute $\tilde{\mathbf{u}}_h(\bar{t})$ vs. $\bar{t}$ averaged over the testing set.

We emphasize that the DL-ROM solution can be queried at any desired time instance $\bar{t} \in [0, T]$, without involving the solution of a dynamic system to determine its evolution up to $\bar{t}$, unlike the FOM or the POD-Galerkin ROM. This latter still requires solving for whole range of discrete times in the interval $[0, \bar{t}]$, with time-step size $\Delta t$ dependent on the desired level of accuracy. In other words, when using the DL-ROM we are free to choose a larger time resolution, to reach the same degree of accuracy, with respect to the time-stepping required in the solution of the POD-Galerkin ROM dynamical system. Indeed, the underlying nature of the FOM in the test case at hand implies very small time-step sizes when both solving the POD-Galerkin ROM and sampling the FOM solution for the snapshot matrix assembly. This feature allows to drastically reduce the testing computational time of DL-ROM with respect to the ones required to compute the FOM or the POD-Galerkin ROM solutions at a given time.

The speed-up introduced by the DL-ROM technique with respect to the solution of the FOM is about 138 times, provided that we evaluate the solution at $N_t = 4000$ time steps as in the FOM; the speed-up increases to 536 times if the DL-ROM approximation is computed instead, ensuring the same degree of accuracy, at $N_t = 1000$ time steps. Compared to the use of the local POD-Galerkin ROM in the best case (i.e., with $N_c = 6$ or 8 local bases), DL-ROM leads to almost 30 times faster computations.

The computational gain is even more remarkable regarding the evaluation of the solution at the final time $\bar{t} = T$: the DL-ROM directly provides it, as $\bar{t}$ is an input of the neural network, whereas a POD-Galerkin ROM still require solving for hundreds or thousands of discrete time instances. In Figure 4.2 (right) we show the DL-ROM, FOM and POD-Galerkin ROM CPU time needed to compute the approximated solution at $\bar{t}$, for $\bar{t} = 1$, 10, 100 and 400 ms averaged over the testing set. We perform the training phase of the POD-Galerkin ROM over the original time interval $(0, T)$ ms and we report the results for $N_c = 8$ local bases, for which the smallest computational time is obtained in Table 4.1. The DL-ROM CPU time to compute $\tilde{\mathbf{u}}_h(\bar{t})$ does not vary over $\bar{t}$ and by choosing $\bar{t} = T$ ms the DL-ROM speed-ups are equal to $7.1 \times 10^4$ and $2.4 \times 10^3$ with respect to the FOM and the POD-Galerkin ROM with $N_c = 8$ local bases[3]. In Figure 4.2 (left) we also show the comparison between the FOM solution and the DL-ROM approximation (with $n = 3$) computed at $P_1 = (9.52, 4.76)$ cm and $P_2 = (1.9, 1.11)$ cm, for the testing-parameter instance $\boldsymbol{\mu}_{test} = 12.9 \cdot (0.132, 0.066)$ cm$^2$/ms. The time evolution of the FOM solution is sharply captured by our DL-ROM technique at both locations.

[3]We did not investigate the case $N_c > 8$ due to the fact that the employing $N_c = 6$ or $N_c = 8$ clusters lead to the same testing computational time.

Last but not least, the weaker constraint on time-stepping used in the DL-ROM also has a positive impact on the size of the dataset used for its training phase. For the case at hand, we can train the DL-ROM on a snapshot matrix containing only 25% of the snapshots used to train the POD-Galerkin ROM – taking $N_t = 1000$ instead of 4000 as in the POD-Galerkin case.

### 4.3.2 Test 2: Two-dimensional slab varying restitution properties

To take into account a case in which parameters also affect the ionic model, we focus on the solution of problem (1.3) in a square slab of cardiac tissue $\Omega = (0, 10\text{ cm})^2$, considering $n_\mu = 3$ parameters possibly reflecting intra- and inter-subjects variability. More precisely, the conductivity tensor takes again the form

$$\mathbf{D}(\mathbf{x}; \mu) = \mu_2 I + (\mu_1 - \mu_2)\mathbf{f}_0 \otimes \mathbf{f}_0,$$

where $\mu_1$ and $\mu_2$ consist of the electric conductivities in the longitudinal and the transversal direction to the fibers $\mathbf{f}_0 = (1, 0)^T$, respectively, and $\mu_3$ regulates the APD by defining

$$g(u, w) = \Big(\epsilon_0 + \frac{\mu_3 w}{c_2 + u}\Big)(-w - Ku(u - b - 1)).$$

The parameters belong to the parameter space $\mathcal{P} = 12.9 \cdot [0.06, 0.2]\ \text{cm}^2/\text{ms} \times 12.9 \cdot [0.03, 0.1]\ \text{cm}^2/\text{ms} \times [0.15, 0.25]$ and the applied current is defined as in Test 1. The FOM dimension is equal to $N_h = 64 \times 64 = 4096$.

For the training phase, we uniformly sample $N_t = 1000$ time instances in the interval $(0, T)$, with $T = 400$ ms, and consider $N_{train} = 5 \times 5 \times 5 = 125$ training-parameters, i.e. $\boldsymbol{\mu}_{train} = (12.9 \cdot (0.06 + i0.035), 12.9 \cdot (0.03 + j0.0175), 0.15 + s0.025)$ with $i, j, s = 0, \ldots, 4$. For the testing phase, $N_{test} = 4 \times 4 \times 4 = 64$ testing-parameter instances have been considered, each of them given by $\boldsymbol{\mu}_{test} = (12.9 \cdot (0.0775 + i0.035), 12.9 \cdot (0.0387 + j0.0175), 0.1625 + s0.025)$ with $i, j, s = 0, \ldots, 3$. The maximum number of epochs is $N_{epochs} = 10000$, the batch size is $N_b = 40$ and, regarding the early-stopping criterion, we stop the training if the loss function does not decrease along 500 epochs.

In Figure 4.3 we show the FOM and the DL-ROM approximation, this latter with $n = 4$, at $\tilde{t} = 319.7$ ms for the testing-parameter instances $\boldsymbol{\mu}_{test} = (12.9 \cdot 0.1125\ \text{cm}^2/\text{ms}, 12.9 \cdot 0.0563\ \text{cm}^2/\text{ms}, 0.1875)$ and $\boldsymbol{\mu}_{test} = (12.9 \cdot 0.1475\ \text{cm}^2/\text{ms}, 12.9 \cdot 0.0737\ \text{cm}^2/\text{ms}, 0.2375)$, together with the relative error defined as in (3.14). The DL-ROM technique is able to capture the strong variability of the solution over the parameter space. Indeed, in Figure 4.3 (top) the tissue is almost completely depolarized whereas in Figure 4.3 (bottom) repolarization has already started. The error indicator, computed as in (3.13) over these $N_{test} = 64$ testing-parameter instances, is equal to $5.4 \times 10^{-3}$.

In Figure 4.4 we compare the FOM and the DL-ROM action potentials (APs) at $\mathbf{x} = (9.524, 4.762)$ cm, by considering the effect of the different parameters separately. More precisely, in Figure 4.4 (left) we let $\mu_3$ vary, i.e. we take $\boldsymbol{\mu}^1_{test} = (12.9 \cdot 0.1125\ \text{cm}^2/\text{ms}, 12.9 \cdot 0.0737\ \text{cm}^2/\text{ms}, 0.1625)$ and $\boldsymbol{\mu}^2_{test} = (12.9 \cdot 0.1125\ \text{cm}^2/\text{ms}, 12.9 \cdot 0.0737\ \text{cm}^2/\text{ms}, 0.2375)$. In Figure 4.4 (right) instead we only vary $\mu_1$ and $\mu_2$, i.e. we take $\boldsymbol{\mu}^1_{test} = (12.9 \cdot 0.0775\ \text{cm}^2/\text{ms}, 12.9 \cdot 0.0387\ \text{cm}^2/\text{ms}, 0.2125)$ and $\boldsymbol{\mu}^2_{test} = (12.9 \cdot 0.1825\ \text{cm}^2/\text{ms}, 12.9 \cdot 0.0912\ \text{cm}^2/\text{ms}, 0.2125)$. In both cases, the DL-ROM correctly reproduces the APD variability and the different depolarization patterns.

Finally, we report in Table 4.2 the training and testing computational times of the DL-ROM, on a GTX 1070 8 GB GPU, by considering either $n_\mu = 2$ or 3 parameters:

- $n_\mu = 2$, $N_{train} = 5 \times 5 = 25$, $N_t = 1000$, with $\mathcal{P} = 12.9 \cdot [0.06, 0.2]\ \text{cm}^2/\text{ms} \times 12.9 \cdot [0.03, 0.1]\ \text{cm}^2/\text{ms}$,

- $n_\mu = 3$, $N_{train} = 5 \times 5 \times 5 = 125$, $N_t = 1000$, with $\mathcal{P} = 12.9 \cdot [0.06, 0.2]\ \text{cm}^2/\text{ms} \times 12.9 \cdot [0.03, 0.1]\ \text{cm}^2/\text{ms} \times [0.15, 0.25]$,

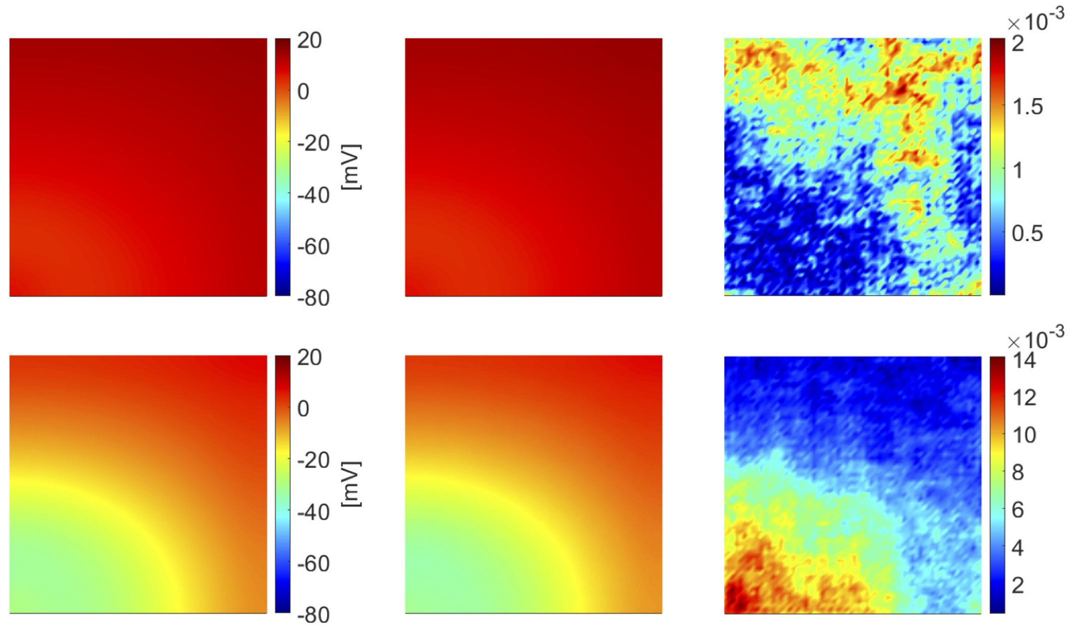


Figure 4.3: *Test 2*: FOM solution (left), DL-ROM solution with $n = 4$ (center) and relative error $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instances $\boldsymbol{\mu}_{test} = (12.9 \cdot 0.1125\ \text{cm}^2/\text{ms}, 12.9 \cdot 0.0563\ \text{cm}^2/\text{ms}, 0.1875)$ (top) and $\boldsymbol{\mu}_{test} = (12.9 \cdot 0.1475\ \text{cm}^2/\text{ms}, 12.9 \cdot 0.0737\ \text{cm}^2/\text{ms}, 0.2375)$ (bottom) at $\tilde{t} = 319.7$ ms. The maximum of the relative error $\boldsymbol{\epsilon}_k$ is about $10^{-3}$.

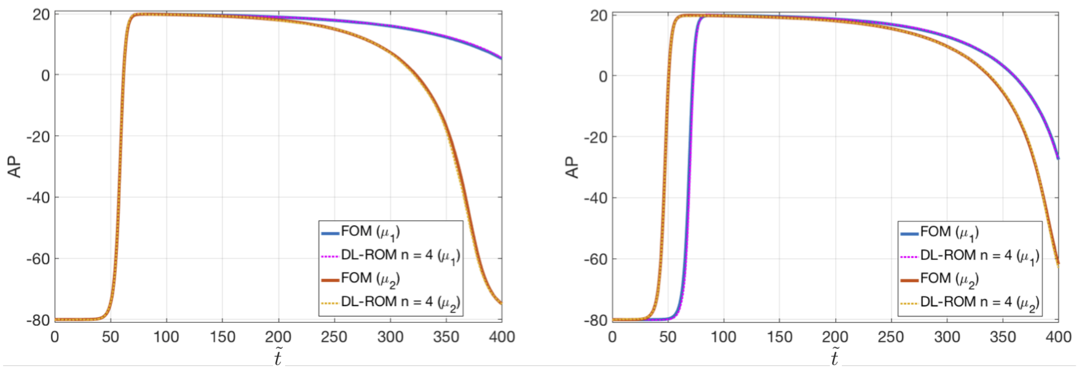


Figure 4.4: *Test 2*: APs obtained through the FOM and the DL-ROM with $n = 4$. Left: $\boldsymbol{\mu}^1_{test} = (12.9 \cdot 0.1125\ \text{cm}^2/\text{ms}, 12.9 \cdot 0.0737\ \text{cm}^2/\text{ms}, 0.1625)$ and $\boldsymbol{\mu}^2_{test} = (12.9 \cdot 0.1125\ \text{cm}^2/\text{ms}, 12.9 \cdot 0.0737\ \text{cm}^2/\text{ms}, 0.2375)$. Right: $\boldsymbol{\mu}^1_{test} = (12.9 \cdot 0.0775\ \text{cm}^2/\text{ms}, 12.9 \cdot 0.0387\ \text{cm}^2/\text{ms}, 0.2125)$ and $\boldsymbol{\mu}^2_{test} = (12.9 \cdot 0.1825\ \text{cm}^2/\text{ms}, 12.9 \cdot 0.0912\ \text{cm}^2/\text{ms}, 0.2125)$. The DL-ROM approximation accurately reproduces the APD variability and the different depolarization patterns.

to analyze the effect of introducing an additional parameter on the training time for a prescribed level of accuracy, keeping the architecture of the network fixed. The training time refers to the overall training and validation time, while the testing one refers to the time needed by the DL-ROM to compute $N_t$ time instances of the solution for a given testing-parameter instance. In this case, considering $N_{train} = 125$ training-parameter instances allows us to reduce the training computational time of a factor 3, even if more parameters are considered, a larger training set is provided. However, we highlight that stating general conclusions about the training complexity and costs, as a function of the number of parameters and the training set dimensions, is far from being trivial, and still represents an open issue in this framework.

| $n_\mu$ | $N_{train}$ | train time | $n_{epochs}$ | test time |
|---|---|---|---|---|
| 2 | 25 | 15 h | 6981 | 0.08 s |
| 3 | 125 | 5 h | 449 | 0.08 s |

Table 4.2: *Test 2*: Number of parameters, training-parameter instances and epochs together with training and testing computational times in the two configurations.

### 4.3.3 Test 3: Two-dimensional slab with ischemic region

We consider the computation of the transmembrane potential in a square slab $\Omega = (0, 10 \text{ cm})^2$ of cardiac tissue in presence of an ischemic (non-conductive) region. The ischemic region may act as anatomical driver of cardiac arrhythmias like tachycardias and fibrillations. The system we want to solve is a slight modification of equations (1.3), accounting for the presence of a non-conductive region which affects both the conductivity tensor and the ionic current term. The ischemic portion of the domain is modeled by replacing the conductivity tensor $\mathbf{D}(\mathbf{x})$, defined in (1.4), with $\bar{\mathbf{D}}(\mathbf{x};\boldsymbol{\mu}) = \sigma(\mathbf{x},\boldsymbol{\mu})\mathbf{D}(\mathbf{x})$, where the function $\sigma(\mathbf{x},\boldsymbol{\mu})$ is given by

$$\begin{aligned} \sigma(\mathbf{x};\boldsymbol{\mu}) &= \rho(\mathbf{x};\boldsymbol{\mu}) + \sigma_0(1-\rho(\mathbf{x};\boldsymbol{\mu})), \\ \rho(\mathbf{x};\boldsymbol{\mu}) &= 1 - \exp\left(-\frac{(x_1-\mu_1)^4 + (x_2-\mu_2)^4}{2\alpha^2}\right). \end{aligned} \tag{4.5}$$

In this case, $n_\mu = 2$ parameters are considered, representing the coordinates of the center of the scar, belonging to the parameter space $\mathcal{P} = [3.5, 6.5 \text{ cm}]^2$. Moreover, $\alpha = 7 \text{ cm}^2$, $\sigma_0 = 10^{-4}$, the transversal and longitudinal conductivities are $\sigma_t = 12.9 \cdot 0.1 \text{ cm}^2/\text{ms}$ and $\sigma_l = 12.9 \cdot 0.2 \text{ cm}^2/\text{ms}$, respectively, and $\mathbf{f}_0 = (1,0)^T$, meaning that the tissue fibers are parallel to the $x-$axis. Similarly, the ionic current $I_{ion}(u,w)$ in (1.3) is replaced by $\bar{I}_{ion}(u,w;\boldsymbol{\mu}) = \rho(\mathbf{x};\boldsymbol{\mu})I_{ion}(u,w)$. The applied current takes the form

$$I_{app}(\mathbf{x},t) = C\exp\left(-\frac{||\mathbf{x}||^2}{\beta}\right)\mathbf{1}_{[0,\bar{t}]}(\tilde{t}),$$

where $C = 100$ mA, $\beta = 0.02 \text{ cm}^2$ and $\bar{t} = 2$ ms, consisting in a Gaussian-shaped applied stimulus with support in a circle with radius almost equal to 3 cm. The parameters appearing in (1.5) are set to $K = 8$, $a = 0.01$, $b = 0.15$, $\varepsilon_0 = 0.002$, $c_1 = 0.2$, and $c_2 = 0.3$, see [Göktepe et al., 2010]. The equations have been discretized in space by considering $N_h = 64\times 64 = 4096$. For the time discretization, we consider a time-step $\Delta t = 0.1/12.9$ over $(0,T)$ with $T = 400$ ms.

For the training phase, we uniformly sample $N_t = 1000$ time instances over $(0,T)$ and consider $N_{train} = 49$ training-parameter instances, with $\boldsymbol{\mu}_{train} = (3.5 + i0.5, 3.5 + j0.5)$, $i,j = 0,\dots,6$. The maximum number of epochs is set equal to $N_{epochs} = 10000$, the batch size is $N_b = 40$ and, regarding the early-stopping criterion, we stop the training if the loss function does not decrease in 500 epochs. For the testing phase, $N_{test} = 36$ testing-parameter instances $\boldsymbol{\mu}_{test} = (3.75 + i0.5, 3.75 + j0.5)$, $i,j = 0,\dots,5$, have been considered.

In Figures 4.5 we show the FOM and the DL-ROM solutions, the latter obtained with $n = 3$ for the testing-parameter instance $\boldsymbol{\mu}_{test} = (6.25, 6.25)$ cm at $\tilde{t} = 100$ and 356 ms, respectively, together with the relative error $\boldsymbol{\epsilon}_k \in \mathbb{R}^{N_h}$ defined in (3.14). In Figure 4.5 (top) the tissue is depolarized except for the region occupied by the scar and surrounding it, which is clearly characterized by a slower conduction. In Figure 4.5 (bottom) the tissue is starting to repolarize and even if the shape of the ischemic region is not sharply reproduced, the DL-ROM solution is able to capture the diseased (non-conductive) nature of this portion of tissue.

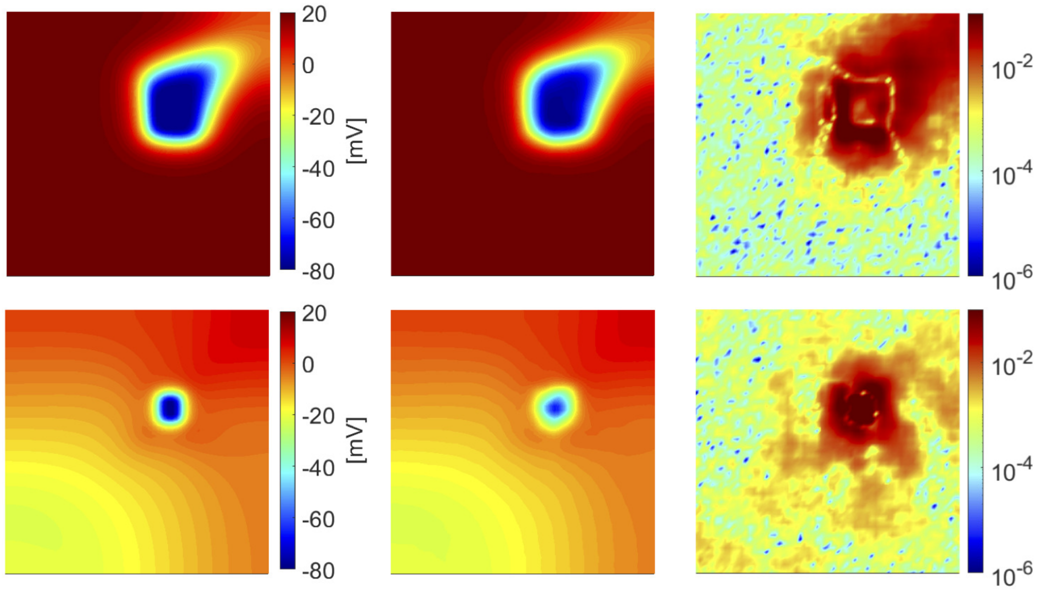


Figure 4.5: *Test 3*: FOM solution (left), DL-ROM solution with $n = 3$ (center) and relative error $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instance $\boldsymbol{\mu}_{test} = (6.25, 6.25)$ cm at $\tilde{t} = 100$ ms (top) and $\tilde{t} = 356$ ms (bottom). The maximum of the relative error $\boldsymbol{\epsilon}_k$ is $10^{-2}$ and it is associated to the diseased tissue.

In Figure 4.6 we compare the APs computed at six selected points $P_1, \dots, P_6$. The DL-ROM is able to provide an accurate reconstruction of the AP at almost all points; the maximum error is associated to the point $P_3$, the closest one to the center of the scar, for $\tilde{t} \geq 200$ ms. However, even in this case, the DL-ROM technique is able to capture the difference, in terms of AP peak values, between the diseased and the healthy tissue.

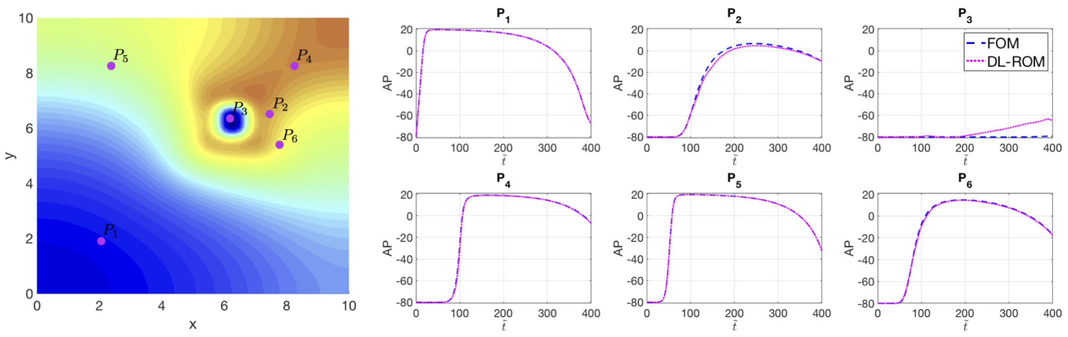


Figure 4.6: *Test 3*: Left: FOM solution evaluated for $\boldsymbol{\mu}_{test} = (6.25, 6.25)$ cm at $\tilde{t} = 400$ ms together with the points $P_1, \dots, P_6$. Right: APs evaluated for $\boldsymbol{\mu}_{test} = (6.25, 6.25)$ cm at points $P_1, \dots, P_6$. The DL-ROM, with $n = 3$, is able to sharply reconstruct the AP in almost all the points and the main features are captured also for the points close to the scar.

The AP variability across the parameter space characterizing both the FOM and the DL-ROM solutions is shown in Figure 4.7. Still with a DL-ROM dimension $n = 3$, we report the

APs for $\boldsymbol{\mu}_{test} = (\mu_{test}, \mu_{test})$ cm, with $\mu_{test} = 3.75, 4.25, 4.75, 5.25, 5.75,\ 6.25$, evaluated at $P = (7.46, 6.51)$ cm. The DL-ROM is able to capture such variability over $\mathcal{P}$; moreover, the larger $\boldsymbol{\mu}_{test}$, the smaller the distance between the point $P$ and the scar, with their proximity impacting on the shape and the values of the AP. In particular, for $\mu_{test} = 6.25$, the point $P$ falls into the *grey zone*.

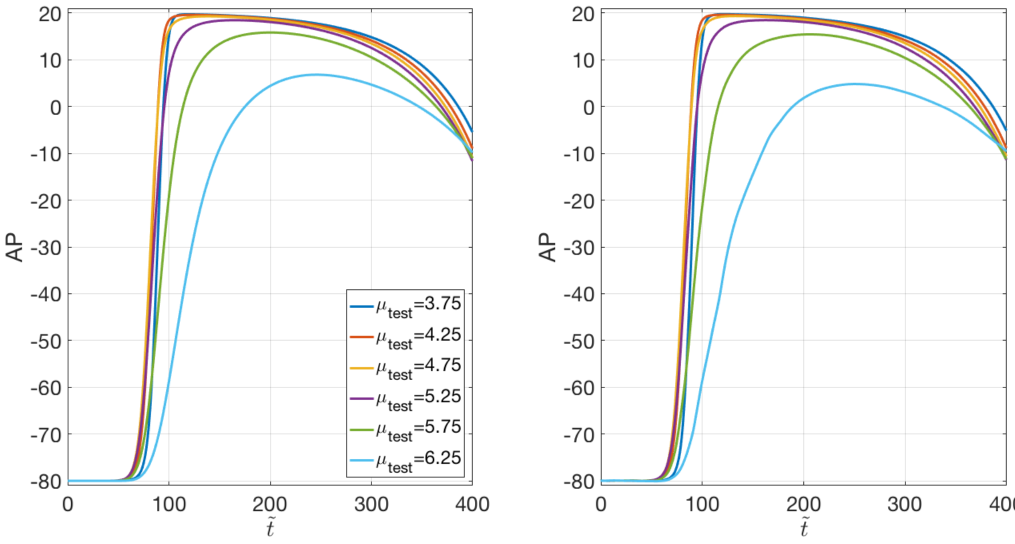


Figure 4.7: *Test 3*: FOM (right) and DL-ROM (left) AP variability over $\mathcal{P}$ at $P = (7.46, 6.51)$ cm. The DL-ROM sharply reconstructs the FOM variability over $\mathcal{P}$.

By using the DL-ROM technique and setting the dimension of the nonlinear trial manifold equal to the dimension of the solution manifold, i.e. $n = 3$, we obtain an error indicator (3.13) of $\epsilon_{rel} = 2.01 \times 10^{-2}$. In order to assess the computational efficiency of DL-ROM, we compare it with the POD-Galerkin ROM relying on $N_c$ local reduced bases; we report in Table 4.3 the maximum and minimum number of basis functions, among all the clusters, required by the POD-Galerkin ROM [Quarteroni et al., 2016, Pagani et al., 2018] to achieve the same accuracy.

| $N_c = 1$ | $N_c = 2$ | $N_c = 4$ | $N_c = 6$ |
|---|---|---|---|
| 250 | 219 | 200 | 193 |
|  | 107 | 35 | 26 |

Table 4.3: *Test 3*: Maximum and minimum dimensions of the local reduced bases (that is, linear trial manifolds) built by the POD-Galerkin ROM for different numbers $N_c$ of clusters.

In Figure 4.8 (left) we compare the CPU time required to solve the FOM (through linear finite elements) over the time interval $(0, T)$, with the one needed by DL-ROM with $n = 3$, and the POD-Galerkin ROM with $N_c = 6$ local reduced bases, at testing time, by varying the FOM dimension $N_h$. Here, with testing time we refer, both for the DL-ROM and the POD-Galerkin ROM, to the time needed to query the ROM over the whole interval $(0, T)$, by using for each technique the proper time resolution, for a given testing-parameter instance. Since the DL-ROM solution can be queried at a given time without requiring any solution of a dynamical system to recover the former time instances, the DL-ROM can employ larger time windows compared to the time-steps required by the solution of the FOM and POD-Galerkin ROM dynamical systems for the cases at hand. This fact also has a positive impact on the data used during the training phase[4]. The speed-up obtained, for each value of $N_h$ considered, is reported in Table 4.4. Both the DL-ROM and the POD-Galerkin ROM allow us to decrease the computational costs associated to the computation of the FOM solution for a testing-parameter instance. However, for a desired level of accuracy, CPU times required by the POD-Galerkin ROM during the testing phase are remarkably higher than the ones required by a DL-ROM with $n = 3$.

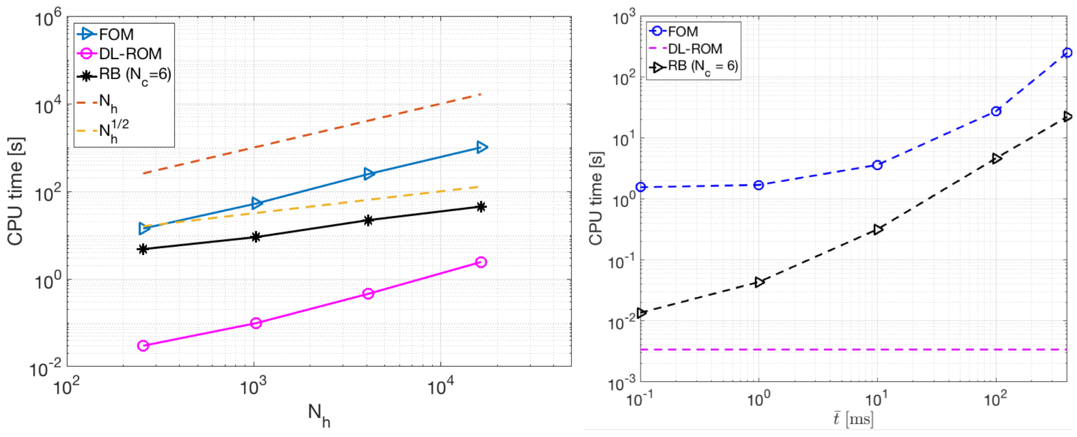


Figure 4.8: *Test 3*: Left: CPU time required to solve the FOM, by DL-ROM at testing time with $n = 3$ and by the POD-Galerkin ROM at testing time with $N_c = 6$ vs. $N_h$. The DL-ROM provides the smallest testing computational time for each $N_h$ considered. Right: FOM, POD-Galerkin ROM and DL-ROM CPU computational times to compute $\tilde{\mathbf{u}}_h(\bar{t};\boldsymbol{\mu}_{test})$ vs. $\bar{t}$ averaged over the testing set. Thanks to the fact that the DL-ROM can be queried at any time istance it is extremely efficient in computing $\tilde{\mathbf{u}}_h(\bar{t};\boldsymbol{\mu}_{test})$ with respect to both the FOM and the POD-Galerkin ROM.

| | $N_h = 256$ | $N_h = 1024$ | $N_h = 4096$ | $N_h = 16384$ |
|---|---|---|---|---|
| FOM vs. DL-ROM | 472 | 536 | 539 | 412 |
| FOM vs. POD-Galerkin ROM | 3 | 6 | 12 | 22 |

Table 4.4: *Test 3*: DL-ROM and POD-Galerkin ROM vs. FOM speed-up by varying $N_h$. The DL-ROM speed-up is remarkably higher than the one obtained by using the POD-Galerkin ROM.

Both the DL-ROM and the POD-Galerkin ROM depend on the FOM dimension $N_h$. In the case of DL-ROM, the dependency on $N_h$ at testing time, for a fixed value of $\Delta t$, is due to the presence of the decoder function; indeed, the process of learning the reduced dynamics (and so the dimension of the nonlinear trial manifold) does not depend on the FOM dimension. On the other hand, the dependence of the POD-Galerkin ROM on the FOM dimension also impacts on the dimension of the local linear trial manifolds: in general, by increasing $N_h$ the dimension of each local linear subspace also increases. Referring to Figure 4.8 (left) and Table 4.4, the CPU time required by the DL-ROM at testing time scales linearly with $N_h$, instead the one required by the POD-Galerkin ROM scales linearly with $\sqrt{N_h}$. In particular, even for the larger FOM dimension considered ($N_h = 16384$ for this test case), our DL-ROM is 19 times faster than the POD-Galerkin ROM. We are not able to run simulations for $N_h > 16384$, because of the limitation of the computing resources we

[4]Indeed, in order to build the snapshot matrix, we uniformly sample $N_t$ time instances of the FOM solution over $T/\Delta t = 4000$ time steps; for each training parameter instance, only 25% of 4000 snapshots are retained from the FOM solution in the DL-ROM case, against 4000 snapshots in the POD-Galerkin ROM case.

have at our disposal. Despite the trend in Figure 4.8 (left) is apparently not favorable for the DL-ROM technique, practice indicates that the CPU time for DL-ROM is smaller than the one for the POD-Galerkin ROM for *small* values of $N_h$, in other words only with very large values of $N_h$ the POD-Galerkin ROM outperforms the DL-ROM strategy. Indeed, a linear fitting of the DL-ROM and the POD-Galerkin ROM CPU times[5] in Figure 4.8 (left) highlights that for $N_h = 65536$ and $N_h = 262144$, DL-ROM could be almost 10 and 5 times, respectively, faster than the POD-Galerkin ROM for the same values of $N_h$. Note that the results of this Section have been obtained by employing the DL-ROM on a single CPU, an architecture which is not favorable to neural networks[6]. Further improvements are expected when employing our DL-ROM on a GPU for a given testing-parameter instance.

We highlight that since the DL-ROM solution can be evaluated at any desired time instance without solving any dynamical system, the resulting computational time entailed by the DL-ROM at testing time are drastically reduced compared to the ones required by the FOM or the POD-Galerkin ROM to compute solutions at a particular time instance. In Figure 4.8 (right) we show the DL-ROM, FOM and POD-Galerkin ROM CPU time needed to compute the approximated solution at $\bar{t}$, for $\bar{t} = 1$, 10, 100 and 400 ms averaged over the testing set and with $N_h = 4096$. We perform the training phase of the POD-Galerkin ROM over the original time interval $(0, T)$ ms and we report the results for $N_c = 6$, the number of clusters for which the smallest computational time is obtained. The DL-ROM CPU time to compute $\tilde{\mathbf{u}}_h(\bar{t}; \boldsymbol{\mu}_{test})$ does not vary over $\bar{t}$ and, by choosing $\bar{t} = T$, the DL-ROM speed-ups are equal to $7.3 \times 10^4$ and $6.5 \times 10^3$ with respect to the FOM and the POD-Galerkin ROM, with $N_c = 6$, computational times.

Regarding the training (offline) times, in the case of a FOM dimension $N_h = 4096$, training the DL-ROM neural network on a GTX 1070 8GB GPU requires about 21 hours, whereas training the POD-Galerkin ROM (with $N_c = 6$ local bases) on 5 cores of a node of the HPC cluster at our disposal requires about 3 hours; in both the cases, the time needed to assemble the snapshot matrix $\mathbf{S}$ is not included. However, the 7 times higher training time of the DL-ROM is justified by the efficiency gained at testing time; indeed, a query to the DL-ROM online requires 0.08 seconds on a GPU, implying a speed-up of about 275 times compared to the POD-Galerkin ROM.

Finally we report the feature maps of the DL-ROM neural network. In Figures 4.9 we show the feature maps of the first convolutional layer of the encoder function $\sigma_1(W_1^k * \mathbf{u}^1(\boldsymbol{\mu_{test}}) + b_1^k)$, for $k = 1, \ldots, 8$, in the DL-ROM neural network when the FOM solutions for the testing-parameter instances $\boldsymbol{\mu}_{test} = (3.75, 3.75)$ cm and $\boldsymbol{\mu}_{test} = (6.25, 6.25)$ cm at $t = 0.2$ ms, are provided as inputs. At this stage, the feature maps retain most of the information present in the FOM solution. Moreover, by considering the two testing-parameter instances, we observe the translation equivariance property [Goodfellow et al., 2016] that convolutional layers hold when applied to the part of cardiac tissue corresponding to the scar. Moving to deeper layers, feature maps become increasingly abstract, and less visually interpretable; however, the extracted high-level features are still related both to the ischemic region and the electrical activation pattern.

### 4.3.4 Test 4: Two-dimensional slab with figure of eight re-entry

The most recognized cellular mechanisms sustaining atrial tachycardia is re-entry [Nattel, 2002]. The particular kind of re-entry we deal with in this test case is the so-called figure of eight re-entry, and can be obtained by solving equations (1.3). To induce the re-entry,

[5] $N_h = 65536$ and $N_h = 262144$ for this test case represent FOM dimensions corresponding to mesh sizes $h$ needed to solve, by means of linear finite elements, the problem on a 3D slab geometry both for physiological and pathological electrophysiology in the case a ten Tusscher-Panfilov ionic model [ten Tusscher and Panfilov, 2006] is used. This latter would indeed require smaller values of $h$ compared to the Aliev-Panfilov model, due to the shape of the AP. See, e.g., [Trayanova, 2011, Plank et al., 2008] for further details.

[6] Indeed, all tests are performed on a node (20 Intel® Xeon® E5-2640 v4 2.4GHz cores), using 5 cores, of our in-house HPC cluster.

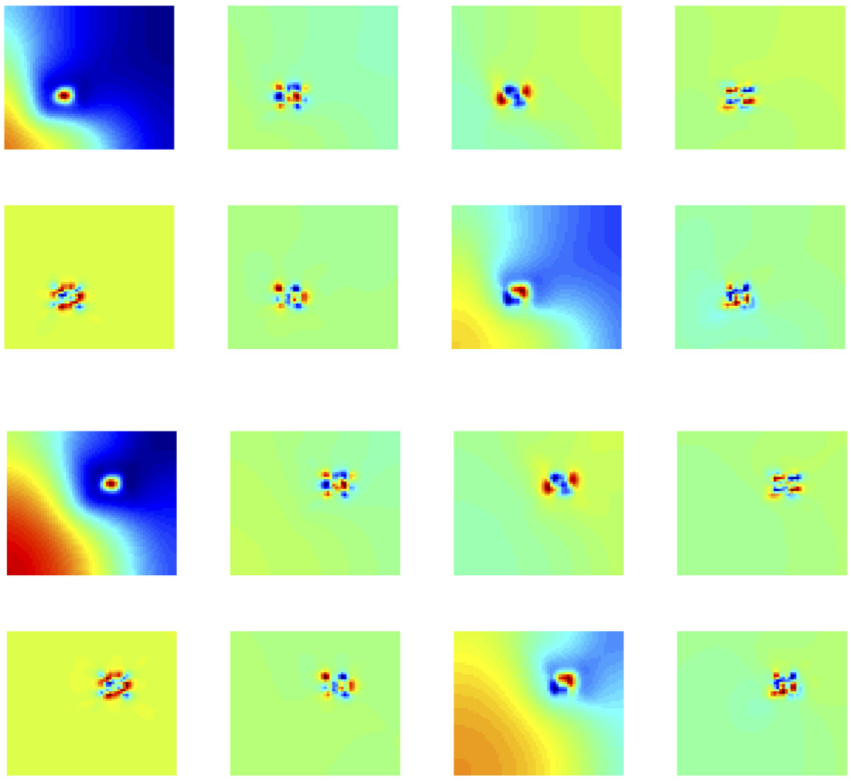


Figure 4.9: *Test 3*: Feature maps of the first convolutional layer of the encoder function in the DL-ROM neural network for the testing-parameter instances $\boldsymbol{\mu}_{test} = (3.75, 3.75)$ cm (top) and $\boldsymbol{\mu}_{test} = (6.25, 6.25)$ cm (bottom) at $\tilde{t} = 0.2$ ms.

we apply a classical S1-S2 protocol [Nagaiah et al., 2013, Colli Franzone et al., 2014]. In particular, we consider a square slab of cardiac tissue $\Omega = (0, 2 \text{ cm})^2$ and apply an initial stimulus (S1) at the bottom edge of the domain, i.e.

$$I^1_{app}(\mathbf{x}, t) = \mathbf{1}_{\Omega_1}(\mathbf{x})\mathbf{1}_{[t^i_1, t^f_1]}(\tilde{t}), \tag{4.6}$$

where $\Omega_1 = \{\mathbf{x} \in \Omega : y \leq 0.1\}$, $t^i_1 = 0$ ms and $t^f_1 = 5$ ms. A second stimulus (S2) under the form

$$I^2_{app}(\mathbf{x}, t; \mu) = \mathbf{1}_{\Omega_2(\mu)}(\mathbf{x})\mathbf{1}_{[t^i_2, t^f_2]}(\tilde{t}), \tag{4.7}$$

with $\Omega_2(\mu) = \{\mathbf{x} \in \Omega : (x-1)^2 + (y-\mu)^2 \leq (0.2)^2\}$, $t^i_2 = 70$ ms and $t^f_2 = 75$ ms, is then applied. Here the parameter $\mu$ is the $y$-coordinate of the center of the second circular stimulus. We analyze two configurations: *(i)* a first case in which both re-entry and non re-entry cases are generated, by considering $\mathcal{P} = [0.5, 1.1]$ cm; *(ii)* a second case in which instead only re-entrant dynamics are taken into account, and $\mathcal{P} = [0.8, 1.1]$ cm. These choices have been made to obtain a re-entry elicited and sustained until $T = 175$ ms. Moreover, we restrict ourselves to the time interval [95, 175] ms, without considering the time window [0, 95) ms in which the re-entry has not arisen yet, and is common to all $\mu$ instances. The time-step is $\Delta t = 0.2/12.9$. We consider a FOM dimension $N_h = 256 \times 256 = 65536$, implying a mesh size $h = 0.0784$ mm; this mesh size is recognized to correctly solve the tiny transition front developing during depolarization of the tissue, see [Trayanova, 2011, Plank et al., 2008]. The fibers are parallel to the $x$-axis and the conductivities in the longitudinal and transversal directions to the fibers are $\sigma_l = 2 \times 10^{-3}$ cm$^2$/ms and $\sigma_t = 3.1 \times 10^{-4}$ cm$^2$/ms, respectively.

The parameters appearing in (1.5) are set to $K = 8$, $a = 0.1$, $b = 0.1$, $\varepsilon_0 = 0.01$, $c_1 = 0.14$, and $c_2 = 0.3$, see [ten Tusscher, 2004].

The snapshot matrix is built by solving problem (1.3), completed with the applied currents (4.6) and (4.7) over $N_t = 400$ time instances. Moreover, we consider $N_{train} = 13$ training-parameter instances uniformly distributed in the parameter space and $N_{test} = 12$ testing-parameter instances, each of them corresponding to the midpoint of two consecutive training-parameter instances. The maximum number of epochs is set equal to $N_{epochs} = 6000$, the batch size is $N_b = 3$, due to the high GPU memory occupation of each sample. Regarding the early-stopping criterion, we stop the training if the loss does not decrease in 1000 epochs.

In Figure 4.10 we show the FOM solution and the DL-ROM one, obtained by setting the reduced dimension to $n = 5$, for the testing-parameter instance $\mu_{test} = 0.9625$ cm, at $\tilde{t} = 141.2$ ms and $\tilde{t} = 157.2$ ms, together with the relative error $\boldsymbol{\epsilon}_k \in \mathbb{R}^{N_h}$ computed according to (3.14).

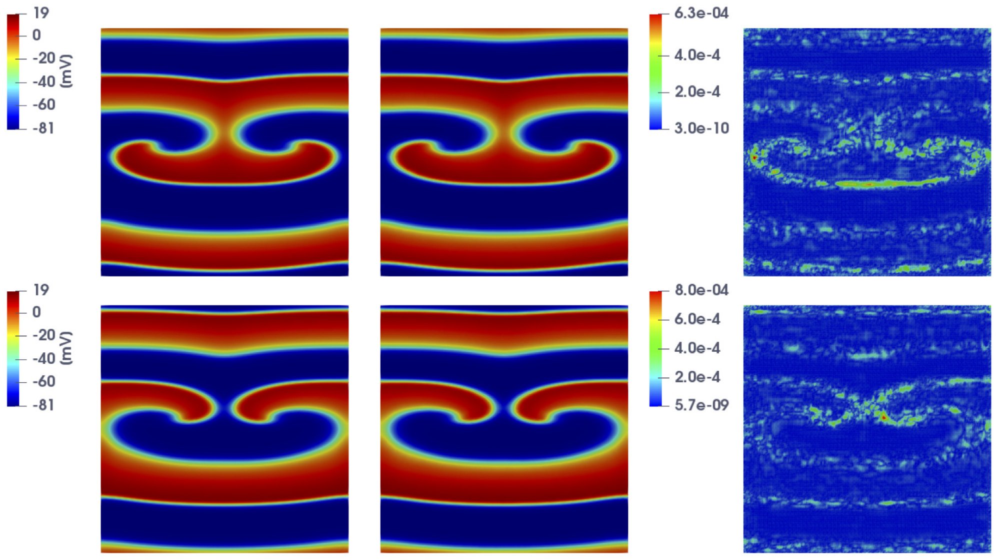


Figure 4.10: *Test 4*: FOM solution (left), DL-ROM one (center) with $n = 5$, and relative error $\boldsymbol{\epsilon}_k$ (right) at $\tilde{t} = 141.2$ ms (top) and $\tilde{t} = 157.2$ ms (bottom), for the testing-parameter instance $\mu_{test} = 0.9625$ cm. The relative error $\boldsymbol{\epsilon}_k$ is below 0.1% at both time instants.

We introduce the relative error $\boldsymbol{\epsilon}_k^s \in \mathbb{R}^{N_h}$, for $k = 1, \dots, N_t$, given by

$$\boldsymbol{\epsilon}_k^s = \frac{|\mathbf{u}^k(\mu_{test}) - \tilde{\mathbf{u}}^k(\mu_{test})|}{\|\mathbf{u}^k(\mu_{test})\|_1} \times 100. \tag{4.8}$$

The trend of (4.8) over time, for the selected testing-parameter instance $\mu_{test} = 0.9625$ cm, is depicted in Figure 4.11; we highlight that the error is, on average, always smaller than 0.3%. In particular, in Figure 4.11 we show the mean, the median, and the first and third quartile (all computed with respect to the spatial coordinates) of the relative error, as well as its minimum. The interquartile range (IQR) shows that the distribution of the error is almost uniform over time, and that the maximum error is associated to the first time instant – this latter being the time instant at which the solution is most different over $\mathcal{P}$.

In Figure 4.12 we show the FOM and the DL-ROM solutions, the latter obtained by setting $n = 5$, for the last time instance, i.e. at $\tilde{t} = 175$ ms, for $\mu_{test} = 0.6125$ cm and $\mu_{test} = 0.9125$ cm, in order to point out the variability of the solution over the parameter space $\mathcal{P} = [0.5, 1.1]$ cm and the ability of DL-ROM to capture it. In particular, in Figure 4.12 we compare the FOM and the DL-ROM solutions for two testing-parameter instances corresponding to *(i)* a

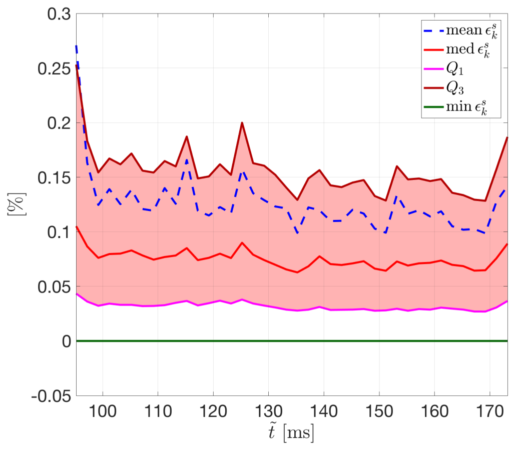


Figure 4.11: *Test 4*: Relative error $\epsilon_k^s$ vs. $\tilde{t}$ with $n = 5$ for the testing-parameter instance $\mu_{test} = 0.9625$ cm (the red band indicates the IQR). The error distribution is almost uniform over time.

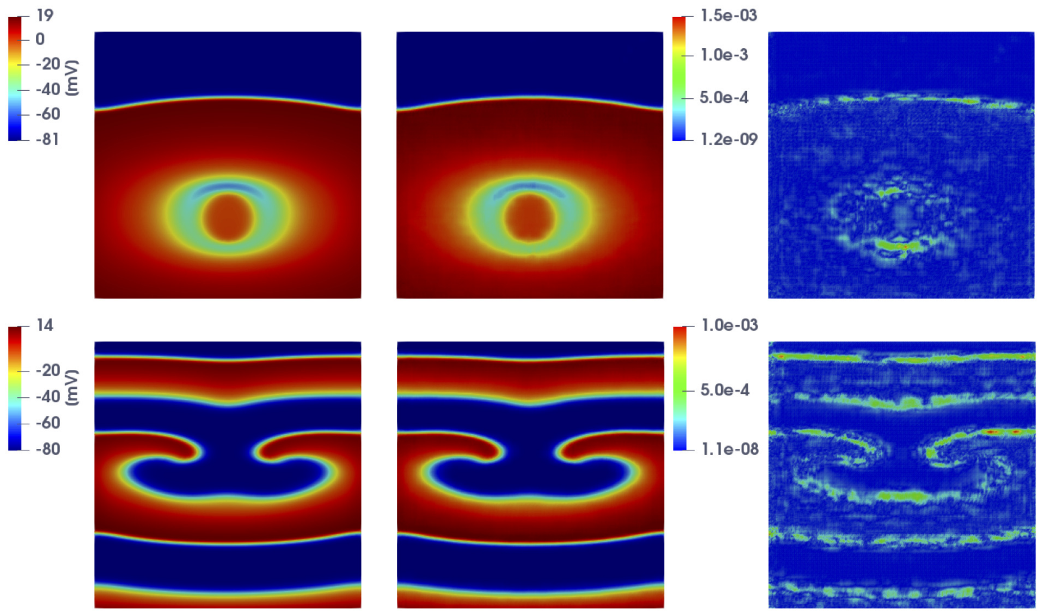


Figure 4.12: *Test 4*: FOM solution (left), DL-ROM one (center), with $n = 5$, and relative error $\epsilon_k$ (right) at $\tilde{t} = 175$ ms, for the testing-parameter instance $\mu_{test} = 0.6125$ cm (top) and $\mu_{test} = 0.9125$ cm (bottom). In both cases the relative error $\epsilon_k$ is below 1%.

case in which the re-entry does not arise (top), since S2 is too far from the front elicited with S1, i.e. the tissue around S2 is no longer in the refractory period and is able to activate again; *(ii)* a case in which the re-entry is elicited, the electrical signal follows an alternative circuit looping back upon itself and developing a self-perpetuating rapid and abnormal activation (bottom).

In Figure 4.13 we show the trend of the relative error (4.8) at a selected time instance given by $\tilde{t} = 147$ ms over the parameter space, reporting the mean, the median, the first and third quartile, as well as its minimum (all computed with respect to the spatial coordinates). We highlight that the error is always smaller than 1%, except for its maximum which is

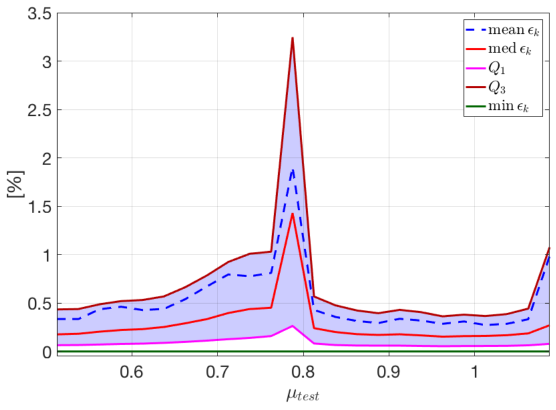


Figure 4.13: *Test 4*: Relative error $\epsilon_k^s$ vs. $\mu_{test}$ with $n = 5$ for the time instance $\tilde{t} = 147$ ms (the violet band indicates the IQR). The maximum error is associated to $\mu_{test} = 0.7875$ cm, the testing-parameter instance between $\mu_{train} = 0.775$ cm (the last value for which re-entry does not arise) and $\mu_{train} = 0.8$ cm (the first value for which re-entry is elicited).

associated to the value of $\mu_{test}$ corresponding to the transition between re-entry and non re-entry dynamics.

Let us now focus on the case in which only re-entrant dynamics are generated, and $\mathcal{P} = [0.8, 1.1]$ cm, in order to compare the FOM, the POD-Galerkin ROM and the DL-ROM approximations. In Figure 4.14 we show the solutions obtained through the POD-Galerkin ROM with $N_c = 2$ (top) and $N_c = 4$ (bottom) local reduced bases, along with the relative error defined in (3.14), for the testing-parameter instance $\mu_{test} = 0.9625$ cm at $\tilde{t} = 175$ ms. In both cases, we have considered the setting yielding the most efficient POD-Galerkin ROM approximation, which require about 40 (30, respectively) seconds to be evaluated. By comparing Figure 4.14 and Figure 4.10 (bottom), we observe that the DL-ROM outperforms the POD-Galerkin ROM in terms of accuracy.

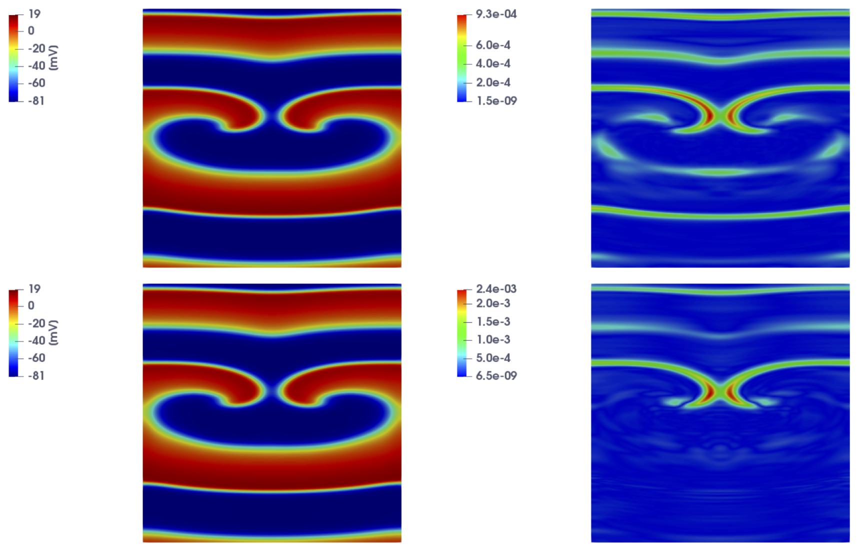


Figure 4.14: *Test 4*: POD-Galerkin ROM solution (left) and relative error $\epsilon_k$ (right) for $N_c = 2$ (top) and $N_c = 4$ (bottom) at $\tilde{t} = 157.2$ ms for $\mu_{test} = 0.9625$ cm.

In Figure 4.15 we show the APs obtained through the FOM, the DL-ROM and the POD-Galerkin ROM (with $N_c = 4$ local reduced bases) for the testing-parameter instance $\mu_{test} = 0.9625$ cm, evaluated at $P_1 = (0.64, 1.11)$ cm and $P_2 = (0.69, 1.03)$ cm. These two points are close to the left core of the figure of eight re-entry, where a shorter APD, and lower peak values of AP, with respect to a healthy case, due to the meandering of the cores, are observed. The AP dynamics at those points is accurately captured by the DL-ROM, while the POD-Galerkin ROM leads to slightly less accurate results requiring larger testing times.

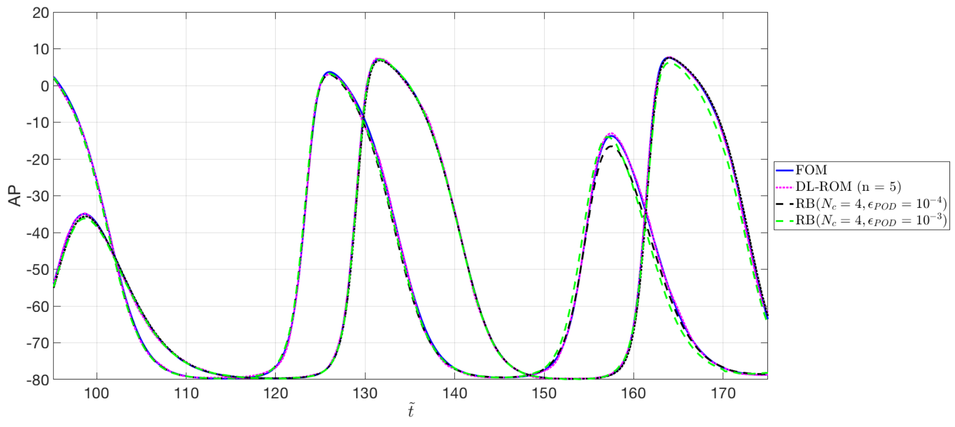


Figure 4.15: *Test 4*: AP obtained through the FOM, the DL-ROM and the POD-Galerkin ROM with $N_c = 4$ for the testing-parameter instance $\mu_{test} = 0.9625$ cm, at $P_1 = (0.64, 1.11)$ cm and $P_2 = (0.69, 1.03)$ cm. The POD-Galerkin ROM approximations are obtained by imposing a POD tolerance $\varepsilon_{POD} = 10^{-4}$ and $10^{-3}$, resulting in error indicator (3.13) values equal to $5.5 \times 10^{-3}$ and $7.6 \times 10^{-3}$, respectively.

We now compare the computational times required by the FOM, the POD-Galerkin ROM (for different values of $N_c$) and the DL-ROM, keeping for all the same degree of accuracy achieved by DL-ROM, i.e. $\epsilon_{rel} = 7.87 \times 10^{-3}$, and running the code on the hardware each implementation is optimized for – a CPU for the FOM and the POD-Galerkin ROM, a GPU[7] for the DL-ROM. In Table 4.5 we report the CPU time needed to compute the FOM solution and the POD-Galerkin ROM approximation (online, at testing phase), both on a full 64 GB node (20 Intel® Xeon® E5-2640 v4 2.4GHz cores), and the GPU time required by the DL-ROM to compute 875 time instances (the same number of time instants considered in the solution of the dynamical systems associated to the FOM and the POD-Galerkin ROM) at testing time, by fixing its dimension to $n = 5$, on an Nvidia GeForce GTX 1070 8 GB GPU. For the sake of completeness, we also report the computational time required by the DL-ROM when employing a single CPU node. It is evident that a POD-Galerkin ROM, built employing a global reduced basis ($N_c = 1$), is not amenable to a complex and challenging pathological cardiac EP problem like the figure of eight re-entry. Using a nonlinear approach, for which the solution manifold is approximated through a piecewise linear trial manifold (as in the case of $N_c = 2$ or $N_c = 4$ local reduced bases) reduces the online computational time. However, the DL-ROM still confirms to provide a more efficient ROM, almost 5 (or 2) times faster on the CPU, and 39 (or 19) faster on the GPU, than the POD-Galerkin ROM with $N_c = 2$ (or $N_c = 4$) local reduced bases.

[7]Indeed, at each layer of a neural network thousands of identical computations must be performed. The most suitable hardware architectures to carry out this kind of operations are GPUs because (i) they have more computational units (cores) and (ii) they have a higher bandwidth to retrieve from memory. Moreover, in applications requiring image processing, as CNNs, the graphics specific capabilities can be further exploited to speed up calculations.

| | time [s] | FOM/ROM dimensions |
|---|---|---|
| FOM (CPU) | 382 | $N_h = 65536$ |
| DL-ROM (CPU/GPU) | 15/1.8 | $n = 5$ |
| POD-Galerkin ROM $N_c = 1$ (CPU) | 103 | $n = 1538$ |
| POD-Galerkin ROM $N_c = 2$ (CPU) | 70 | $n = 1158, 751$ |
| POD-Galerkin ROM $N_c = 4$ (CPU) | 33 | $n = 435, 365, 298, 45$ |

Table 4.5: *Test 4*: POD-Galerkin ROM and DL-ROM computational times along with FOM and reduced trial manifold(s) dimensions. DL-ROM provides a more efficient ROM with respect to the POD-Galerkin ROMs.

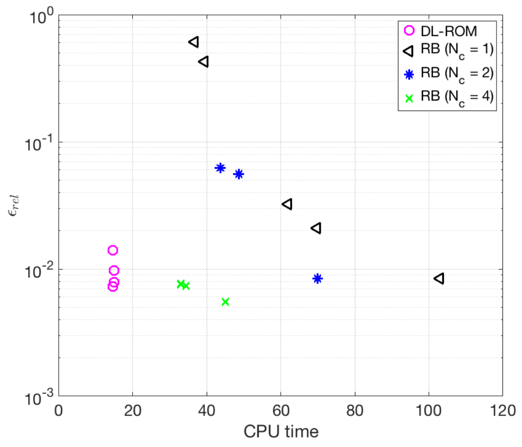


Figure 4.16: *Test 4*: Error indicator $\epsilon_{rel}$ vs. CPU testing computational time for different values of $N_c$ and $\varepsilon_{POD}$. The DL-ROM outperforms the POD-Galerkin ROM in terms of both efficiency and accuracy.

In Figure 4.16 we show the trend of the error indicator (3.13) over the testing set versus the CPU time both for the DL-ROM and the POD-Galerkin ROM at testing phase. Slight improvements of the performance of DL-ROM, in terms of accuracy, are obtained for a small increase of the DL-ROM dimension $n$, coherently with our previous findings in Chapter 3. Indeed, the DL-ROM is able, also in this case, to accurately represent the solution manifold by a reduced nonlinear trial manifold of dimension $n_\mu + 1 = 2$; for the case at hand, we report the results for $n = 5$ (very close to the intrinsic dimension $n_\mu + 1 = 2$ of the problem, and much smaller than the POD-Galerkin ROM dimension), providing slightly smaller values of the error indicator (3.13) than in the case $n = 2$. Regarding instead the POD-Galerkin ROM, in Figure 4.16 we report results obtained for different tolerances $\varepsilon_{POD} = 10^{-4}$, $5 \cdot 10^{-4}$, $10^{-3}$, $5 \cdot 10^{-3}$, $10^{-2}$. In the cases $N_c = 2$ and $N_c = 4$ we only report the results related to the smallest POD tolerances, which indeed allow us to meet the prescribed accuracy on the approximation of the gating variable, which would otherwise impact dramatically on the overall accuracy of the POD-Galerkin ROM. Moreover, we do not consider more than $N_c = 4$ local reduced bases in order not to generate too small local linear subspaces, which would be otherwise unable to approximate the variability of the solution over the parameter space $\mathcal{P}$ accurately. Indeed, by considering a larger number of clusters, the dimension of some linear subspaces becomes so small that the error would start to increase compared to the one obtained with fewer clusters. As shown in Figure 4.16, the proposed DL-ROM outperforms the POD-Galerkin ROM in terms of both efficiency and accuracy.

Regarding the training (offline) times, training the DL-ROM neural network on a GTX 1070 8GB GPU requires about 64 hours, whereas training the POD-Galerkin ROM (with

$N_c = 4$ local bases) on a full node (20 Intel® Xeon® E5-2640 v4 2.4GHz cores) of a HPC cluster requires about 4 hours; in both cases, the time needed to assemble the snapshot matrix $\mathbf{S}$ is not included. The DL-ROM training time is related to the value chosen during the hyperparameters tuning for the batch size, i.e. $N_b = 3$; indeed we highlight that by choosing a slightly higher value of $N_b$, it is possible to decrease the GPU computational training time as long as we look for a lower accuracy. The fact that the DL-ROM training time is 16 times higher than the POD-Galerkin ROM one is again justified by the efficiency introduced at testing time. Indeed, a query to the DL-ROM online requires 1.2 seconds on a GPU, implying a speed-up of about 28 times compared to the POD-Galerkin ROM.

### 4.3.5 Test 5: Three-dimensional ventricle geometry

We consider the solution of the system (1.3) coupled with the A-P ionic model (1.5) in a three-dimensional left ventricle (LV) geometry, obtained from the 3D Human Heart Model provided by Zygote [zyg, 2014]. Here, we consider a single ($n_\mu = 1$) parameter, given by the longitudinal conductivity in the fibers direction. The conductivity tensor takes the form

$$\mathbf{D}(\mathbf{x};\mu) = \sigma_t I + (\mu - \sigma_t)\mathbf{f}_0 \otimes \mathbf{f}_0, \tag{4.9}$$

where $\sigma_t = 12.9 \cdot 0.02$ mm$^2$/ms; $\mathbf{f}_0$ is determined at each mesh point through a rule-based approach, by solving a suitable Laplace problem [Rossi et al., 2014]. The resulting fibers field is reported in Figure 4.17.

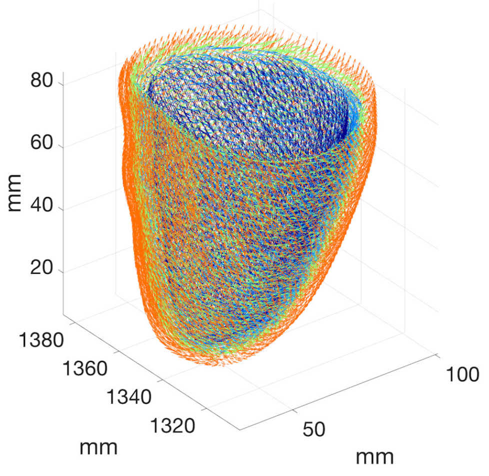


Figure 4.17: *Test 5*: Fibers field on the Zygote LV geometry.

The applied current is defined as

$$I_{app}(\mathbf{x},t) = \frac{C}{(2\pi)^{3/2}\alpha} \exp\left(-\frac{||\mathbf{x}-\bar{\mathbf{x}}||^2}{2\beta}\right)\mathbf{1}_{[0,\bar{t}]}(\tilde{t}),$$

where $\bar{t} = 2$ ms, $C = 1000$ mA, $\alpha = 50$, $\beta = 50$ mm$^2$, $\bar{\mathbf{x}} = [44.02, 1349.61, 63.28]^T$ mm.

In order to build the snapshot matrix $\mathbf{S}$, we solve problem (1.3) completed with the conductivity tensor (4.9) on a mesh made by $N_h = 16365$ vertices over the interval $(0, T)$ with $T = 300$ ms and time-step $\Delta t = 0.1/12.9$. We uniformly sample $N_t = 1000$ time instances in $(0, T)$ and we zero-padded [Goodfellow et al., 2016] the snapshot matrix to reshape each column in a 2D square matrix. The parameter space is provided by $\mathcal{P} = 12.9 \cdot [0.04, 0.4]$ mm$^2$/ms; here we consider $N_{train} = 25$ training-parameter instances and $N_{test} = 24$ testing-parameter instances computed as in Test 4. In this case, the maximum number of epochs is set to $N_{epochs} = 30000$, the batch size is $N_b = 40$ and the training is stopped if the loss does not decrease over 4000 epochs.

In Figure 4.18 we report the FOM solution for two testing-parameter instances, i.e. $\mu = 12.9 \cdot 0.0739$ mm$^2$/ms and $\mu = 12.9 \cdot 0.1991$ mm$^2$/ms, at $\tilde{t} = 276$ ms, to show the variability of the FOM solution over the parameter space. As expected, front propagation is faster for larger values of the parameter $\mu$.

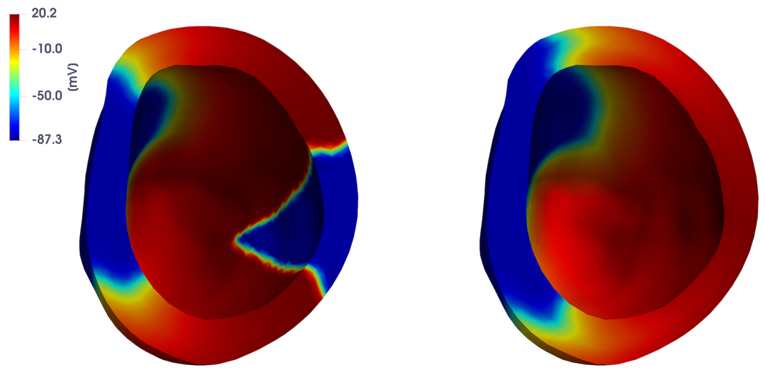


Figure 4.18: *Test 5*: FOM solutions for $\mu = 12.9 \cdot 0.0739$ mm$^2$/ms (left) and $\mu = 12.9 \cdot 0.1991$ mm$^2$/ms (right) at $\tilde{t} = 276$ ms. By increasing the value of $\mu$ the wavefront propagates faster.

In Figures 4.19 and 4.20 we report the FOM and DL-ROM solutions, the latter with $n = 10$, at $\tilde{t} = 42.1$ ms and $\tilde{t} = 222.1$ ms, for two testing-parameter instances, $\mu_{test} = 12.9 \cdot 0.1435$ mm$^2$/ms and $\mu_{test} = 12.9 \cdot 0.3243$ mm$^2$/ms. The DL-ROM approximation is essentially as accurate as the FOM solution.

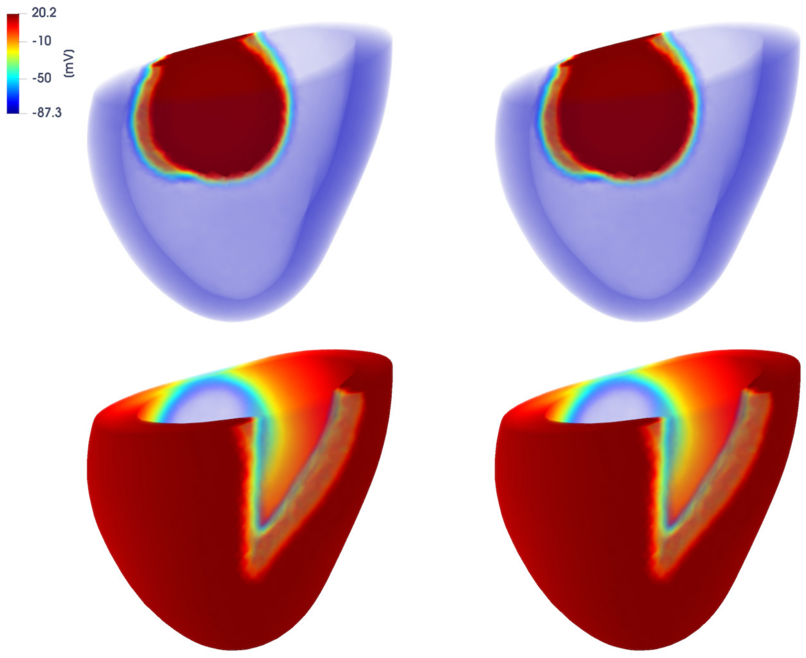


Figure 4.19: *Test 5*: FOM solution (left) and DL-ROM one (right), with $n = 10$, at $\tilde{t} = 42.1$ ms (top) and $\tilde{t} = 276$ ms (bottom), for the testing-parameter instance $\mu_{test} = 12.9 \cdot 0.1435$ mm$^2$/ms.

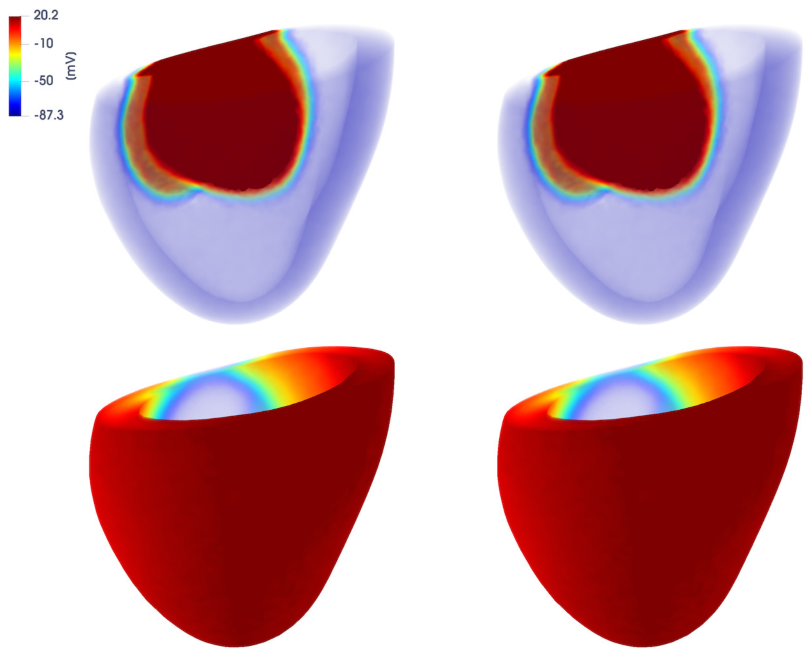


Figure 4.20: *Test 5*: FOM solution (left) and DL-ROM one (right), with $n = 10$, at $\tilde{t} = 42.1$ ms (top) and $\tilde{t} = 276$ ms (bottom), for the testing-parameter instance $\mu_{test} = 12.9 \cdot 0.3243$ mm$^2$/ms.

Also for this test case, it is possible to build a reduced nonlinear trial manifold of dimension very close to the intrinsic one – $n_\mu + 1 = 2$ – as long as the maximum number of epochs $N_{epochs}$ is increased; the choice $n = 10$ is obtained as the best trade-off between accuracy and efficiency of the DL-ROM approximation in this case.

The DL-ROM approximation can also replace the FOM solution when evaluating outputs of interest. For instance, in Figure 4.21 we show the FOM and DL-ROM activation maps (ACs), the latter obtained by choosing $n = 10$ as DL-ROM dimension. Given the electric potential $u = u(\mathbf{x}, t; \boldsymbol{\mu})$, the (unipolar) activation map at a point $\mathbf{x} \in \Omega$ is evaluated as the minimum time at which the AP peak reaches $\mathbf{x}$, that is

$$AC(\mathbf{x}; \boldsymbol{\mu}) = \arg \min_{t \in (0,T)} \left( u(\mathbf{x}, t; \boldsymbol{\mu}) = \max_{t \in (0,T)} u(\mathbf{x}, t; \boldsymbol{\mu}) \right).$$

Here we compare the activation maps $AC_{FOM}$ and $AC_{DL-ROM}$ obtained through the FOM and the DL-ROM, respectively, by evaluating the maximum of the relative error

$$\boldsymbol{\epsilon}_{AC}(\mathbf{x}; \boldsymbol{\mu}) = \frac{|AC_{FOM}(\mathbf{x}; \boldsymbol{\mu}) - AC_{DL-ROM}(\mathbf{x}; \boldsymbol{\mu})|}{|AC_{FOM}(\mathbf{x}; \boldsymbol{\mu})|}$$

over the $N_h$ mesh points; in the case $\mu_{test} = 12.9 \cdot 0.31$, the maximum relative error is equal to $4.32 \times 10^{-5}$.

In Figure 4.22 (left) we report the APs obtained by the FOM and the DL-ROM, this latter with $n = 10$, computed at point $P = (36.56, 1329.59, 28.82)$ mm for the testing-parameter instance $\mu_{test} = 12.9 \cdot 0.31$ mm$^2$/ms. For the sake of comparison, we also report the POD-Galerkin ROM approximation, with $N_c = 1$, of dimension $n = 10$ and $n = 120$. Clearly, in dimension $n = 10$ the DL-ROM approximation is far more accurate than the POD-Galerkin ROM approximation; to reach the same accuracy (about $\epsilon_{rel} = 5.9 \times 10^{-3}$, measured through the error indicator (3.13)) achieved by the DL-ROM with $n = 10$, $n = 120$ POD modes

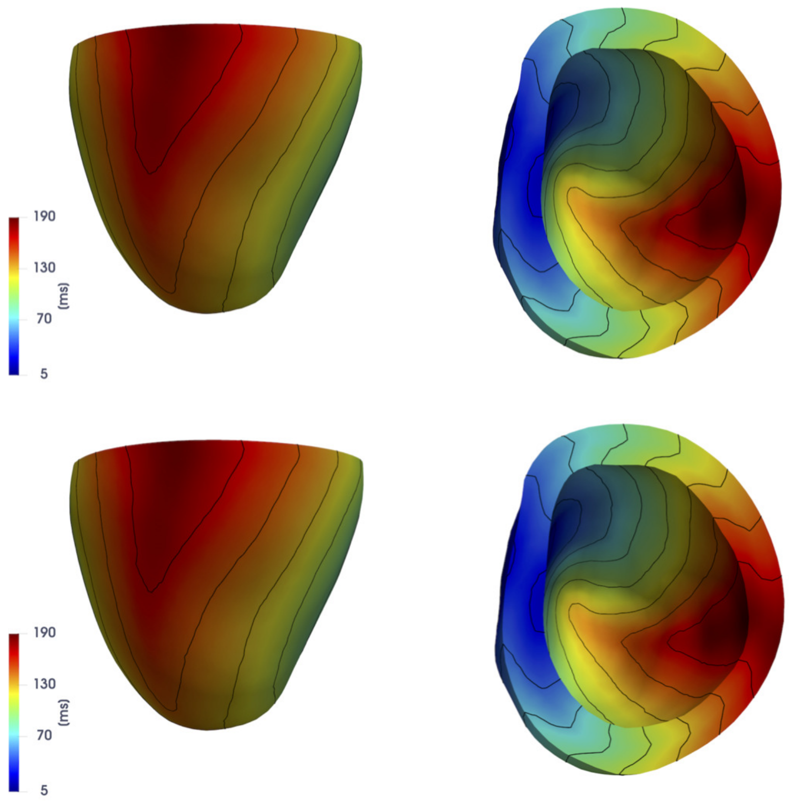


Figure 4.21: *Test 5*: FOM (top) and DL-ROM (bottom) ACs for the testing-parameter instance $\mu_{test} = 12.9 \cdot 0.31$ mm$^2$/ms with $n = 10$.

would be required. In Figure 4.22 (right) we highlight instead the improvements, in terms of efficiency, enabled by the use of the DL-ROM technique; we report the CPU time required to solve the FOM for a testing-parameter instance, the one required by DL-ROM (of dimension $n = 10$) at testing time and by the POD-Galerkin ROM with $N_c = 4$ ($n = 68, 81, 82, 45$), by using the time resolution each technique requires and by varying the FOM dimension $N_h$ on a 6-core platform[8]. The FOM solution with $N_h = 16365$ DOFs requires about 40 minutes to be computed, against 57 seconds required by the DL-ROM approximation, thus implying a speed-up almost equal to 41 times.

Regarding the training (offline) times for this test case, featuring a FOM dimension $N_h = 16365$, training the DL-ROM neural network on a GTX 1070 8GB GPU requires about 160 hours, whereas training the POD-Galerkin ROM (with $N_c = 4$ local bases) on a full node (20 Intel® Xeon® E5-2640 v4 2.4GHz cores) of a HPC cluster requires about 28 hours; in both cases, the time needed to assemble the snapshot matrix $\mathbf{S}$ is not included. We report also the GPU testing computational time of DL-ROM which is equal to 0.35 seconds thus obtaining a speed-up, with respect the POD-Galerkin ROM testing time with $N_c = 4$, equal to 172. The efficiency introduced at testing time justifies the higher training time of DL-ROM.

## 4.4 Discussion

The numerical results, presented in this Chapter, show that the DL-ROM technique, proposed in Chapter 3, allows to accurately capture complex front propagation processes in cardiac

[8]Numerical tests have been performed on a MacBook Pro Intel Core i7 6-core with 16 GB RAM.

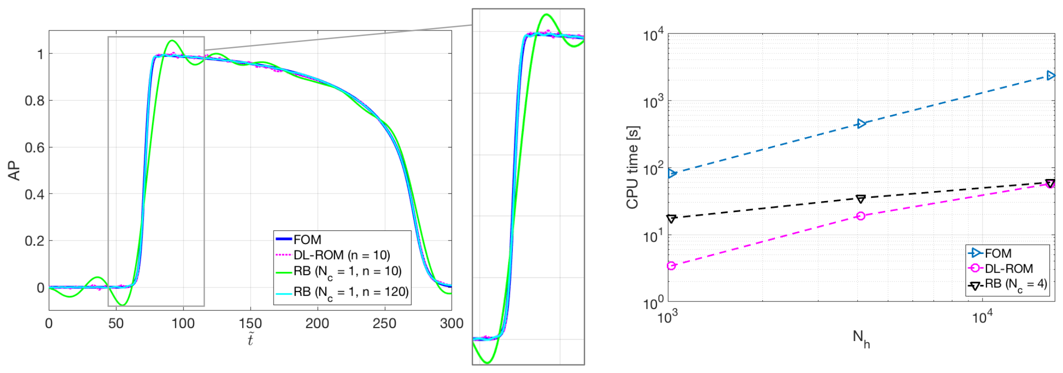


Figure 4.22: *Test 5*: Left: FOM, DL-ROM and POD-Galerkin ROM APs for the testing-parameter instance $\mu_{test} = 12.9 \cdot 0.31$ mm$^2$/ms. For the same $n$, the DL-ROM is able to provide more accurate results than the POD-Galerkin ROM. Right: CPU time required to solve the FOM, by DL-ROM with $n = 10$ and by the POD-Galerkin ROM with $N_c = 6$ at testing time vs. $N_h$. The DL-ROM is able to provide a speed-up equal to 41.

EP, both in physiological and pathological scenarios. Moreover, it provides ROMs that are orders of magnitude more efficient than the ones obtained by common linear (projection-based) ROMs, built for instance through a POD-Galerkin RB method, for a prescribed level of accuracy. Through the use of DL-ROM, it is possible to overcome the main computational bottlenecks shown by POD-Galerkin ROMs, when addressing parametrized problems in cardiac EP. The most critical points related to *(i)* the linear superimposition of modes which linear ROMs are based on; *(ii)* the need to account for the gating variables when solving the reduced dynamics, even if not required; and *(iii)* the necessity to use (very often expensive) hyper-reduction techniques to deal with terms that depend nonlinearly on either the transmembrane potential or the input parameters, are all avoided by the DL-ROM technique, which finally yields more efficient and accurate approximation than POD-Galerkin ROMs. Moreover, larger time resolutions can be employed when using DL-ROM, compared to the ones required by the numerical solution of a dynamical systems through a FOM or a POD-Galerkin ROM. Indeed, the DL-ROM approximation can be queried at any desired time, without requiring to solve a dynamical system until that time, thus drastically decreasing the computational time required to compute the approximated solution at any given time.

Outputs of clinical interest, such as ACs and APs, can be more efficiently evaluated by the DL-ROM technique than by a FOM built through the FE method, while maintaining a high level of accuracy. The numerical tests carried out in this Chapter are a proof-of-concept of the DL-ROM technique ability to investigate intra- and inter-subjects variability, towards performing multi-scenario analyses in real-time and, ultimately, supporting decisions in clinical practice. In this respect, the use of DL-ROM techniques can foster assimilation of clinical data with physics-driven computational models.

So far, the training time required by the DL-ROM technique appears to be the major computational bottleneck, even if it is completely compensated by the wider computational efficiency provided at testing time. The DL-ROM depends on $N_h$ and this may lead to long training computational times for large FOM dimension. Enhancing efficiency also during the training phase, which we afford by means of the POD-enhanced DL-ROM, represents the focus of the next Chapter.

Chapter 5

# Deep learning-based reduced order models enhanced by POD

In this Chapter[1], we address a key aspect in the construction of DL-ROMs, related with computational efficiency during the training phase. Although extremely efficient at testing time, when evaluating the PDE solution for any new testing-parameter instance, DL-ROMs introduced in Chapter 3, require an expensive training stage, because of the extremely large number of network parameters to be estimated. In this Chapter we propose a possible way to avoid an expensive training stage of DL-ROMs, by *(i)* performing a prior dimensionality reduction through proper orthogonal decomposition (POD), and *(ii)* relying on a multi-fidelity pretraining stage, where different physical models can be efficiently combined. The proposed POD DL-ROM is then assessed on different – both scalar and vector – nonlinear time-dependent parametrized PDEs (including advection-diffusion-reaction, structural mechanics and fluid dynamics problems) to show the flexibility of this approach and its remarkable computational savings.

## 5.1 Making ROMs non-intrusive by ANNs & deep learning

Merging classical projection-based ROMs, such as the POD-Galerkin method, and ANNs has been shown to be a suitable approach in view of making ROMs less intrusive and more efficient. In this respect, several strategies have been proposed to replace the projection stage in classical ROMs for parametrized PDEs and model the reduced dynamics. For instance, in [Guo and Hesthaven, 2018, Guo and Hesthaven, 2019, Hesthaven and Ubbiali, 2018, Kast et al., 2020, Wang et al., 2019] ANN-based or Gaussian Process (GP) regression models have been proposed to approximate the mapping from the input parameters (and, possibly, time) to the reduced coefficients, as an alternative to the assembling and solution of the reduced order system arising from POD-Galerkin ROMs, however still using a linear trial manifold built through POD; similar strategies have also been introduced in [Kani and Elsheikh, 2017, Mohan and Gaitonde, 2018, Wan et al., 2018, Pulch and Youssef, 2020, Bērzinš et al., 2020].

In particular, in [Hesthaven and Ubbiali, 2018, Wang et al., 2019] the intrinsic coordinates are approximated by means of a DFNN. In the POD DL-ROM technique introduced in this Chapter, a further level of dimensionality reduction is performed, that is the dimension of the intrinsic coordinates is decreased by matching the minimal dimension of the problem.

[1]This Chapter is mainly based on the paper [Fresca and Manzoni, 2021].

Moreover, we exploit convolutional layers which result more suited to high-dimensional spatial data, thus implying lower computational costs, with respect to dense layers. In [Guo and Hesthaven, 2019] a data-driven RB method is presented where GPs are employed as regression model for the approximation of the intrinsic coordinates. The regression functions are represented as the linear combinations of several tensor products of two GPs, one depending on the variable time and the other taken as function of the input parameters. In particular, to decompose the training data associated to each element of the intrinsic coordinates vector into several time- and parameter-modes, SVD is applied. Then GPs are trained to approximate the modes of each intrinsic coefficient. This approach is extended in [Kast et al., 2020], where a multi-fidelity setup in used in two ways: first in the reduced basis construction and secondly in the GP regression. However, when dealing with high POD dimensions, the previous strategies may entail the training of a very high number of regression models. By means of the POD DL-ROM instead, we are able to model and approximate the entire intrinsic coordinates vector all at once and no additional SVDs are required.

Furthermore, a hybrid strategy proposed in [Chen et al., 2020] merges ANN-based regression models and PINNs, training the network by minimizing the mean squared residual error of the reduced order equation on a set of points in the parameter space; similar results can be found in [Kani and Elsheikh, 2018].

## 5.2 (Randomized) POD for dimensionality reduction

POD exploits the SVD of the snapshot matrix $\mathbf{S}$ to compute an orthonormal basis, which is optimal in a least-squares sense (see Section 2.3). Due to the linear superimposition of modes assumption, POD is not able to compute a reduced subspace of dimension equal (or close) to the dimension of the solution manifold, that is to say $n_\mu + 1$ by using the notation introduced in Chapter 2, while preserving an acceptable degree of accuracy with respect to the FOM solution, when dealing with problems featuring coherent structures propagating over time. In Chapters 3 and 4 we showed, by computing the optimal-POD or the POD-Galerkin ROM solutions, that the dimension $N$ of the linear trial manifold built by means of POD, must be much higher than $n_\mu + 1$ in order to ensure the same level of accuracy obtained with a DL-ROM. Hence, although POD does not capture the intrinsic dimension of the problem under consideration it is still able to perform a moderate dimensionality reduction, thus yielding a dimension $N \ll N_h$. This latter feature is exploited in the POD DL-ROM strategy.

Computing the SVD can be extremely time consuming for large-scale problems; if we deal with a snapshot matrix $\mathbf{S} \in \mathbb{R}^{N_h \times N_{train} N_t}$ characterized by a large FOM dimension $N_h$ and/or a high number of training-parameter or time instances, the computational time and the memory needed to compute the SVD may become prohibitive. For every large values of $N_h$ and $N_{train} N_t$, the time and the memory required by the SVD are superlinear in $N_h$ and $N_{train} N_t$ [Drineas et al., 2006]. In order to speed up computations, we rely on the randomized matrix approximation technique developed in [Halko et al., 2011]. Randomized SVD (rSVD) computes an approximated SVD using randomization. It consists of two main stages: *(i)* in the first one an approximated basis for Col($\mathbf{S}$), i.e. $\mathbf{Q}$, is obtained by using randomization, and *(ii)* in the second one the SVD of a smaller matrix $\mathbf{B}$ is computed and then the SVD of $\mathbf{S}$ is recovered by means of $\mathbf{Q}$. More precisely, a Gaussian random matrix $\mathbf{\Omega} \in \mathbb{R}^{N_{train} N_t \times m}$ is drawn, where $N \leq m \leq N_{train} N_t$ and $m - N$ is called the oversampling parameter. The matrix

$$\mathbf{Y} = (\mathbf{S}\mathbf{S}^T)^q \mathbf{S}\mathbf{\Omega}$$

is then assembled, by normally setting $q = 1$ or 2, and an iterative QR factorization, i.e. $\mathbf{Y} = \mathbf{Q}\mathbf{R}$, is computed where $\mathbf{Q} \in \mathbb{R}^{N_h \times m}$. Computing $\mathbf{Q}$ in such a way consists in applying an adaptive randomized range finder algorithm to approximate the range of $\mathbf{S}$ by means of a matrix $\mathbf{Q}$ whose colums are orthonormal, i.e. $\mathbf{S} \approx \mathbf{Q}\mathbf{Q}^T\mathbf{S}$ [Halko et al., 2011]. Once $\mathbf{Q}$ has been computed, $N$ columns are selected, i.e. $\mathbf{Q} \in \mathbb{R}^{N_h \times N}$, and the SVD of the matrix

$$\mathbf{B} = \mathbf{Q}^T\mathbf{S} = \tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{Z}}$$

is computed. The SVD factorization of $\mathbf{S}$ is recovered by simply setting

$$\mathbf{V} = \mathbf{Q}\tilde{\mathbf{V}}.$$

We refer to $\mathbf{V}$ as the rPOD basis matrix. For futher details about rSVD we refer to [Halko et al., 2011, Szlam et al., 2014].

By applying rSVD, we build a reduced linear trial manifold $\tilde{\mathcal{S}}_h = \mathrm{Col}(\mathbf{V})$ of dimension $N \ll N_h$, approximating the solution manifold $\mathcal{S}_h$. The ROM approximation is then provided by

$$\mathbf{u}_h(t; \boldsymbol{\mu}) \approx \tilde{\mathbf{u}}_h(t; \boldsymbol{\mu}) = \mathbf{V}\mathbf{u}_N(t; \boldsymbol{\mu}),$$

where $\tilde{\mathbf{u}}_h : [0, T) \times \mathcal{P} \to \tilde{\mathcal{S}}_h$. Here $\mathbf{u}_N(t; \boldsymbol{\mu}) \in \mathbb{R}^N$, for each $t \in [0, T)$ and $\boldsymbol{\mu} \in \mathcal{P}$, denotes an approximation of the vector of the intrinsic coordinates of the ROM approximation on the linear subspace generated through rSVD, that is

$$\mathbf{u}_N(t; \boldsymbol{\mu}) \approx \mathbf{V}^T\mathbf{u}_h(t; \boldsymbol{\mu}).$$

We finally define the whole set

$$\mathcal{S}_N = \{\mathbf{V}^T\mathbf{u}_h(t; \boldsymbol{\mu}) \mid t \in [0, T) \text{ and } \boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{n_\mu}\} \subset \mathbb{R}^N, \tag{5.1}$$

of the intrinsic coordinates, when $(t, \boldsymbol{\mu})$ varies in $[0, T) \times \mathcal{P}$, which is a, possibly nonlinear, manifold of dimension $n_\mu + 1 \leq \dim(\mathcal{S}_N) \ll N$.

## 5.3 Pretraining

Directly training a model to solve a specific task can be very demanding if the model is complex and hard to optimize, or if the task is very difficult. It is sometimes more effective to train a simpler model to solve the task, then make the model more complex. It can also be more effective to train the model to solve a simpler task, then move on to the final task. In a broad sense, these strategies are known as *pretraining* [Goodfellow et al., 2016]. In particular, fine tuning a pretrained model is equivalent to *transfer learning*, if the data on which the model is fine tuned is of a different nature from the original data used to pretrain the model. Pretraining is then a form of transfer learning, where a pretrained model is used as initial state of the network [Taylor and Stone, 2009]; this strategy works extremely well in many objects classification tasks [Yosinski et al., 2014] and natural language processing problems [Devlin et al., 2018]. In the area of scientific ML, pretraining has been used, for example, in [Haghighat et al., 2020] where a pretrained neural network is used to perform parameter identification on a new dataset.

By means of pretraining, we are able to combine models with different fidelities (e.g., by considering coarser/finer spatial discretizations, different physical laws, more/less parameters or larger/smaller parameter spaces) in order to enhance the training phase of the POD DL-ROM. In particular, we train the POD DL-ROM neural network on an initial simpler configuration and use the optimal weights and biases found by this training procedure as initializers of the POD DL-ROM neural network on a more complex problem (see Figure 5.1). Together with dimensionality reduction by rPOD, this strategy has been proved to be a cornerstone in view of drastically reducing the training computational time of the POD DL-ROM with respect to the time needed to train the neural network, from scratch, on the more complex task.

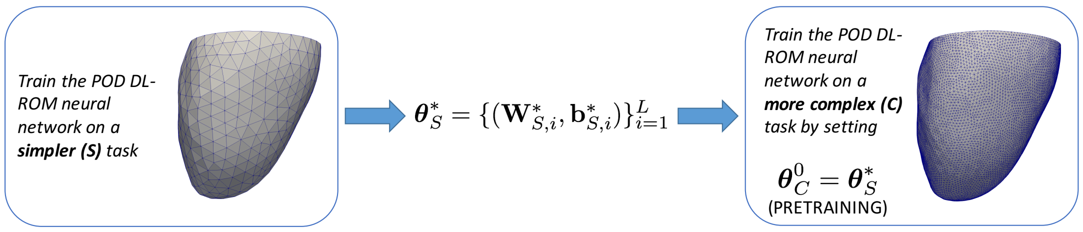

Train the POD DL-ROM neural network on a **simpler (S)** task
$\theta_S^* = \{(\mathbf{W}_{S,i}^*, \mathbf{b}_{S,i}^*)\}_{i=1}^L$
Train the POD DL-ROM neural network on a **more complex (C)** task by setting
$\theta_C^0 = \theta_S^*$
(PRETRAINING)

Figure 5.1: Pretraining workflow.

## 5.4 POD-enhanced DL-ROMs (POD DL-ROMs)

The POD DL-ROM strategy then represents a suitable combination of the best features of DL algorithms and POD – namely, the non-intrusive character of the former, and the simplicity, combined with the rigorous mathematical foundation, of the latter – at the same time mutually compensating their weaknesses – namely, the curse of dimensionality of DL-ROM for increasing FOM dimensions, and the modest approximation properties of POD-based linear trial manifolds for some classes of nonlinear parametrized PDEs.

As previously pointed out, DL-ROM explicitly depends on $N_h$ and, when the FOM dimension is large, this may lead to long training computational times, even if it remains very efficient at testing time. In order to overcome the major computational bottleneck of DL-ROMs, we extend them by approximating, by means of neural networks, the intrinsic coordinates manifold $\mathcal{S}_N$. More precisely, we build a nonlinear ROM to approximate $\boldsymbol{V}^T\mathbf{u}_h(t;\boldsymbol{\mu}) \approx \tilde{\mathbf{u}}_N(t;\boldsymbol{\mu})$ by

$$\tilde{\mathbf{u}}_N(t;\boldsymbol{\mu}) = \boldsymbol{\Psi}_N(\mathbf{u}_n(t;\boldsymbol{\mu})), \tag{5.2}$$

where $\boldsymbol{\Psi}_N : \mathbb{R}^n \to \mathbb{R}^N$, $\boldsymbol{\Psi}_N : \mathbf{s}_n \mapsto \boldsymbol{\Psi}_N(\mathbf{s}_n)$, $n \ll N$, is a nonlinear, differentiable function. In this way, the manifold $\mathcal{S}_N$ is approximated by the $n$-dimensional reduced nonlinear trial manifold

$$\tilde{\mathcal{S}}_n = \{\boldsymbol{\Psi}_N(\mathbf{u}_n(t;\boldsymbol{\mu})) \mid \mathbf{u}_n(t;\boldsymbol{\mu}) \in \mathbb{R}^n,\ t \in [0,T) \text{ and } \boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{n_\mu}\} \subset \mathbb{R}^N, \tag{5.3}$$

where $\tilde{\mathbf{u}}_N : [0,T) \times \mathcal{P} \to \tilde{\mathcal{S}}_n$. The function $\mathbf{u}_n : [0,T) \times \mathcal{P} \to \mathbb{R}^n$ denotes the minimal coordinates of $\tilde{\mathbf{u}}_N$ on the linear trial manifold $\tilde{\mathcal{S}}_n$. Our goal is, again, to set-up a ROM whose dimension $n$ is as close as possible to the intrinsic dimension $n_{\boldsymbol{\mu}} + 1$ of the solution manifold $\mathcal{S}_h$, i.e. $n \geq n_{\boldsymbol{\mu}} + 1$, in order to correctly capture the degrees of freedom of the set $\mathcal{S}_N$ by containing its size [Lee and Carlberg, 2020].

To model the relationship between each pair $(t, \boldsymbol{\mu}) \mapsto \mathbf{u}_n(t, \boldsymbol{\mu})$, and to describe the reduced dynamics on the reduced nonlinear trial manifold $\tilde{\mathcal{S}}_n$, we consider a nonlinear map under the form

$$\mathbf{u}_n(t;\boldsymbol{\mu}) = \boldsymbol{\Phi}_n(t, \boldsymbol{\mu}), \tag{5.4}$$

where $\boldsymbol{\Phi}_n : [0,T) \times \mathbb{R}^{n_\mu} \to \mathbb{R}^n$ is a differentiable, nonlinear function.

As in DL-ROM (see Chapter 3), both the reduced dynamics and the reduced nonlinear trial manifold must be learnt. Now, in the first case we aim at learning the dynamics of the set $\mathcal{S}_N$ on the nonlinear trial manifold $\tilde{\mathcal{S}}_n$ in terms of minimal coordinates by means of a DFNN; indeed, we set the function $\boldsymbol{\Phi}_n$ in (5.4) equal to

$$\boldsymbol{\Phi}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF}) = \boldsymbol{\phi}_n^{DF}(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF}),$$

where $\boldsymbol{\theta}_{DF}$ denotes again the vector of parameters of the DFNN, collecting all the corresponding weights and biases of each layer of the DFNN.

For the description of the reduced nonlinear trial manifold we instead employ the decoder function of a convolutional AE, that is, we define the function in (5.2) as

$$\boldsymbol{\Psi}_h(\mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF});\boldsymbol{\theta}_D) = \mathbf{f}_N^D(\mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF});\boldsymbol{\theta}_D),$$

where $\mathbf{f}_N^D$ depends on the vector $\boldsymbol{\theta}_D$ of parameters of the convolutional and dense layers of the decoder function.

By combining the two previous stages, the POD DL-ROM approximation $\tilde{\mathbf{u}}_N$ takes the form

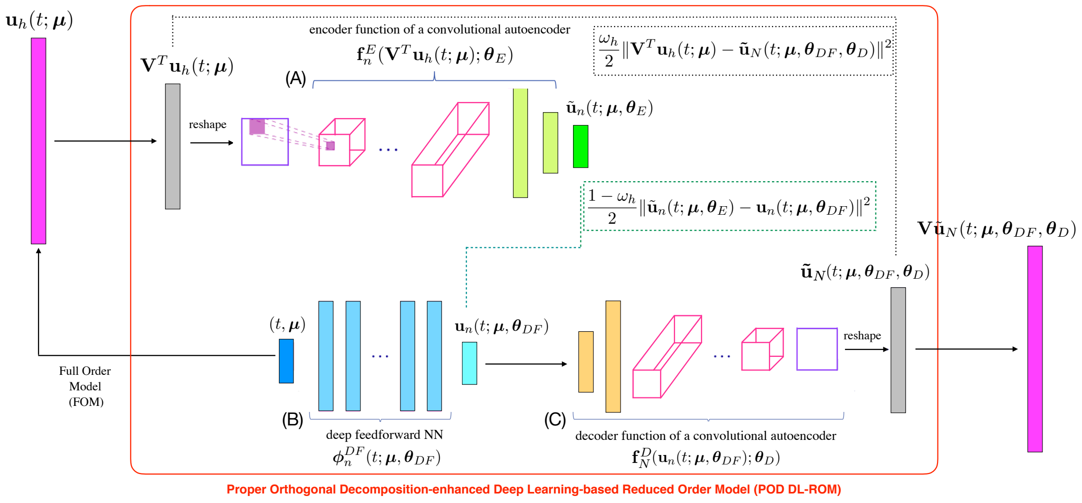
$$\tilde{\mathbf{u}}_N(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF},\boldsymbol{\theta}_D) = \mathbf{f}_N^D(\boldsymbol{\phi}_n^{DF}(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF});\boldsymbol{\theta}_D). \tag{5.5}$$

The encoder function of the convolutional AE can then be exploited to map the intrinsic coordinates $\mathbf{V}^T\mathbf{u}_h$ associated to $(t,\boldsymbol{\mu})$ onto a low-dimensional representation

$$\tilde{\mathbf{u}}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_E) = \mathbf{f}_n^E(\mathbf{V}^T\mathbf{u}_h(t;\boldsymbol{\mu});\boldsymbol{\theta}_E),$$

where $\mathbf{f}_n^E$ denotes the encoder function, depending upon a vector $\boldsymbol{\theta}_E$ of parameters.

The architecture of the POD DL-ROM neural network, employed at training time, is the one shown in Figure 5.2. As in Chapter 3, at testing time we can discard the encoder function.



$\mathbf{u}_h(t;\boldsymbol{\mu})$
$\mathbf{V}^T\mathbf{u}_h(t;\boldsymbol{\mu})$
encoder function of a convolutional autoencoder
$\mathbf{f}_n^E(\mathbf{V}^T\mathbf{u}_h(t;\boldsymbol{\mu});\boldsymbol{\theta}_E)$
(A)
$\frac{\omega_h}{2}\|\mathbf{V}^T\mathbf{u}_h(t;\boldsymbol{\mu}) - \tilde{\mathbf{u}}_N(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF},\boldsymbol{\theta}_D)\|^2$
reshape
$\tilde{\mathbf{u}}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_E)$
$\frac{1-\omega_h}{2}\|\tilde{\mathbf{u}}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_E) - \mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF})\|^2$
$\mathbf{V}\tilde{\mathbf{u}}_N(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF},\boldsymbol{\theta}_D)$
$\tilde{\mathbf{u}}_N(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF},\boldsymbol{\theta}_D)$
$(t,\boldsymbol{\mu})$
$\mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF})$
reshape
Full Order Model (FOM)
(B)
deep feedforward NN
$\boldsymbol{\phi}_n^{DF}(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF})$
(C)
decoder function of a convolutional autoencoder
$\mathbf{f}_N^D(\mathbf{u}_n(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF});\boldsymbol{\theta}_D)$
Proper Orthogonal Decomposition-enhanced Deep Learning-based Reduced Order Model (POD DL-ROM)

Figure 5.2: POD DL-ROM architecture. Starting from the FOM solution $\mathbf{u}_h(t;\boldsymbol{\mu})$, the intrinsic coordinates $\mathbf{V}^T\mathbf{u}_h(t;\boldsymbol{\mu})$ are computed, by means of rSVD, and the neural network provides as output $\tilde{\mathbf{u}}_N(t;\boldsymbol{\mu})$, an approximation of them. The reconstructed solution $\tilde{\mathbf{u}}_h(t;\boldsymbol{\mu})$ is then recovered trough the rPOD basis matrix.

Computing the ROM approximation (5.5), by means of POD DL-ROM thus consists in solving the optimization problem (3.8) where now the per-example loss function is given by

$$\begin{aligned}\mathcal{L}(t^k,\boldsymbol{\mu}_i;\boldsymbol{\theta}) = &\frac{\omega_h}{2}\|\mathbf{V}^T\mathbf{u}_h(t^k;\boldsymbol{\mu}_i) - \tilde{\mathbf{u}}_N(t^k;\boldsymbol{\mu}_i,\boldsymbol{\theta}_{DF},\boldsymbol{\theta}_D)\|^2\\ &+\frac{1-\omega_h}{2}\|\tilde{\mathbf{u}}_n(t^k;\boldsymbol{\mu}_i,\boldsymbol{\theta}_E) - \mathbf{u}_n(t^k;\boldsymbol{\mu}_i,\boldsymbol{\theta}_{DF})\|^2.\end{aligned} \tag{5.6}$$

The POD DL-ROM approximation of the FOM solution $\tilde{\mathbf{u}}_h(t;\boldsymbol{\mu}) \approx \mathbf{u}_h(t;\boldsymbol{\mu})$ is then recovered by means of the rPOD basis matrix

$$\tilde{\mathbf{u}}_h(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF},\boldsymbol{\theta}_D) = \mathbf{V}\tilde{\mathbf{u}}_N(t;\boldsymbol{\mu},\boldsymbol{\theta}_{DF},\boldsymbol{\theta}_D).$$

## 5.4.1 Extension to vector problems

We have also extended the POD DL-ROM technique to treat vector problems, by exploiting the analogy with RGB images. By considering the spatial discretization of a PDE, whose solution is a $d$-dimensional vector field, the initial value problem (2.1) can be rewritten as

$$\begin{cases} \dot{\mathbf{u}}_h^1(t;\boldsymbol{\mu}) = \mathbf{f}^1(t, \mathbf{u}_h^1(t;\boldsymbol{\mu}), \dots, \mathbf{u}_h^d(t;\boldsymbol{\mu}); \boldsymbol{\mu}) & t \in (0,T), \\ \vdots \\ \dot{\mathbf{u}}_h^d(t;\boldsymbol{\mu}) = \mathbf{f}^d(t, \mathbf{u}_h^1(t;\boldsymbol{\mu}), \dots, \mathbf{u}_h^d(t;\boldsymbol{\mu}); \boldsymbol{\mu}) & t \in (0,T), \\ \mathbf{u}_h^1(0;\boldsymbol{\mu}) = \mathbf{u}_0^1(\boldsymbol{\mu}), \\ \vdots \\ \mathbf{u}_h^d(0;\boldsymbol{\mu}) = \mathbf{u}_0^d(\boldsymbol{\mu}), \end{cases} \tag{5.7}$$

where $\mathbf{u}_h^i : [0,T) \times \mathcal{P} \to \mathbb{R}^{N_h^i}$ is the solution of the $i$-th equation in (5.7), $\mathbf{u}_0^i : \mathcal{P} \to \mathbb{R}^{N_h^i}$ is the $i$-th initial datum, $\mathbf{f}^i : (0,T) \times \mathbb{R}^{N_h^i} \times \mathcal{P} \to \mathbb{R}^{N_h^i}$ describes the dynamics of $\mathbf{u}_h^i(t;\boldsymbol{\mu})$ and $i = 1, \dots, d$, with $d = 2, 3$. Depending on the problem at hand, some of the equations appearing in (5.7) might not involve the derivatives and related initial conditions; this is what happens, for instance, in the case of unsteady Navier-Stokes equations for incompressible flows, where the equation expressing flow incompressibility involves the velocity components but no time derivatives.

Provided the solution of (5.7), associated to a particular instance $(t, \boldsymbol{\mu})$, and the orthonormal basis $\mathbf{V}_i \in \mathbb{R}^{N_h^i \times N}$, with $i = 1, \dots, d$, found though rSVD, we compute the intrinsic components $\mathbf{V}_1^T \mathbf{u}_h^1(t;\boldsymbol{\mu}), \dots, \mathbf{V}_d^T \mathbf{u}_h^d(t;\boldsymbol{\mu})$, reshape each component in a square matrix of dimension $(\sqrt{N}, \sqrt{N})$, where $N = 4^m$ with $m \in \mathbb{N}$, and stack them together thus forming a tensor with $d$ channels (a sketch in Figure 5.3).

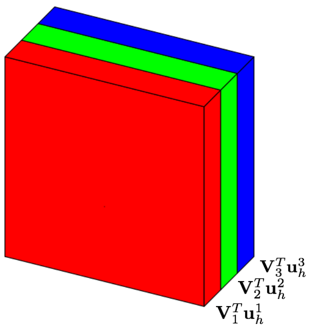

$\mathbf{V}_3^T\mathbf{u}_h^3$
$\mathbf{V}_2^T\mathbf{u}_h^2$
$\mathbf{V}_1^T\mathbf{u}_h^1$

Figure 5.3: Each vectorial component of the solution of problem (5.7) is reshaped in a square matrix and they are stacked together thus forming a tensor of dimension $(\sqrt{N}, \sqrt{N}, 3)$.

This approach allows $N_h^i$, $i = 1, \dots, d$, to be different. Indeed, it is the rPOD dimension $N$ of the different vectorial components that must be kept equal for $i = 1, \dots, d$. We remark that stacking all components together allows to reduce the number of parameters and then the training and testing computational times of POD DL-ROM.

Regarding the optimization algorithm, the values of the hyperparameters of the POD DL-ROM and the configuration of the neural network we refer to Chapters 3 and 4. The input and the output of the POD DL-ROM are normalized by applying transformation (3.11) to each channel of the $d$-dimensional tensor. Once training is performed, the reconstructed solution is rescaled to the original values by applying the inverse transformation of (3.11). In the vector case, the snapshot matrix $\mathbf{S} \in \mathbb{R}^{\sum_i N_h^i \times N_{train} N_t}$ takes the form

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 \\ \dots \\ \mathbf{S}_d \end{bmatrix} = \begin{bmatrix} \mathbf{u}_h^1(t^1;\boldsymbol{\mu}_1) \mid \dots \mid \mathbf{u}_h^1(t^{N_t};\boldsymbol{\mu}_1) \mid \dots \mid \mathbf{u}_h^1(t^1;\boldsymbol{\mu}_{N_{train}}) \mid \dots \mid \mathbf{u}_h^1(t^{N_t};\boldsymbol{\mu}_{N_{train}}) \\ \dots \\ \mathbf{u}_h^d(t^1;\boldsymbol{\mu}_1) \mid \dots \mid \mathbf{u}_h^d(t^{N_t};\boldsymbol{\mu}_1) \mid \dots \mid \mathbf{u}_h^d(t^1;\boldsymbol{\mu}_{N_{train}}) \mid \dots \mid \mathbf{u}_h^d(t^{N_t};\boldsymbol{\mu}_{N_{train}}) \end{bmatrix}.$$

We detail the algorithms through which the training and the testing of the neural network are performed in Algorithms 3 and 4. During the training phase, the optimal parameters of the POD DL-ROM are found by solving the optimization problem (3.8)-(5.6) through the back-propagation and ADAM algorithms (see Algorithm 3). At testing time, the encoder function is instead discarded (see Algorithm 4).

**Algorithm 3** POD DL-ROM training algorithm

```
Input: Parameter matrix $\mathbf{M} \in \mathbb{R}^{(n_\mu+1)\times N_s}$, snapshot matrix $\mathbf{S} \in \mathbb{R}^{\sum_i N_h^i \times N_s}$, training-validation splitting fraction $\alpha$, starting learning rate $\eta$, batch size $N_b$, maximum number of epochs $N_{epochs}$, early stopping criterion, number of minibatches $N_{mb} = (1-\alpha)N_s/N_b$.
Output: Optimal model parameters $\boldsymbol{\theta}^* = (\boldsymbol{\theta}_E^*, \boldsymbol{\theta}_{DF}^*, \boldsymbol{\theta}_D^*)$.

1: Compute rPOD basis matrix $\mathbf{V} = [\mathbf{V}_1|\dots|\mathbf{V}_d]^T$
2: Randomly shuffle $\mathbf{M}$ and $\mathbf{S}$
3: Split data in $\mathbf{M} = [\mathbf{M}^{train}, \mathbf{M}^{val}]$ and $\mathbf{S} = [\mathbf{S}^{train}, \mathbf{S}^{val}]$ ($\mathbf{M}^{val}, \mathbf{S}^{val} \in \mathbb{R}^{\sum_i N_h^i \times \alpha N_s}$)
4: Compute intrinsic coordinates $\mathbf{S}_N^{train} = [\mathbf{S}_1^{train}|\dots|\mathbf{S}_d^{train}]^T$ where $\mathbf{S}_i^{train} = \mathbf{V}_i^T \mathbf{S}_i^{train}$ with $i = 1, \dots, d$
5: Compute intrinsic coordinates $\mathbf{S}_N^{val} = [\mathbf{S}_1^{val}|\dots|\mathbf{S}_d^{val}]^T$ where $\mathbf{S}_i^{val} = \mathbf{V}_i^T \mathbf{S}_i^{val}$ with $i = 1, \dots, d$
6: Normalize data in $\mathbf{M}$ and $\mathbf{S}_N = [\mathbf{S}_N^{train}, \mathbf{S}_N^{val}]$
7: Randomly initialize $\boldsymbol{\theta}^0 = (\boldsymbol{\theta}_E^0, \boldsymbol{\theta}_{DF}^0, \boldsymbol{\theta}_D^0)$
8: $n_e = 0$
9: while ($\neg$early-stopping and $n_e \le N_{epochs}$) do
10:     for $k = 1 : N_{mb}$ do
11:         Sample a minibatch $(\mathbf{M}^{batch}, \mathbf{S}_N^{batch}) \subseteq (\mathbf{M}^{train}, \mathbf{S}_N^{train})$
12:         $\mathbf{S}_N^{batch} = \text{reshape}(\mathbf{S}_N^{batch}) \in \mathbb{R}^{N_b \times \sqrt{N} \times \sqrt{N} \times d}$
13:         $\widetilde{\mathbf{S}}_n^{batch}(\boldsymbol{\theta}_E^{N_{mb}n_e+k}) = \mathbf{f}_n^E(\mathbf{S}_N^{batch}; \boldsymbol{\theta}_E^{N_{mb}n_e+k})$
14:         $\mathbf{S}_n^{batch}(\boldsymbol{\theta}_{DF}^{N_{mb}n_e+k}) = \boldsymbol{\phi}_n^{DF}(\mathbf{M}^{batch}; \boldsymbol{\theta}_{DF}^{N_{mb}n_e+k})$
15:         $\widetilde{\mathbf{S}}_N^{batch}(\boldsymbol{\theta}_{DF}^{N_{mb}n_e+k}, \boldsymbol{\theta}_D^{N_{mb}n_e+k}) = \mathbf{f}_N^D(\mathbf{S}_n^{batch}(\boldsymbol{\theta}_{DF}^{N_{mb}n_e+k}); \boldsymbol{\theta}_D^{N_{mb}n_e+k})$
16:         $\widetilde{\mathbf{S}}_N^{batch} = \text{reshape}(\widetilde{\mathbf{S}}_N^{batch}) \in \mathbb{R}^{N_b \times N \times d}$
17:         Accumulate loss (5.6) on $(\mathbf{M}^{batch}, \mathbf{S}_N^{batch})$ and compute $\widehat{\nabla}_\theta \mathcal{J}$
18:         $\boldsymbol{\theta}^{N_{mb}n_e+k+1} = \text{ADAM}(\eta, \widehat{\nabla}_\theta \mathcal{J}, \boldsymbol{\theta}^{N_{mb}n_e+k})$
19:     end for
20:     Repeat instructions 12-16 on $(\mathbf{M}^{val}, \mathbf{S}_N^{val})$ with the updated weights $\boldsymbol{\theta}^{N_{mb}n_e+k+1}$
21:     Accumulate loss (5.6) on $(\mathbf{M}^{val}, \mathbf{S}_N^{val})$ to evaluate early-stopping criterion
22:     $n_e = n_e + 1$
23: end while
```

**Algorithm 4** POD DL-ROM testing algorithm

```
Input: Testing parameter matrix $\mathbf{M}^{test} \in \mathbb{R}^{(n_\mu+1)\times(N_{test}N_t)}$, rPOD basis matrix $\mathbf{V}$, optimal model parameters $(\boldsymbol{\theta}_{DF}^*, \boldsymbol{\theta}_D^*)$.
Output: ROM approximation matrix $\widetilde{\mathbf{S}}_h \in \mathbb{R}^{\sum_i N_h^i \times (N_{test}N_t)}$.

1: Load $\boldsymbol{\theta}_{DF}^*$ and $\boldsymbol{\theta}_D^*$
2: $\mathbf{S}_n(\boldsymbol{\theta}_{DF}^*) = \boldsymbol{\phi}_n^{DF}(\mathbf{M}^{test}; \boldsymbol{\theta}_{DF}^*)$
3: $\widetilde{\mathbf{S}}_N(\boldsymbol{\theta}_{DF}^*, \boldsymbol{\theta}_D^*) = \mathbf{f}_N^D(\mathbf{S}_n(\boldsymbol{\theta}_{DF}^*); \boldsymbol{\theta}_D^*)$
4: $\widetilde{\mathbf{S}}_N = \text{reshape}(\widetilde{\mathbf{S}}_N)$
5: $\widetilde{\mathbf{S}}_h = \mathbf{V}\widetilde{\mathbf{S}}_N$
```

## 5.5 Numerical results

We assess the performance of the POD DL-ROM technique, by focusing on the training and testing computational times needed to construct and deploy a POD DL-ROM, and the use of pretraining, on three test cases: unsteady advection-diffusion-reaction, elastodynamics and unsteady Navier-Stokes equations for incompressible flows.

To evaluate the performance of the POD DL-ROM we rely on the two error indicators (3.13) and (3.14). The coefficient $\omega_h$ in (5.6) is set equal to 0.5 by following the results presented in Test 3 of Section 3.4. The rPOD dimension $N$ is selected, in all test cases, in such a way that $\epsilon_{rel}(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h) \approx 10^{-4}$, whereas the dimension of the nonlinear trial manifold $n$ is set trying to match the dimension of the solution manifold $n_\mu + 1$.

The POD DL-ROM neural network is implemented by means of the `Tensorflow` DL framework [Abadi et al., 2016].

### 5.5.1 Test 1: unsteady advection-diffusion-reaction equation

The first test case we consider deals with the solution of the following advection-diffusion-reaction system

$$\begin{cases} \dfrac{\partial u}{\partial t} - \operatorname{div}(\mu_1 \nabla u) + \mathbf{b}(\mu_2) \cdot \nabla u + cu = f(\mu_3, \mu_4) & (\mathbf{x}, t) \in \Omega \times (0, T), \\ \mu_1 \nabla u \cdot \mathbf{n} = 0 & (\mathbf{x}, t) \in \partial\Omega \times (0, T), \\ u(0) = 0 & \mathbf{x} \in \Omega, \end{cases} \tag{5.8}$$

in a two-dimensional square domain $\Omega = (0, 1)^2$, where

$$f(\mathbf{x}, t; \mu_3, \mu_4) = 10 \exp(-((x - \mu_3)^2 + (y - \mu_4)^2)/0.07^2),$$

and

$$\mathbf{b}(\mathbf{x}, t; \mu_2) = [\cos(\pi/\mu_2 t), \sin(\pi/\mu_2 t)]^T.$$

Here, the $n_\mu = 4$ parameters belong to the parameter space $\mathcal{P} = [0.002, 0.005] \times [30, 70] \times [0.4, 0.6]^2$. The equations have been discretized in space through linear finite elements by considering $N_h = 10657$ DOFs. For time integration, we use a BDF of order 2 by considering a time-step $\Delta t = 2\pi/20$ over $(0, T)$, with $T = 10\pi$. For different values of $\mu_3$ and $\mu_4$ the solution of (5.8) exhibits different patterns, due to the location of the distributed source; the dependence on $\mu_1$ and $\mu_2$ impact instead on the relative importance of diffusion and advection terms, and on the direction of this latter. We expect that for the case at hand POD-Galerkin ROMs might involve a large number of basis functions. Moreover, the dependence on $\mu_3$ and $\mu_4$ makes the problem nonaffine in the parameters.

The number of time instances is set to $N_t = 100$ and we consider $N_{train} = 5 \times 5 \times 5 \times 4 = 500$ training-parameter instances uniformly distributed in each parametric direction. For the testing phase, $N_{test} = 4 \times 4 \times 4 \times 3 = 192$ testing-parameter instances have been considered. The maximum number of epochs is set equal to $N_{epochs} = 10000$, the batch size is $N_b = 120$ and, regarding the early-stopping criterion, we stop the training if the loss function does not decrease within 500 epochs. We set $N = 64$ and the dimension of the reduced nonlinear trial manifold to $n = 5$, i.e. equal to $n_\mu + 1$. The training and testing phases of the POD DL-ROM neural network are performed on a Tesla V100 32GB GPU.

In Figure 5.4 we show the FOM and the POD DL-ROM solutions, for the testing-parameter instances $\boldsymbol{\mu}_{test} = (0.425, 0.425, 35, 0.0045)$ and $\boldsymbol{\mu}_{test} = (0.575, 0.475, 45, 0.0045)$ at $t = 29.53$, respectively, together with the relative error (3.14).

The comparison between some components of the intrinsic coordinates vector $\mathbf{V}^T\mathbf{u}_h$ and their POD DL-ROM approximation, for the testing parameter instance $\boldsymbol{\mu}_{test} = (0.575, 0.475, 45, 0.0045)$, is shown is Figure 5.5. We remark that, as expected, the first components are the

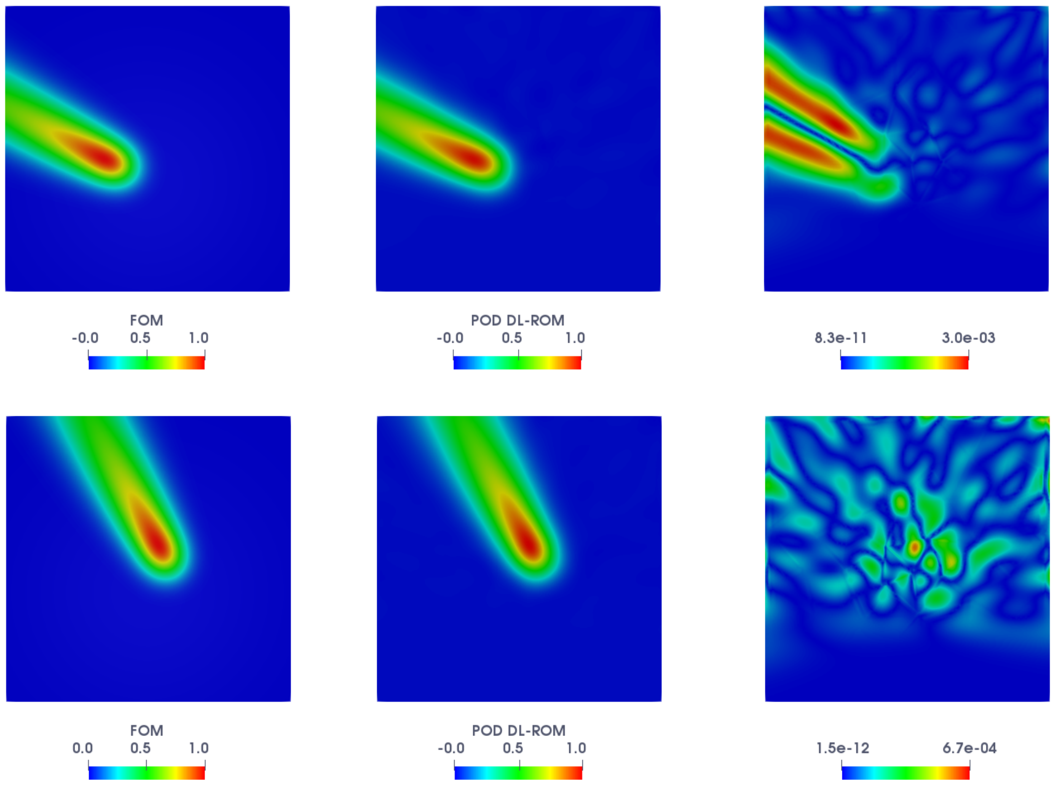


Figure 5.4: *Test 1*: FOM (left), POD DL-ROM (center), with $n = 5$ and $N = 64$, solutions and relative error $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instances $\boldsymbol{\mu}_{test} = (0.425, 0.425, 35, 0.0045)$ (top) and $\boldsymbol{\mu}_{test} = (0.575, 0.475, 45, 0.0045)$ (bottom) at $t = 29.53$.

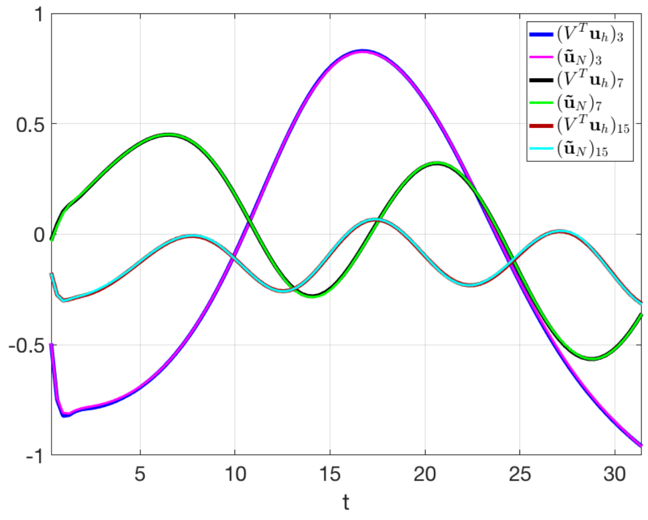


Figure 5.5: *Test 1*: Comparison between the intrinsic coordinates $\mathbf{V}^T\mathbf{u}_h$ components and the POD DL-ROM approximation $\tilde{\mathbf{u}}_N$ for the testing-parameter instance $\boldsymbol{\mu}_{test} = (0.575, 0.475, 45, 0.0045)$.

ones retaining most of the energy of the system; thus being the ones with higher magnitude

[Quarteroni et al., 2016].

In Figure 5.6 (left) we show the CPU computational times required by the exact SVD and the rSVD to compute the factorization and thus performing the first level of dimensionality reduction required by the POD DL-ROM with respect to $N$. The CPU time of the exact SVD does not change by varying the rPOD dimension, whereas the one of rSVD increases with respect to $N$. We remark that the two times are almost the same for $N = 4096$, a dimension for which constructing a ROM could in principle be avoided. We set $N = 64$ and, for this value, the use of rSVD allows a speed-up equal to 32 with respect to the exact SVD. The trend of the relative error (3.14) over time, for the selected testing-parameter instance $\boldsymbol{\mu}_{test} = (0.575, 0.475, 45, 0.0045)$, is depicted in Figure 5.6 (right). In particular, in Figure 5.6 (right) we show the mean (over the domain), the median, and the first and third quartile of the relative error, as well as its minimum. The interquartile range (IQR) shows that the distribution of the error is almost uniform over time.

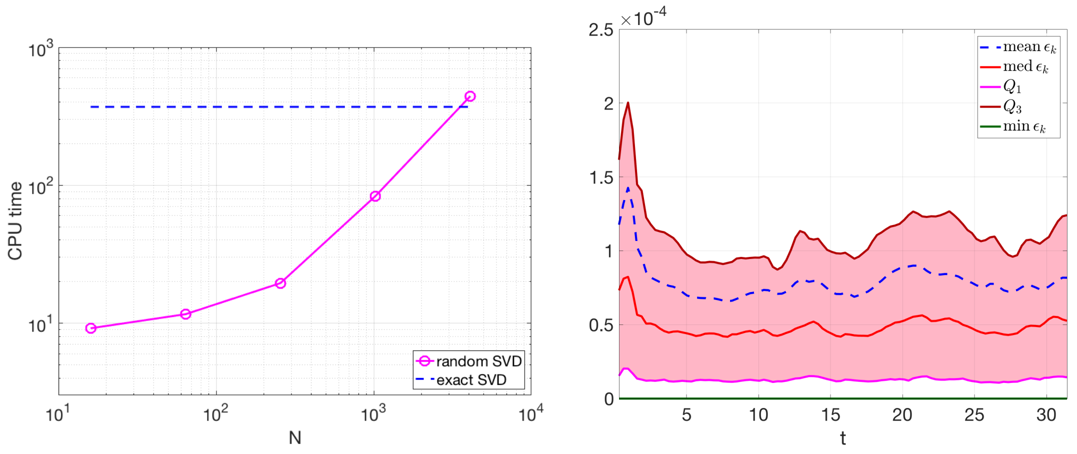


Figure 5.6: *Test 1*: Left: SVD and rSVD CPU times vs. $N$. Right: Trend of the relative error over time for $\boldsymbol{\mu}_{test} = (0.575, 0.475, 45, 0.0045)$.

In Figure 5.7 (left) we show the error indicator (3.13) computed on the FOM and POD DL-ROM solutions, the FOM and optimal-POD ones, and the intrinsic coordinates $\mathbf{V}^T\mathbf{u}_h$ and the approximated ones $\tilde{\mathbf{u}}_N$. The trend of $\epsilon_{rel}(\mathbf{u}_h, \tilde{\mathbf{u}}_h)$, for $N = 16$, is dictated by the projection error $\epsilon_{rel}(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h)$, meaning that, for $N = 16$, the rPOD dimension is too small to accurately reconstruct the FOM solution. For $N \geq 64$, the error indicator $\epsilon_{rel}(\mathbf{u}_h, \tilde{\mathbf{u}}_h)$ remains almost the same and this is related to the trend of $\epsilon_{rel}(\mathbf{V}^T\mathbf{u}_h, \tilde{\mathbf{u}}_N)$. As observed for the DL-ROM in Chapters 3 and 4, this behavior is related to the fact that an increase of $N$ only entails the addition of few parameters to the POD DL-ROM neural network, i.e. the approximation capability of the network remains almost the same while the input increases, thus resulting in a more difficult task.

The GPU training and testing computational times versus $N$ are pointed out in Figure 5.7 (right). The training time refers to the total time required for the training and validation phases; for the sake of completeness we also show the number of epochs $n_e$ along with $N$. The training time varies between 2 h 30 m and 4 h 40 m, we remark that we deal with $N_{train} = 500$ training-parameter instances. The testing time consists instead in the time required to compute $N_t$ time instances for a testing-parameter instance. The trend is proportional to $N^{1/2}$ and, for example, for $N = 64$ the testing time is equal to $4.2 \times 10^{-3}$ s.

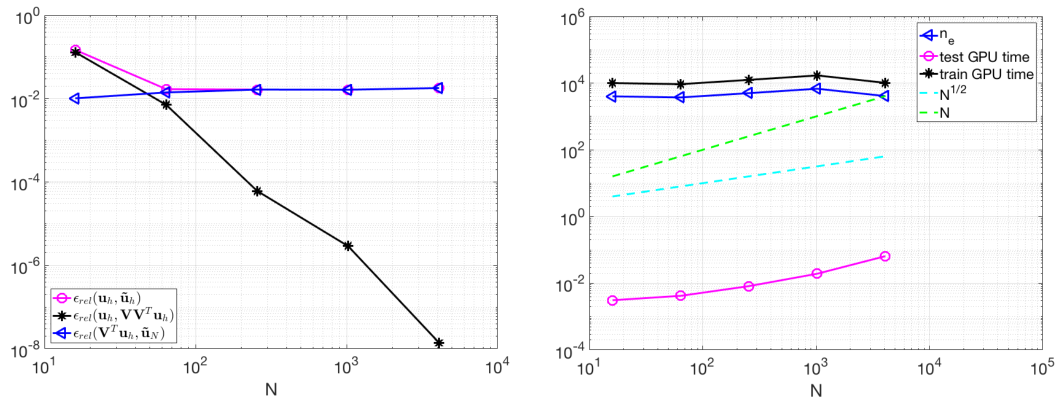


Figure 5.7: *Test 1*: Left: Error indicator $\epsilon_{rel}$ vs. $N$. Right: GPU training and testing computational times vs. $N$.

### 5.5.2 Test 2: elastodynamics equations

We now consider the solution of an elastodynamics problem, consisting of the following initial/boundary-value problem [Gurtin, 1982] for nonlinear elasticity equations, in a three-dimensional slab $\Omega = (0, 1)$ cm $\times(0, 5)$ cm $\times(0, 1)$ cm

$$\begin{cases} \rho \dfrac{\partial^2 \mathbf{d}}{\partial t^2} - \operatorname{div}(\mathbf{P}(\mathbf{d})) = \mathbf{f} & (\mathbf{x}, t) \in \Omega \times (0, T), \\ \mathbf{d} = \mathbf{0} & (\mathbf{x}, t) \in \Gamma_D \times (0, T), \\ \mathbf{P}(\mathbf{d})\mathbf{n} = \mathbf{0} & (\mathbf{x}, t) \in \Gamma_N \times (0, T), \\ \mathbf{d}(0) = \mathbf{0} & \mathbf{x} \in \Omega, \\ \dfrac{\partial \mathbf{d}}{\partial t}(0) = \mathbf{0} & \mathbf{x} \in \Omega. \end{cases} \tag{5.9}$$

Here we consider $\rho = 1$ kg/cm$^3$, $\mathbf{f} = (-0.01, 0, -0.02)$ kg/(cm s$^2$), $\Gamma_D = \{(x, z) \in (0, 1)^2, y = 0\}$ and $\Gamma_N = \partial\Omega \backslash \Gamma_D$. We consider a St. Venant-Kirchhoff constutive law involving a hyperelastic nonlinear model to describe the behavior of compressible materials [Ogden, 1997], characterized by the following strain energy function

$$\psi(\mathbf{F}) = \nu \mathbf{E} : \mathbf{E} + \frac{\lambda}{2}(\operatorname{tr}(\mathbf{E}))^2.$$

Here $\mathbf{E} = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}) = \frac{1}{2}(\mathbf{C} - \mathbf{I})$ is the Green-Lagrange strain tensor, $\nu$ and $\lambda$ are the Lamé coefficients and its first Piola-Kirchhoff stress tensor is given by

$$\mathbf{P}(\mathbf{F}) = \mathbf{F}\left(2\nu\mathbf{E} + \lambda \operatorname{tr}(\mathbf{E})\mathbf{I}\right).$$

The $n_\mu = 2$ parameters coincides with the Young modulus and the Poisson ratio, belonging to the parameter space $\mathcal{P} = [1, 3]$ Pa $\times$ $[0.25, 0.42]$, and they appear in the expression of the Lamé coefficients as follows

$$\nu = \frac{\mu_1}{2(1 + \mu_1)} \quad \text{and} \quad \lambda = \frac{\mu_1 \mu_2}{(1 + \mu_2)(1 - 2\mu_2)}.$$

Equations (5.9) are discretized in space by means of P2 finite elements thus generating a dynamical system of dimension $N_h = 5674 \times 3 = 17022$. For the time integration, we use the generalized-$\alpha$ method [Chung and Hulbert, 1993] over the interval $(0, T)$, with $T = 15$ s and a time-step $\Delta t = 0.2$ s.

We consider $N_t = 75$ time instances over $(0, T)$, $N_{train} = 10 \times 5 = 50$ training-parameter instances $\boldsymbol{\mu}_{train} = (1 + i2/9, 0.25 + j0.0425)$, for $i = 0, \dots, 9$ and $j = 0, \dots, 4$, and $N_{test} = 9 \times 4 = 36$ testing-parameter instances $\boldsymbol{\mu}_{test} = (1.111 + i2/9, 0.2712 + j0.0425)$, for $i = 0, \dots, 9$ and $j = 0, \dots, 4$. We set the rPOD dimension to $N = 64 \times 3 = 192$ and the dimension of the nonlinear trial manifold $\tilde{S}_n$ to $n = 3$, that is we match the dimension $n_\mu + 1$ of the solution manifold. The maximum number of epochs is $N_{epochs} = 10000$, the batch size is $N_b = 20$ and, regarding the early-stopping criterion, we stop the training if the loss function does not decrease along 500 epochs. The training and testing phases are performed on a GTX 1070 8GB GPU.

In Figure 5.8 we show the FOM solution and the POD DL-ROM one, with $n = 3$, along with the relative error (3.14), for the testing-parameter instance $\boldsymbol{\mu}_{test} = (2.88 \text{ Pa}, 0.3987)$ at $T = 15$ s. The maximum relative error, which is associated to the portion of the domain undergoing the maximum displacement, is about $10^{-3}$.

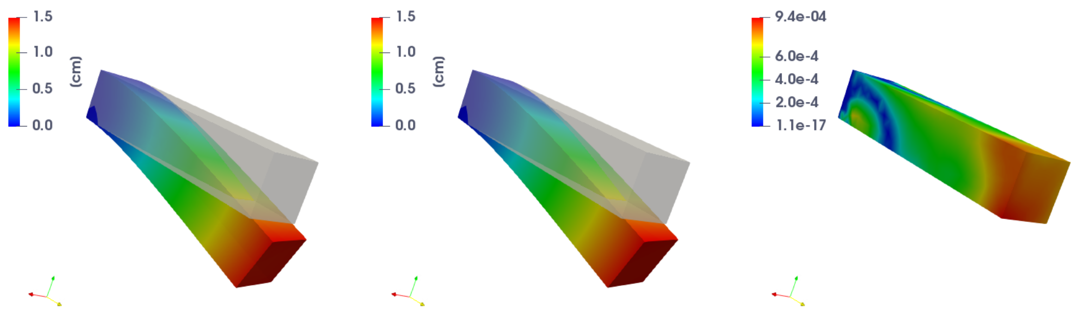


Figure 5.8: *Test 2*: FOM (left), POD DL-ROM (center) solutions and relative error $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instance $\boldsymbol{\mu}_{test} = (2.88 \text{ Pa}, 0.3987)$ at $T = 15$ s, with $n = 3$.

In Figure 5.9 we report the FOM and POD DL-ROM, where $n = 3$, $x$, $y$ and $z$ components of the vector displacement over the longitudinal axis, i.e. the line which connects the two points $P_1 = (0.5, 0, 0.5)$ cm and $P_2 = (0.5, 5, 0.5)$ cm, for the testing-parameter instance $\boldsymbol{\mu}_{test} = (2.88 \text{ Pa}, 0.3987)$ at $T = 15$ s. All the three components are accurately captured by the POD DL-ROM.

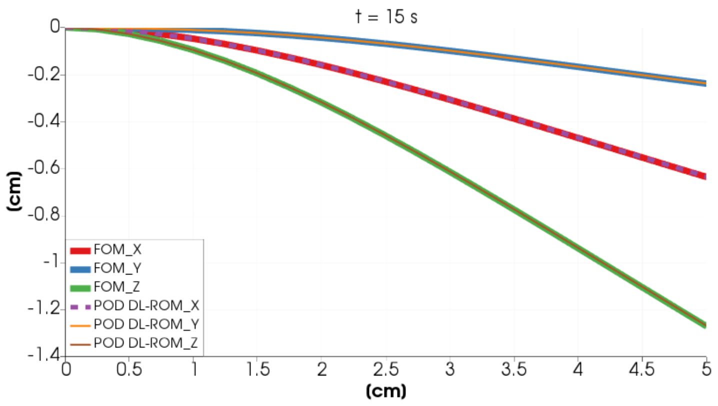


Figure 5.9: *Test 2*: FOM and POD DL-ROM solutions components over the longitudinal axis, for the testing-parameter instance $\boldsymbol{\mu}_{test} = (2.88 \text{ Pa}, 0.3987)$ at $T = 15$ s, with $n = 3$.

Here we also want to investigate how the use of pretraining, involving different fidelity models, impacts on the POD-DL-ROM technique, starting from the previous low-fidelity model. In particular, we consider the nearly-incompressible Neo-Hookean constitutive law [Ogden, 1997], an hyperelastic nonlinear model whose strain energy function is specified in terms of an isochoric-volumetric splitting

$$\psi(\mathbf{F}) = \underbrace{\frac{G}{2}\left(\bar{I}_1 - 3\right)}_{\psi_{\text{iso}}} + \underbrace{\frac{K}{4}\left((J-1)^2 + (\ln J)^2\right)}_{\psi_{\text{vol}}},$$

where $\bar{I}_1 = J^{-2/3} I_1 = J^{-2/3}\text{tr}(\mathbf{C})$, $J = \det(\mathbf{F})$, $G$ is the shear modulus and $K$ the bulk modulus. The coefficients $G$ and $K$ depend on the Young modulus and the Poisson coefficient and are defined as follow

$$G = \frac{\mu_1}{2(1+\lambda)} \quad \text{and} \quad K = \frac{2}{3}G + \lambda.$$

The $n_\mu = 2$ parameters belong to the parameter space $\mathcal{P} = [0.1, 1]\ \text{Pa} \times [0.3, 0.45]$. This time, the external stimulus takes the form $\mathbf{f} = (-0.001t, 0, -0.002t)$, the final time is $T = 22.5$ s, that is we enlarge the time interval, and the time-step is set equal to 0.25 s. We consider $N_t = 90$ time instances over $(0, T)$, $N_{train} = 50$ training-parameter instances and $N_{test} = 36$ testing-parameter instances uniformly distributed over the parameter space. We use the optimal weights ans biases found on the first low-fidelity model, as initial guess for the parameters of the POD DL-ROM on this second configuration.

In Figure 5.10 we show the FOM and DL-ROM solutions, with $n = 3$, together with the relative error (3.14), for the testing-parameter instances $\boldsymbol{\mu}_{test} = (0.25\ \text{Pa}, 0.32)$ and $\boldsymbol{\mu}_{test} = (0.95\ \text{Pa}, 0.43)$ at $T = 22.5$ s.

In Figure 5.11 we compare the FOM and POD DL-ROM, with $n = 3$, components of the vector displacement over the longitudinal axis for the testing-parameter instance $\boldsymbol{\mu}_{test} = (0.25\ \text{Pa}, 0.32)$ at $T = 22.5$ s.

In Table 5.1 we finally compare the GPU total and testing computational times of the POD DL-ROM neural network with and without the use of pretraining. In particular, the use of pretraining allows to strongly reduce the total training and validation time. The testing computational time, which refers to the time needed by the POD DL-ROM to compute $N_t = 90$ time instances for a testing-parameter instance, is equal to 0.006 s, and is remarkably lower than the final time $T = 22.5$ s, that is our technique is able to return even faster than real-time solutions.

| | #params | #epochs | total time | test [$s$] |
|---|---|---|---|---|
| POD DL-ROM | 270259 | 6490 | 63 m | 0.006 |
| POD DL-ROM PRETRAINED | 270259 | 1519 | 15 m | 0.006 |

Table 5.1: *Test 2*: GPU computational times of pretrained and from scratch POD DL-ROM.

### 5.5.3 Test 3: Navier-Stokes equations

We finally focus on the unsteady Navier-Stokes equations for incompressible flows in primitive variables (velocity and pressure) [Quarteroni and Valli, 1994]

$$\begin{cases} \rho\dfrac{\partial \mathbf{u}}{\partial t} + \rho\mathbf{u}\cdot\nabla\mathbf{u} - \nabla\cdot\boldsymbol{\sigma}(\mathbf{u}, p) = \mathbf{0} & (\mathbf{x}, t) \in \Omega \times (0, T), \\ \nabla\cdot\mathbf{u} = 0 & (\mathbf{x}, t) \in \Omega \times (0, T), \\ \mathbf{u} = \mathbf{0} & (\mathbf{x}, t) \in \Gamma_{D_1} \times (0, T), \\ \mathbf{u} = \mathbf{h} & (\mathbf{x}, t) \in \Gamma_{D_2} \times (0, T), \\ \boldsymbol{\sigma}(\mathbf{u}, p)\mathbf{n} = \mathbf{0} & (\mathbf{x}, t) \in \Gamma_N \times (0, T), \\ \mathbf{u}(0) = \mathbf{0} & \mathbf{x} \in \Omega, \end{cases} \tag{5.10}$$

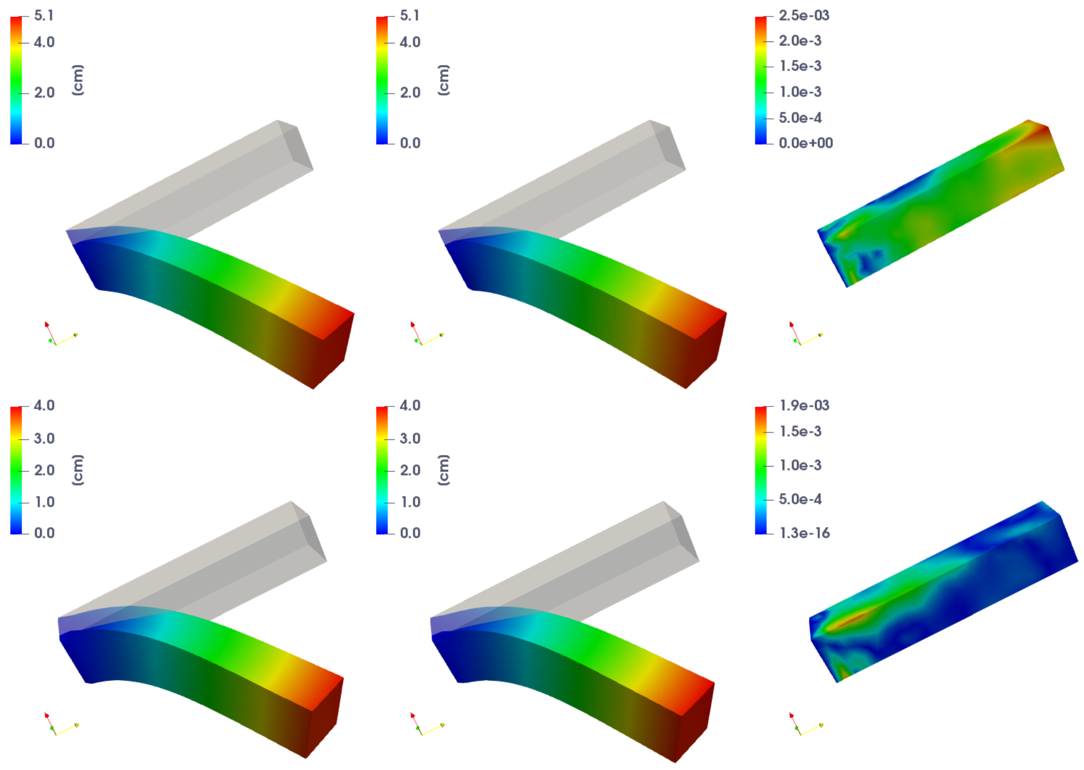


Figure 5.10: *Test 2*: FOM (left), POD DL-ROM (center) solutions and relative error $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instances $\boldsymbol{\mu}_{test} = (0.25 \text{ Pa}, 0.32)$ (top) and $\boldsymbol{\mu}_{test} = (0.95 \text{ Pa}, 0.43)$ (bottom) at $T = 22.5$ s, with $n = 3$.

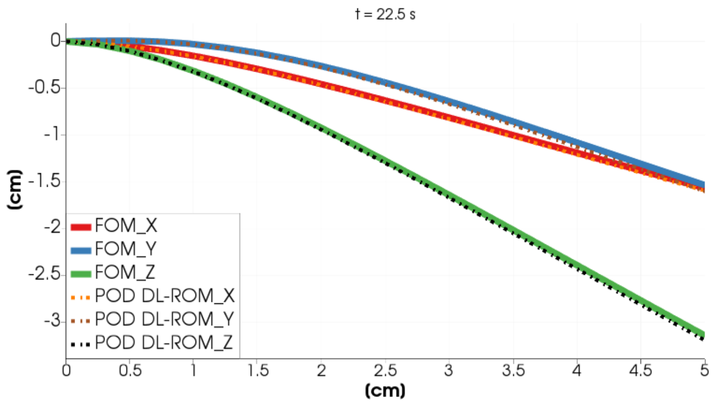


Figure 5.11: *Test 2*: FOM and POD DL-ROM solutions components over the longitudinal axis, for the testing-parameter instance $\boldsymbol{\mu}_{test} = (0.25 \text{ Pa}, 0.32)$ at $T = 22.5$ s, with $n = 3$.

where $\mathbf{u}$ is the fluid velocity, $p$ its pressure, $\mathbf{n}$ the (outward directed) normal unit vector to $\partial\Omega$, $\rho$ the fluid density and $\boldsymbol{\sigma}$ is the stress tensor defined as

$$\boldsymbol{\sigma}(\mathbf{u}, p) = -p\mathbf{I} + 2\nu\boldsymbol{\epsilon}(\mathbf{u}).$$

Here $\nu$ denotes the dynamic viscosity of the fluid, while the strain tensor is given by

$$\boldsymbol{\epsilon}(\mathbf{u}) = \frac{1}{2}\big(\nabla \mathbf{u} + \nabla \mathbf{u}^T\big).$$

We consider the flow around a cylinder test case, a well-known benchmark for the evaluation of numerical algorithms for incompressible Navier-Stokes equations in the laminar case. The domain consists in a two-dimensional pipe with a circular obstacle, i.e. $\Omega = (0, 2.2) \times (0, 0.41) \backslash \bar{B}_r(0.2, 0.2)$ with radius $r = 0.05$ (see Figure 5.12 for a sketch of the geometry). The density of the fluid is $\rho = 1$, on $\Gamma_{D_1} = \{x \in [0, 2.2], y = 0\} \cup \{x \in [0, 2.2], y = 0.41\} \cup \partial B_r(0.2, 0.2)$ no-slip boundary conditions are applied, while the outflow boundary is $\Gamma_N = \{x = 2.2, y \in [0, 0.41]\}$. On the boundary $\Gamma_{D_2}$ a parabolic inflow profile is prescribed

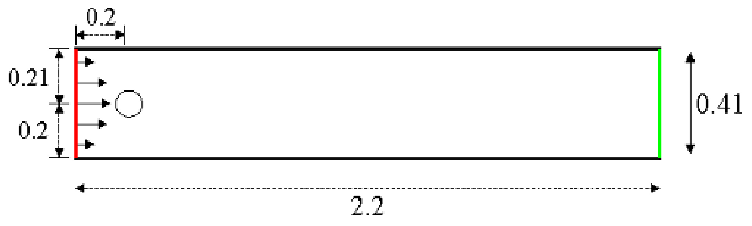


Figure 5.12: *Test 3*: Sketch of the geometry.

$$\mathbf{h}(\mathbf{x}, t; \mu) = \left( \frac{4U(t,\mu)y(0.41 - y)}{0.41^2}, 0 \right),$$

where $U(t;\mu) = \mu \sin(\pi t/8)$. We consider as parameter ($n_\mu = 1$) $\mu \in \mathcal{P} = [1, 2]$ which reflects on the Reynolds number varying in the range [66,133]. Equations (5.10) have been discretized in space by means of P2-P1 finite elements and in time through a BDF of order 2 with semi-implicit treatment of the convective term [Forti and Dedè, 2015] over the time interval $T = 8$ with a time-step $\Delta t = 2 \times 10^{-3}$.

We uniformly sample $N_t = 400$ time instances and consider $N_{train} = 11$ and $N_{test} = 10$ training- and testing-parameter instances uniformly distributed over $\mathcal{P}$. We are interested in reconstructing the velocity field, then the FOM dimension is equal to $N_h = 32446 \times 2 = 64892$, the dimension of the rPOD basis is $N = 256 \times 2 = 512$ and the one of the nonlinear trial manifold is $n = 2$. We highlight the possibility, by using POD DL-ROM, to reconstruct the field of interest, i.e the velocity $\mathbf{u}$, without the need of taking into account the approximation of the pressure $p$.

In Figure 5.13 we compare the FOM and POD DL-ROM solutions, the latter for $n = 2$, for two testing-parameter instances $\mu_{test} = 1.05$ (Re = 70) and $\mu_{test} = 1.75$ (Re = 117) at $t = 5.64$. We highlight the strong variability of the solution over the parameter space $\mathcal{P}$; indeed in Figure 5.13 (top) we do not assist to vortex shedding whereas in Figure 5.13 (bottom) it is present. Hence, the POD DL-ROM approximation is able to accurately capture this remarkable variability.

The computational training and testing time of the POD DL-ROM neural network on a Tesla V100 32 GB GPU are equal to 50 minutes and 0.1 seconds, respectively.

We remark that ensuring ROM stability in the classical POD-Galerkin framework usually requires additional computational efforts, such as a suitable enrichment of the velocity reduced basis, and a consequent increase of the size of the ROM; see, e.g., [Dal Santo et al., 2019, Ballarin et al., 2015, Rozza and Veroy, 2007].

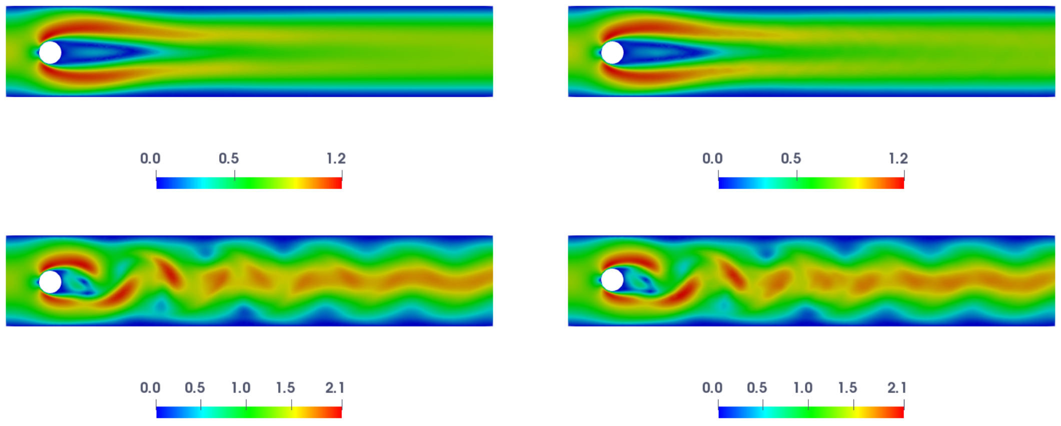


Figure 5.13: *Test 3*: FOM (left) and POD DL-ROM (right) solutions for the testing-parameter instances $\mu_{test} = 1.05$ (top) and $\mu_{test} = 1.75$ (bottom) at $t = 5.64$, with $n = 2$.

## 5.6 Discussion

In this Chapter, we proposed a strategy to enhance DL-ROMs in order to make the offline training stage dramatically faster. This strategy, which we refer to as POD DL-ROM, overcomes the main computational bottleneck of the DL-ROM technique, namely the (strong) limitation related to the FOM dimension $N_h$. In particular, it exploits dimensionality reduction of FOM snapshots by means of randomized POD (rSVD) and a suitable multi-fidelity pretraining stage. Moreover, the POD DL-ROM approximations retain all the features of DL-ROM solutions enabling extremely efficient testing computational times. Through the numerical test cases assessed in this Chapter, POD DL-ROMs have shown to yield extremely efficient numerical approximations to (scalar and vector) nonlinear time-dependent parametrized PDEs, ultimately leading to the possibility to solve in more than real-time, during the online testing stage, parametrized PDEs modeling physical phenomena whose time scale is seconds.

Chapter 6

# POD DL-ROMs for cardiac electrophysiology (I): benchmark problems

In this Chapter we assess the performance of the POD DL-ROM technique, proposed in Chapter 5, on some relevant numerical benchmark test cases in cardiac electrophysiology (EP), both in physiological and pathological conditions, similarly to what we did in Chapter 4 for the DL-ROM technique. We reconsider some numerical experiments already presented in Section 4.3 to compare the increased efficiency entailed by the use of POD DL-ROM with respect to the case of the DL-ROM and the POD-Galerkin ROM. Regarding the reduced order model (ROM) construction, we investigate the use of pretraining in three scenarios: *(i)* increasing the full order model (FOM) dimension, *(ii)* enlarging the dimension of the parameter space $\mathcal{P}$ and *(iii)* varying the geometry of the problem. Moreover, we apply the POD DL-ROM technique to problems where the FOM has been obtained through either the finite element (FE) method or NURBS-based Isogeometric Analysis (IGA).

## 6.1 Moving from DL-ROM to POD DL-ROM

The numerical results presented in Section 4.3 show that the resulting DL-ROM technique, formerly introduced in Chapter 3, allows one to accurately capture complex fronts propagation processes, both in physiological and pathological scenarios.

The proposed DL-ROM can efficiently provide solutions to parametrized cardiac EP problems, thus enabling multi-scenario analyses in pathological cases, especially if compared to common linear (projection-based) ROMs, built for instance through a POD-Galerkin RB method. In particular, the benefits introduced by the use of DL-ROM, in cardiac EP problems, can be summarized as follows:

- the dimension of the DL-ROM can be kept extremely small, very close (or even equal) to the dimension of the solution manifold $n_\mu + 1$;
- the DL-ROM can be queried at any desired time instant, without requiring the solution of a dynamical system until that time, differently from projection-based ROMs such as, e.g, POD-Galerkin ROMs;
- the time resolution required by the DL-ROM can be chosen to be larger than the one required by the numerical solution of dynamical systems in cardiac EP;

- DL-ROMs avoid the use of (intrusive and very often extremely expensive) hyper-reduction techniques, which are required by POD-Galerkin ROMs *(i)* to handle efficiently those terms that depend nonlinearly on either the transmembrane potential or the input parameters, and *(ii)* to account for the dynamics of the gating variables. Indeed, DL-ROMs allow us to approximate the electric potential without the need of computing the gating variables.

For all these reasons, DL-ROMs completely avoid both the assembly and the projection stages typical of traditional projection-based ROMs, thus substantially improving computational efficiency.

However, the DL-ROM depends on $N_h$, which ultimately might entail long training computational times for large FOM dimension, as the ones entailed by cardiac EP problems. The POD DL-ROM technique introduced in Chapter 5, thanks to a prior dimensionality reduction relying on POD and a suitable multi-fidelity pretraining, greatly enhances the efficiency of the DL-ROM during the training phase, thus dramatically decreasing training computational times, as shown by the numerical results discussed in this Chapter.

## 6.2 Test 1: Two-dimensional slab

In this Section we are interested in the approximation of the solution of the problem presented in Subsection 4.3.1 by means of POD DL-ROM. We recall that the parameters ($n_\mu = 2$) consist of the longitudinal and transversal conductivities to the fibers direction $\mathbf{f}_0 = (1, 0)^T$.

In Figure 6.1 we report the FOM and POD DL-ROM solutions, the latter obtained by setting $n = 3$ and $N = 64$, along with the relative error (3.14), for the testing-parameter instance $\boldsymbol{\mu}_{test} = 12.9 \cdot (0.1825, 0.0912)$ cm$^2$/ms at $t = 47.7$ ms (top) and $t = 379.7$ ms (bottom).

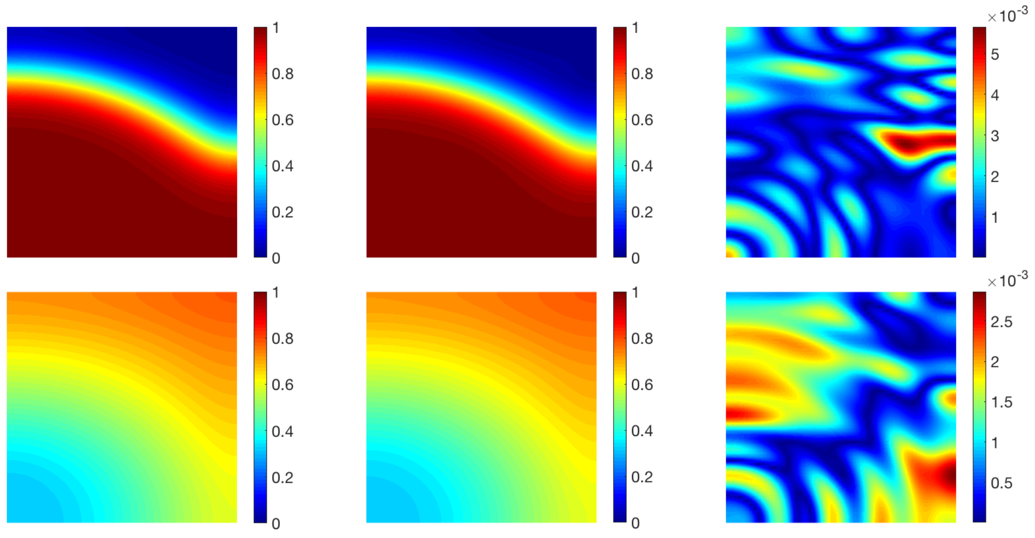


Figure 6.1: *Test 1*: FOM (left), POD DL-ROM (center) solutions and relative error $\boldsymbol{\epsilon}_k$ (right) for the testing-parameter instance $\boldsymbol{\mu}_{test} = 12.9 \cdot (0.1825, 0.0912)$ cm$^2$/ms at $t = 47.7$ ms and $t = 379.7$ ms, with $n = 3$.

The comparison among the DL-ROM and POD DL-ROM neural networks number of parameters, the GPU training and validation time for one epoch, the total number of epochs, the total GPU training and validation (total time) and testing GPU computational times are reported in Table 6.1. We also report the total CPU offline and online computational times[1] required by the POD-Galerkin ROM with $N_c = 6$ clusters, which corresponds to the choice providing the most efficient results (see Subection 4.3.1), by keeping for all the models the same degree of accuracy $\epsilon_{rel} = 4.03 \times 10^{-3}$ and by following the approach running the code on the hardware it is optimized for. The use of POD DL-ROM not only leads us to even faster testing computational times with respect to DL-ROM, due to the remarkable reduction of the number of parameters of the neural network, but also reduces both the DL-ROM and the POD-Galerkin ROM total times of a factor 37.5 and 5, respectively. Then, POD DL-ROM etablishes to be the most efficient ROM both at training and testing stages.

| | #params | train - val [$s/epoch$] | #epochs | total time | test [$s$] |
|---|---|---|---|---|---|
| DL-ROM (GPU) | 2342595 | 7.5 - 0.8 | 6981 | 15 h | 0.08 |
| POD DL-ROM (GPU) | 269057 | 1.5 - 0.15 | 866 | 24 m | 0.015 |
| POD-Galerkin ROM ($N_c = 6$) | - | - | - | 115 m | 8 |

Table 6.1: *Test 1*: DL-ROM, POD DL-ROM and POD-Galerkin ROM computational times.

We now investigate the use of pretraining, multi-fidelity approach aimed at further decreasing training times, in two different scenarios:

- increasing the FOM dimension $N_h$;
- increasing the dimension of the parameter space $\mathcal{P}$.

First, we use the optimal parameters, weights and biases, of the POD DL-ROM neural network found in the case $N_h = 4096$, to initialize the POD DL-ROM neural network parameters associated to $N_h = 128 \times 128 = 16384$ and $N_h = 256 \times 256 = 65536$, by fixing $N = 64$ for a prescribed degree of accuracy $\epsilon_{rel} = 4.03 \times 10^{-3}$. The use of pretraining is possible in this framework because the POD DL-ROM neural network does not depend on $N_h$, but only on $N$. Pretraining then allows to drastically reduce the GPU training computational times as shown in Table 6.2, where we compare the training times, in presence of pretraining, with the ones of the network trained from scratch. The testing computational times increase with respect to the case $N_h = 4096$ due to the matrix multiplication $\mathbf{V}\mathbf{u}_N$, since $\mathbf{V}$ has a higher number of rows.

| | #params | #epochs | total time | test [$s$] |
|---|---|---|---|---|
| POD DL-ROM ($N_h = 16384$) | 269057 | 1378 | 38 m | 0.06 |
| POD DL-ROM PRETRAINED ($N_h = 16384$) | 269057 | 165 | 5 m | 0.06 |
| POD DL-ROM ($N_h = 65536$) | 269057 | 1540 | 42 m | 3 |
| POD DL-ROM PRETRAINED ($N_h = 65536$) | 269057 | 461 | 12 m | 3 |

Table 6.2: *Test 1*: GPU computational times of pretrained and from scratch POD DL-ROM for $N_h = 16384$ and 65536.

Then, we report the results referring to a larger parameter space by setting $N_h = 4096$ and $N = 64$. In particular, we use the optimal parameters associated to $\mathcal{P} = 12.9 \cdot [0.06, 0.2] \times 12.9 \cdot [0.03, 0.1]$ cm$^2$/ms as initial guess of the POD DL-ROM neural network parameters in the case $\mathcal{P} = 12.9 \cdot [0.02, 0.2] \times 12.9 \cdot [0.01, 0.1]$ cm$^2$/ms. We show the GPU training computational times in Table 6.3 for a prescribed level of accuracy, i.e. $\epsilon_{rel} = 4.03 \times 10^{-3}$,

[1]Here we employ a full 64 GB node (20 Intel® Xeon® E5-2640 v4 2.4GHz cores) of the HPC cluster available at MOX, Politecnico di Milano.

and a fixed number of training-parameter instances $N_{train} = 25$, using pretraining and not, respectively. By means of pretraining, the total time is remarkably decreased also in this case.

| | #params | #epochs | total time | test $[s]$ |
|---|---|---|---|---|
| POD DL-ROM | 269057 | 1486 | 41 m | 0.015 |
| POD DL-ROM PRETRAINED | 269057 | 588 | 16 m | 0.015 |

Table 6.3: *Test 1*: GPU computational times of pretrained and from scratch POD DL-ROM for $\mathcal{P} = 12.9 \cdot [0.02, 0.2] \times 12.9 \cdot [0.01, 0.1]$ cm$^2$/ms.

## 6.3 Test 2: Two-dimensional slab with ischemic region

In this Section we focus on the application of the POD DL-ROM technique on a two-dimensional slab including an ischemic region depending on a set of input parameters, namely the coordinates of the center of the scar belonging to the parameter space $\mathcal{P} = [3.5, 6.5 \text{ cm}]^2$. The set-up of the FOM (apart from the FOM dimension equal to $N_h = 128 \times 128 = 16384$ in this case), the parameter space, the number of training- and testing-parameter instances, the number of time instances and the hyperparameters of the neural network, are the same as the ones provided in Test 3 of section 4.3.

In Figure 6.2 we show the FOM and the POD DL-ROM solutions, obtained by setting the dimension of the reduced trial manifolds equal to $n = 3$ and $N = 256$, along with the relative error $\boldsymbol{\epsilon}_k$, defined in (3.14), for the testing-parameter instance $\boldsymbol{\mu}_{test} = (6.25, 6.25)$ cm at $\tilde{t} = 94$ ms, and for the same level of accuracy $\epsilon_{rel}$ achieved by DL-ROM on this test case. Even if the high gradients of the solution around the ischemic region are not sharply reproduced, the POD DL-ROM approximation is able to capture the location and the diseased nature of this portion of tissue.

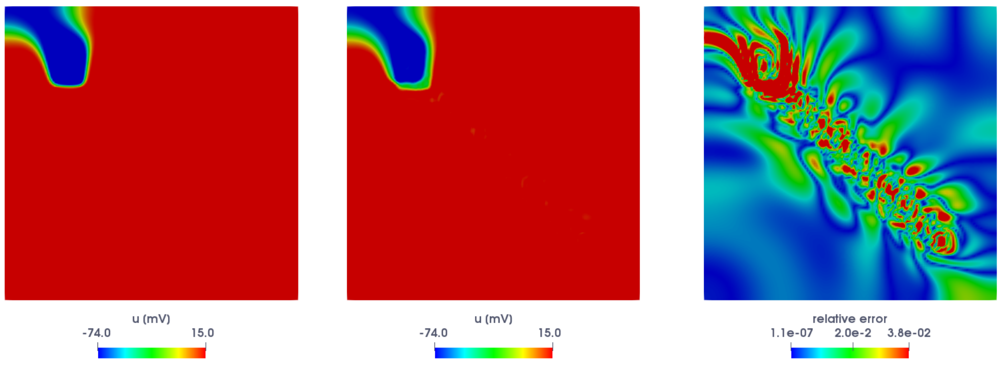


Figure 6.2: *Test 2*: FOM (left) and POD DL-ROM (center) solutions, with $n = 3$ and $N = 256$, and relative error $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instance $\boldsymbol{\mu}_{test} = (6.25, 6.25)$ cm at $\tilde{t} = 94$ ms.

In Table 6.4 we compare the performance of the DL-ROM, the POD DL-ROM and the POD-Galerkin ROM with $N_c = 6$ clusters[2], reporting the total and testing computational times for a prescribed level of accuracy. The DL-based ROM techniques provide more efficient ROMs, if compared to the POD-Galerkin ROM, at testing time. Moreover, the use of POD DL-ROM, relying on a first data dimensionality reduction of a factor $N_h/N = 64$, drastically

[2]The simulation is performed on 10 cores of 1.7 TB node (192 Intel® Xeon Platinum® 8160 2.1GHz cores) of the HPC cluster available at MOX, Politecnico di Milano.

reduces the total training and validation time. Once again, POD DL-ROM provides the most efficient ROM both at training and testing time.

| | #params | #epochs | total time | test $[s]$ |
|---|---|---|---|---|
| DL-ROM (GPU) | 8645765 | 7949 | 90 h | 0.35 |
| POD DL-ROM (GPU) | 293595 | 560 | 35 m | 0.01 |
| POD-Galerkin ROM ($N_c = 6$) | - | - | 68 h | 27 |

Table 6.4: *Test 2*: DL-ROM, POD DL-ROM and POD-Galerkin ROM computational times.

## 6.4 Test 3: Two-dimensional slab with figure of eight re-entry

In this Section we apply the POD DL-ROM technique to Test 4 of Section 4.3, namely the figure of eight re-entry test case where the parameter is the $y$-coordinate of the center of the second applied stimulus. The set-up of the FOM, the number of training- and testing-parameter instances, the number of time instances and the hyperparameters of the neural network are the same provided in Test 4 of Section 4.3. The parameter space is given by $\mathcal{P} = [0.8, 1.1]$ cm.

In Figure 6.3 we report the FOM and the POD DL-ROM solutions, the latter with $n = 5$ and $N = 1024$, along with the relative error $\boldsymbol{\epsilon}_k$, for the testing-parameter instance $\mu_{test} = 0.9125$ cm at $\tilde{t} = 147$ ms, corresponding to almost the same level of accuracy $\epsilon_{rel}$ achieved by a DL-ROM on this problem.

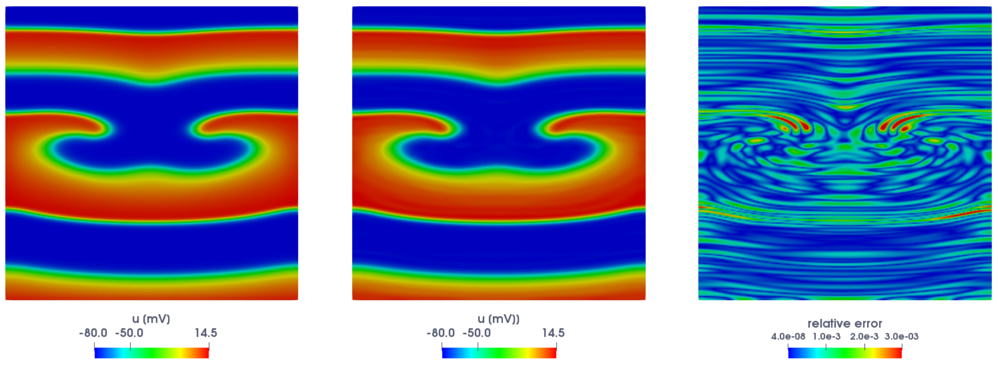


Figure 6.3: *Test 3*: FOM (left) and POD DL-ROM (center) solutions, with $n = 5$ and $N = 1024$, and relative error $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instance $\mu_{test} = 0.9125$ cm at $\tilde{t} = 147$ ms.

The comparison among the DL-ROM, the POD DL-ROM and the POD-Galerkin ROM with $N_c = 4$ clusters[3] training and testing computational times obtained by keeping the same degree of accuracy for the three models, is provided in Table 6.5. The use of POD DL-ROM introduces a first level of dimensionality reduction, by means of the rSVD, equal to $N_h/N = 64$, which reflects in the striking reduction of the total training and validation time with respect to the ones of the DL-ROM and the POD-Galerkin ROM. Finally, we remark that the POD DL-ROM technique results to be the most efficient both at training and testing time.

[3]Tests are performed on a full 64 GB node (20 Intel® Xeon® E5-2640 v4 2.4GHz cores) of our in-house HPC cluster.

|  | #params | #epochs | total time | test $[s]$ |
|---|---|---|---|---|
| DL-ROM (GPU) | 33891843 | 5633 | 64 h | 0.6 |
| POD DL-ROM (GPU) | 395907 | 12738 | 138 m | 0.4 |
| POD-Galerkin ROM ($N_c = 4$) | - | - | 238 m | 33 |

Table 6.5: *Test 3*: DL-ROM, POD DL-ROM and POD-Galerkin ROM computational times.

## 6.5 Test 4: Two-dimensional rectangular slab

We now focus on the analysis of the POD DL-ROM performance on a rectangular slab, by varying the longitudinal conductivity to the fibers direction, when applied to a problem involving a FOM obtained by means of NURBS-based IGA. Indeed, as pointed out in Chapter 1, smooth NURBS basis functions control and limit numerical dispersion thus accurately capturing steep fronts. Few attempts have been made in order to build ROMs starting from IGA snapshots, see, e.g., [Salmoiraghi et al., 2016, Garotta et al., 2020, Rinaldi et al., 2015, Haghighat et al., 2020]. As outlined in Subsections 1.3.2 and 1.4.1, regular NURBS basis functions allow a smooth representation of the computational domain. An attempt in this direction is provided by [Tencer and Potter, 2020] where the authors propose a nonlinear manifold learning technique based on deep AEs for reduced order modeling of physical systems on unstructured meshes. Even if still introducing a level of approximation in the construction of the computational domain, unstructured meshes are characterized by a lower error of approximation with respect to the structured ones on complex geometries. The smoothness of the geometry is instead completely maintained by the use of NURBS basis functions.

In the following we will denote by $N_h$ the dimension of the finite-dimensional space $X_h$ spanned by NURBS basis functions, previously indicated with $n$ in Chapter 1, in order to standardize the discussion.

We consider the two-dimensional coupled PDE-ODE nonlinear system consisting in the Bidomain equation (1.1) coupled with the R-M ionic model (1.6) in a rectangular slab of cardiac tissue $\Omega = (0, 10)$ cm $\times$ $(0, 2)$ cm. The parameter ($n_\mu = 1$) consists in the electric intracellular conductivity in the longitudinal direction to the fibers, i.e., the conductivity tensor $\mathbf{D}_i(\mathbf{x}; \boldsymbol{\mu})$ takes the form

$$\mathbf{D}_i(\mathbf{x}; \boldsymbol{\mu}) = \sigma_t^i I + (\mu - \sigma_t^i)\mathbf{f}_0 \otimes \mathbf{f}_0,$$

where $\mathbf{f}_0 = (1, 0)^T$ and the parameter space is $\mathcal{P} = 2.3 \cdot [10^{-4}, 10^{-3}]\ \Omega^{-1}\text{cm}^{-1}$. The remaining intra- and extracellular conductivities are set equal to $\sigma_t^i = 2.4 \times 10^{-4}\ \Omega^{-1}\text{cm}^{-1}$, $\sigma_l^e = 1.5 \times 10^{-3}\ \Omega^{-1}\text{cm}^{-1}$ and $\sigma_t^e = 1 \times 10^{-3}\ \Omega^{-1}\text{cm}^{-1}$, respectively. The parameters of the R-M ionic model are given by $u_{th} = 13$ mV, $v_p = 100$ mV, $G = 1.5\ \text{ms}^{-1}$, $\eta_1 = 4.4\ \text{ms}^{-1}$, $\eta_2 = 1.2 \times 10^{-2}$ and $\eta_3 = 1$, see, e.g., [Gerardo-Giorda, 2007]. Aiming at investigating the effect of different spatial discretizations on the performance of POD DL-ROM, we provide snapshots approximated by means of different polynomials orders and global order continuities. In particular, we consider two discretization settings: P1/C0 NURBS basis functions, by considering a FOM dimension $N_h = 161 \times 33 = 5313$, and P2/C1 NURBS basis functions, where $N_h = 165 \times 35 = 5705$, with the same number of mesh elements $n_{el} = 5120$. Time integration is performed over the interval $(0, T)$, with $T = 150$ ms and a time-step $\Delta t = 0.05$ ms, through the BDF of order 2. The intracellular applied current takes the form

$$I_{app}^i(\mathbf{x}, t) = C\mathbf{1}_{\Omega_{app}}(\mathbf{x})\mathbf{1}_{[t^i, t^f]}(t), \tag{6.1}$$

where $C = 100$ mA, $\Omega_{app} = \{\mathbf{x} \in \Omega : x \leq 0.2\}$, $t^i = 0$ ms and $t^f = 1$ ms.

For the training phase, we uniformly sample $N_t = 1500$ time instances in the interval $(0, T)$ and consider $N_{train} = 11$ training-parameter instances uniformly distributed in the parameter space. For the testing phase, $N_{test} = 10$ testing-parameter instances have been considered, each of them corresponding to the midpoint of two consecutive training-parameter instances. The maximum number of epochs is $N_{epochs} = 30000$, the batch size is $N_b = 40$ and, regarding the early-stopping criterion, we stop the training if the loss function does not decrease along 1000 epochs.

In Figure 6.4 we report the FOM and POD DL-ROM solutions, the latter with $n = 2$ and $N = 256$, together with the relative error (3.14), arising from P1/C0 and P3/C2 NURBS-based IGA for the testing-parameter instance $\mu_{test} = 0.0199\ \Omega^{-1}\text{cm}^{-1}$ at $t = 109$ ms.

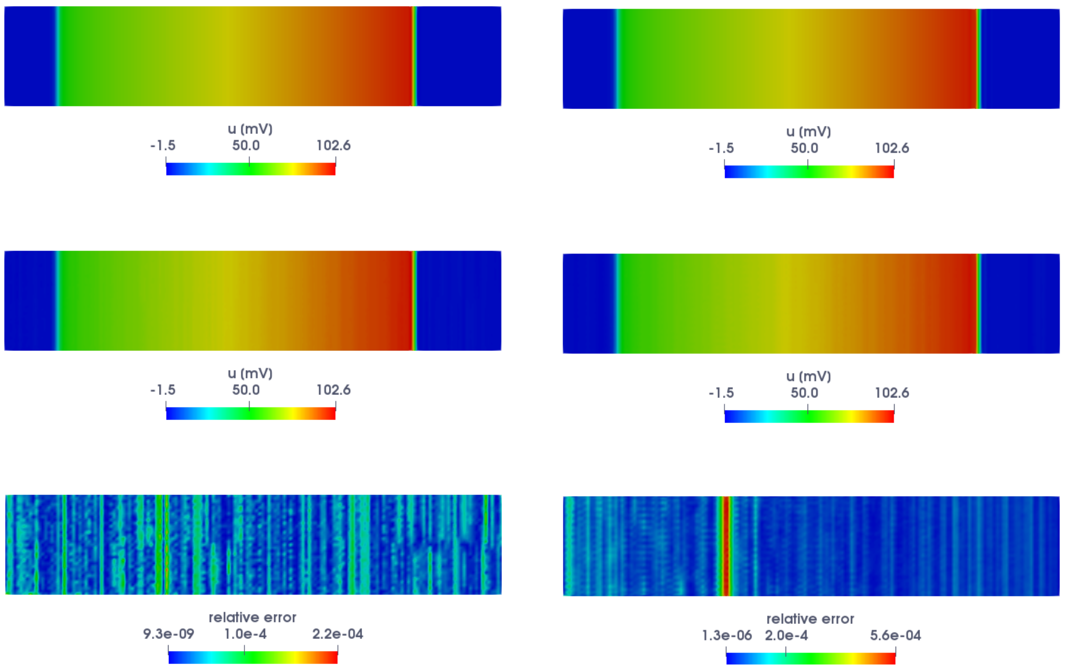


Figure 6.4: *Test 4*: FOM solution (top), POD DL-ROM one (center), with $n = 2$ and $N = 256$, and $\boldsymbol{\epsilon}_k$ (bottom), obtained by means of P1/C0 (left) and P3/C2 (right) NURBS basis functions, for the testing-parameter instance $\mu_{test} = 0.0199\ \Omega^{-1}\text{cm}^{-1}$ at $t = 109$ ms.

In Tables 6.6 and 6.7 we compare the reconstruction error $\epsilon_{rel}(\mathbf{u}_h, \tilde{\mathbf{u}}_h)$, the error between the intrinsic coordinates and the approximated ones $\epsilon_{rel}(\mathbf{V}^T\mathbf{u}_h, \tilde{\mathbf{u}}_N)$, the projection error $\epsilon_{rel}(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h)$ and the number of epochs $n_e$, by varying $N$, both for P1/C0 and P3/C2 basis functions. We use a k-fold cross-validation strategy [Goodfellow et al., 2016], with $k = 5$, in order to decrease the variance of the generalization error, i.e. how much an estimator varies as function of data sampling, and mitigate the effect of randomization intrinsic to the neural network.

By comparing the P1/C0 and the P2/C3 errors between the FOM solution and the optimal-POD one, that is the projection of the FOM solution onto the linear subspace generated through rPOD, and by varying $N$, it seems that the use of P1/C0 leads to a slightly better accuracy. In other words, in the case of P3/C2 NURBS a higher number of modes is required. A similar analysis can be found in [Zhu et al., 2017a, Zhu et al., 2017b] where the decay of POD eigenvalues of the snapshot matrices obtained by means of P2/C1 NURBS-based IGA, P2/C0 NURBS-based IGA and P2 FE method, are compared, however, on much simpler problems with respect to the one treated here. We highlight that the projection error limits the recostruction error for $N = 16, 64$ both for P1/C0 and P3/C2 basis functions whereas the error $\epsilon_{rel}(\mathbf{V}^T\mathbf{u}_h, \tilde{\mathbf{u}}_N)$ increases with $N$ due to the fact that the dimension of the input, and of the output, is higher and the reconstruction task is more complex. The latter determines the reconstruction error for $N = 256$. In particular, P3/C2 basis functions achieve a smaller reconstruction error, determined by $\epsilon_{rel}(\mathbf{V}^T\mathbf{u}_h, \tilde{\mathbf{u}}_N)$. The slightly higher accuracy on the error between the intrinsic coordinates and the approximated ones is achieved through a higher number of epochs, meaning that P3/C2 NURBS basis functions are less subject to overfitting than P1/C0; indeed all the results shown here are obtained by fulfilling the early-stopping criterion during the training.

| | $\epsilon_{rel}(\mathbf{V}^T\mathbf{u}_h, \tilde{\mathbf{u}}_N)$ | $\epsilon_{rel}(\mathbf{u}_h, \tilde{\mathbf{u}}_h)$ | $\epsilon_{rel}(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h)$ | $n_e$ |
|---|---|---|---|---|
| $N = 16$ | $3.3 \times 10^{-3}$ | $1.2 \times 10^{-1}$ | $1.2 \times 10^{-1}$ | 6328 |
| $N = 64$ | $4.3 \times 10^{-3}$ | $4.3 \times 10^{-2}$ | $4.3 \times 10^{-2}$ | 9003 |
| $N = 256$ | $7.9 \times 10^{-3}$ | $7.9 \times 10^{-3}$ | $2.7 \times 10^{-14}$ | 11616 |

Table 6.6: *Test 4*: P1/C0 error indicators $\epsilon_{rel}(\mathbf{V}^T\mathbf{u}_h, \tilde{\mathbf{u}}_N)$, $\epsilon_{rel}(\mathbf{u}_h, \tilde{\mathbf{u}}_h)$, $\epsilon_{rel}(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h)$ and number of epochs vs. $N$.

| | $\epsilon_{rel}(\mathbf{V}^T\mathbf{u}_h, \tilde{\mathbf{u}}_N)$ | $\epsilon_{rel}(\mathbf{u}_h, \tilde{\mathbf{u}}_h)$ | $\epsilon_{rel}(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h)$ | $n_e$ |
|---|---|---|---|---|
| $N = 16$ | $2 \times 10^{-3}$ | $1.3 \times 10^{-1}$ | $1.3 \times 10^{-1}$ | 8334 |
| $N = 64$ | $2.7 \times 10^{-3}$ | $6.5 \times 10^{-2}$ | $6.5 \times 10^{-2}$ | 10241 |
| $N = 256$ | $7.3 \times 10^{-3}$ | $7.3 \times 10^{-3}$ | $1.1 \times 10^{-13}$ | 13170 |

Table 6.7: *Test 4*: P3/C2 error indicators $\epsilon_{rel}(\mathbf{V}^T\mathbf{u}_h, \tilde{\mathbf{u}}_N)$, $\epsilon_{rel}(\mathbf{u}_h, \tilde{\mathbf{u}}_h)$, $\epsilon_{rel}(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h)$ and number of epochs vs. $N$.

In Tables 6.8 and 6.9 we show the training and validation time for one epoch, the total time and the testing time for P1/C0 and P3/C2 NURBS basis functions, respectively, obtained by running the code on a Tesla V100 32GB GPU. Both for P1/C0 and P3/C2 the testing computational time increases with $N$. The P1/C0 testing times are slightly smaller than the P3/C2 ones due to the fact that the matrix $\mathbf{V}$ has a lower number of rows, because of the FOM dimension $N_h$. The training and validation times per epoch are exactly the same in the two cases, depending the POD DL-ROM neural network only on $N$. The P3/C2 total times are slightly higher than the ones associated to P1/C0 NURBS basis functions due to the higher number of epochs. At testing time, it is possible to achieve very efficient results both for P1/C0 and P3/C2 basis functions.

| | train - val $[s/epoch]$ | total time | test $[s]$ |
|---|---|---|---|
| $N = 16$ | 1.15 | 105 m | 0.021 |
| $N = 64$ | 1.25 | 163 m | 0.029 |
| $N = 256$ | 1.4 | 271 m | 0.053 |

Table 6.8: *Test 4*: Computational times for P1/C0 NURBS basis functions vs. $N$.

| | train - val $[s/epoch]$ | total time | test $[s]$ |
|---|---|---|---|
| $N = 16$ | 1.15 | 138 m | 0.022 |
| $N = 64$ | 1.25 | 186 m | 0.032 |
| $N = 256$ | 1.4 | 307 m | 0.053 |

Table 6.9: *Test 4*: Computational times for P3/C2 NURBS basis functions vs. $N$.

In conclusion, no strong evidence of a better performance of the POD DL-ROM technique on P1/C0 or P3/C2 basis functions is obtained. The errors obtained in the two cases are indeed very close, as well as the total and testing times; indeed POD DL-ROM provides accurately results and efficient computational times either with P1/C0 or smoother P3/C2 NURBS basis functions.

## 6.6 Test 5: Two-dimensional curved rectangular surface

In this Section we focus again on the solution of the Bidomain equations (1.1) coupled with the R-M ionic model (1.6), by varying the longitudinal conductivity to the fibers direction as in the previous example, in a rectangular surface geometry curved along the $y$-direction (see Figure 6.5). We aim at investigating the effect of the curved geometry, in a benchmark test case, on the performance of the POD DL-ROM technique and the use of pretraining in a geometrical setting. The equations have been discretized in space through P3/C2 NURBS-based IGA and the FOM dimension is equal to $N_h = 325 \times 35 = 11305$. The final time is $T = 150$ ms. The configurations of both the FOM and the POD DL-ROM are equal to the ones provided in the previous example, as well as the parameter under investigation. We set the dimension of the linear trial manifold $\tilde{\mathcal{S}}_h$ to $N = 256$, which results in a projection error $\epsilon_{rel}(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h) = 6 \times 10^{-3}$.

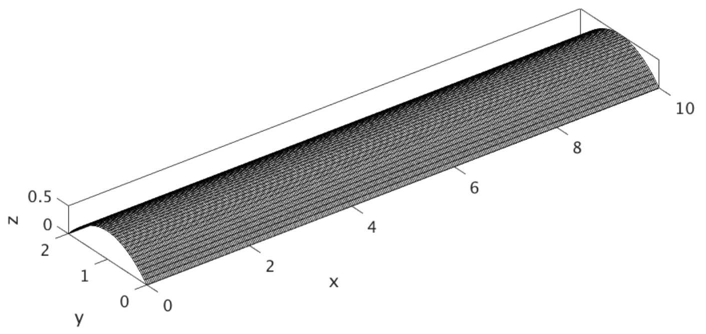


Figure 6.5: *Test 5*: Curved rectangle surface geometry constructed by means of P3/C2 NURBS basis functions.

In Figure 6.6 we report the FOM and the POD DL-ROM solutions, the latter obtained with $n = 2$, together with the relative error (3.14), for the testing-parameter instance $\mu_{test} = 0.199\ \Omega^{-1}\text{cm}^{-1}$ at $t = 62$ ms. The error indicator $\epsilon_{rel}$ is equal to $1.8 \times 10^{-2}$.

Here we investigate the use of pretraining when considering a two-stage procedure relying on different computational geometries. More precisely, we use the optimal weights and biases found in Test 4, that is the rectangle slab test case, where the snapshots are obtained by means of P3/C2 NURBS basis functions over $N_h = 5705$ control points, to initialize the parameters of the POD DL-ROM neural network over the problem detailed in this Section. We highlight that the POD DL-ROM does not depend explicitly on the FOM dimension $N_h$, thus allowing the use of pretraining, but only on the rPOD dimension $N$.

In Table 6.10 we compare the number of parameters, the number of epochs, the total training and validation time and the testing time of the POD DL-ROM neural network trained with and without pretraining, for a prescribed level of accuracy, on a Tesla V100 32GB GPU. The number of parameters and the GPU testing computational times, i.e. the time employed by the POD DL-ROM neural network to compute $N_t = 1500$ time instances for a testing-parameter instance, are the same in the two cases. There we increased the output dimension of the first and second dense layers in the encoder and decoder functions, respectively, with respect to other tests involving similar dimensions $N$ and $n$, in order to achieve a better accuracy. The use of pretraining drastically reduces the number of epochs and, consequently the total time. In particular, relying on pretraining allows a speed-up almost equal to 5 with respect to the time needed to train the neural network from scratch.

The numerical results reported in this Section have shown that employing POD DL-ROM on a more complex geometry, that is the curved rectangle, and using pretraining in a geometrical setting, by providing accurate and efficient solutions, are indeed feasible.

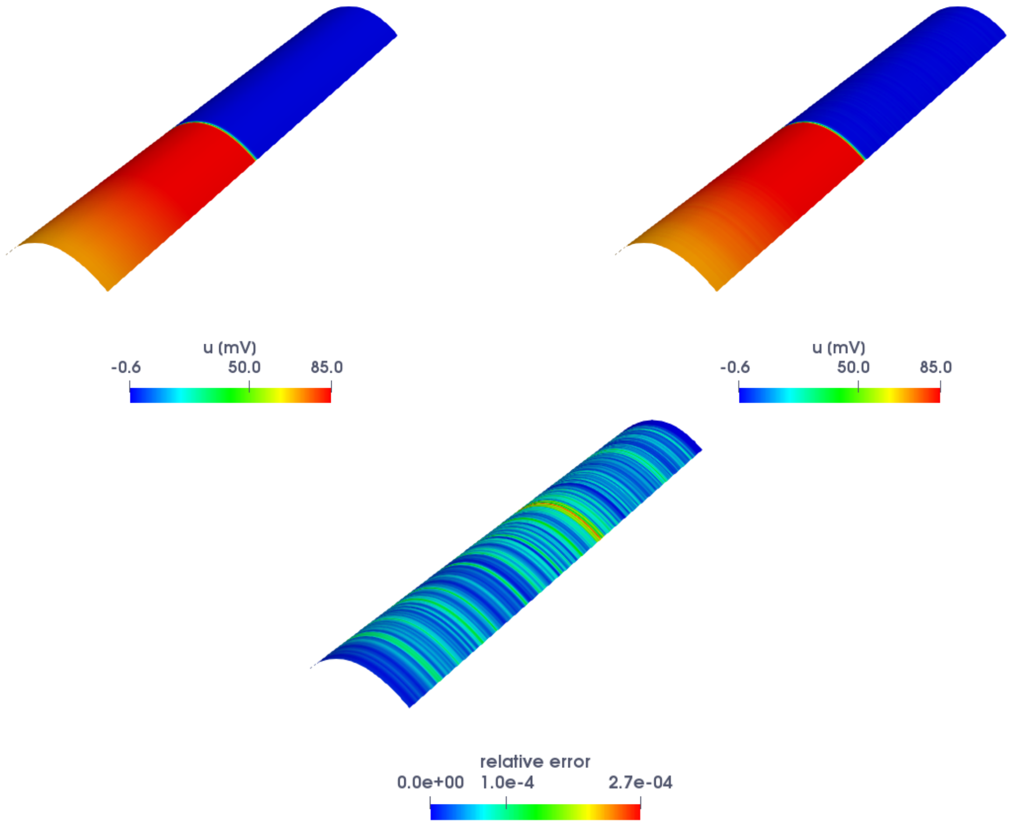


Figure 6.6: *Test 5*: FOM solution (left), POD DL-ROM one (right), with $n = 2$ and $N = 256$, and relative error $\epsilon_k$ (center), for the testing-parameter instance $\mu_{test} = 0.199\ \Omega^{-1}\text{cm}^{-1}$ at $t = 62$ ms.

| | #params | #epochs | total time | test $[s]$ |
|---|---|---|---|---|
| POD DL-ROM | 2382171 | 21159 | 8.2 h | 0.09 |
| POD DL-ROM PRETRAINED | 2382171 | 4419 | 1.7 h | 0.09 |

Table 6.10: *Test 5*: GPU computational times of pretrained and from scratch POD DL-ROM.

## 6.7 Discussion

In this Chapter we assessed the numerical accuracy and efficiency of the POD DL-ROM technique, previously introduced in Chapter 5, on benchmark problems of interest in cardiac EP. The POD DL-ROM enhancements, introduced in order to decrease the training complexity of DL-ROMs, consisting in a first dimensionality reduction, operated by means of rSVD, and the use of a multi-fidelity pretraining, lead, both on FE-method and NURBS-based IGA spatially discretized snapshots, analyzing in the latter case the effects of the basis functions regularity on the performance of the POD DL-ROM, to very efficient training and testing times, thus achieving real-time solutions during the testing phase. In the next Chapter we aim at assessing the performance of the POD DL-ROM on relevant test cases in cardiac EP on realistic geometries, both in physiological and pathological scenarios. These example demonstrate the ability of the POD DL-ROM technique in accurately and efficiently solving remarkably challenging tasks for reduced order modeling due to the steep wavefronts, the complex activation patterns associated to pathological scenarios, the high FOM dimension and the complexity of the geometries.

Chapter 7

# POD DL-ROMs for cardiac electrophysiology (II): realistic geometries and pathological scenarios

In this Chapter, we apply the POD DL-ROM technique, presented in Chapter 5, to relevant problems in cardiac electrophysiology (EP), both in physiological and pathological scenarios, solved on realistic geometries. In particular, we deal with the three-dimensional Zygote template left ventricle (LV) [zyg, 2014] and the idealized surface left atrium (LA) geometries, and high full order model (FOM) dimensions $N_h$, in order to study real scenarios. We remark the great flexibility of POD DL-ROMs in dealing with different spatial discretization, such as the finite element (FE) method [Quarteroni and Valli, 1994] and Isogeometric Analysis (IGA) [Cottrell et al., 2009], as shown in Chapter 6. Moreover, we investigate the construction of a DL-based ROM on the very challenging re-entry break-up problem. Dealing with realistic geometries, large-scale problems, i.e. high FOM dimensions $N_h$, and pathological scenarios, such as re-entries, shows the feasibility of POD DL-ROM to be integrated in the clinical practice in order to compute outputs of interest, e.g. activation maps (ACs), action potential durations (APDs), electrograms (EGMs) and location of rotors' cores.

## 7.1 Test 1: Three-dimensional left ventricle geometry

In this Section, we focus on the application of the POD DL-ROM technique on the three-dimensional Zygote LV geometry test case. We recall that we are interested in the solution of the Monodomain equation (1.3) coupled with the A-P ionic model (1.5) and we consider a single ($n_\mu = 1$) parameter, given by the longitudinal conductivity in the fibers direction, belonging to the parameter space $\mathcal{P} = 12.9 \cdot [0.04, 0.4]$ mm$^2$/ms. The set-up of the FOM (except for the FOM dimension, equal to $N_h = 65503$ in this case), the parameter space, the number of training and testing-parameter instances, the number of time instances and the hyperparameters of the neural network are the same as the ones provided in Test 5 of Section 4.3.

In Figure 7.1 we report the FOM and POD DL-ROM solutions, the latter with $n = 2$ and $N = 256$, and the relative error $\boldsymbol{\epsilon}_k$, at $\tilde{t} = 297.1$ ms, for the testing-parameter instance $\mu_{test} = 12.9 \cdot 0.3243$ mm$^2$/ms. The POD DL-ROM approximation accurately reconstructs the FOM solution, the maximum values of the error $\boldsymbol{\epsilon}_k$ being associated to a very small region in the spatial domain.

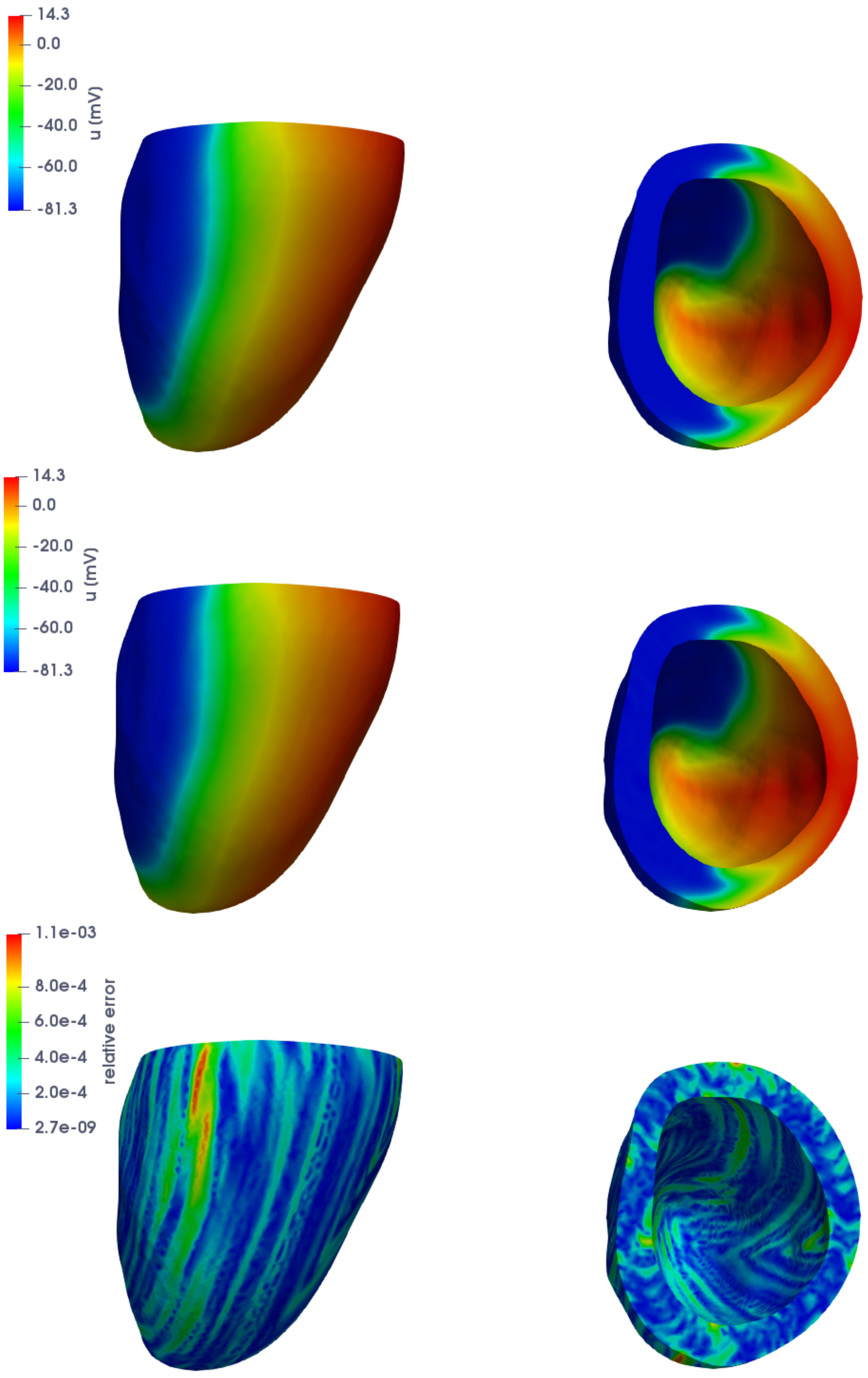


Figure 7.1: *Test 1*: FOM (top), POD DL-ROM (center), with $n = 3$ and $N = 256$, solutions and relative error $\epsilon_k$ (bottom), for the testing-parameter instance $\mu_{test} = 12.9 \cdot 0.3243$ mm$^2$/ms at $\tilde{t} = 297.1$ ms.

In Table 7.1 we report the CPU FOM computational time together with the training (offline), and testing (online) times required by the POD-Galerkin ROM[1] with $N_c = 4$ local bases, and the POD DL-ROM training (total training and validation time) and testing times obtained on a GTX 1070 8GB GPU. The POD DL-ROM allows to achieve a training speed-up equal to 34 and a testing one of $4.8\times10^5$, if compared to the POD-Galerkin ROM training and testing times, respectively. In particular, a POD DL-ROM approximation enables extremely efficient testing computational times, even faster than real-time solutions (here $T = 0.3$ ms).

| FOM | POD-Galerkin ROM: train | POD-Galerkin ROM: test | POD DL-ROM: train | POD DL-ROM: test |
|---|---|---|---|---|
| 3.5 h | 28 h | 120 s | 49 m | 0.25 ms |

Table 7.1: *Test 1*:FOM, POD-Galerkin ROM and POD DL-ROM computational times.

Regarding the FOM dimension, we point out that it was not possible to handle the FOM dimension $N_h = 65503$, by maintaining the batch size $N_b = 40$ and performing the DL-ROM training on the GTX 1070 8GB GPU, due to the high memory consumption necessary to store the model and the data mini-batch on the GPU. By applying instead the POD DL-ROM technique this issue is fixed thanks to the fact that, in this latter case, the neural network only depends on the rPOD dimension of the linear trial manifold, i.e. $N$.

## 7.2 Test 2: Left atrium surface geometry

In this Section, we consider the solution of the Bidomain equations (1.1) coupled with the A-P ionic model (1.5) on an idealized LA surface geometry. The direction of the cardiac fibers is determined by following the same strategy adopted in [Patelli et al., 2017, Rossi et al., 2014], where a vector field directed as the gradient of the solution of a Laplace problem defined on the atrial surface is assigned to the LA. The equations have been discretized in space by means of P2 NURBS basis functions, the majority with a global C1 continuity, with $N_h = 61732$. Time integration is performed over the interval $(0, T)$ with $T = 200$ and a time-step $\Delta t = 0.2$. Provided the position of the Bachmann bundle $\bar{\mathbf{x}} = (\bar{x}, \bar{y}, \bar{z})^T = (-1.51, 0.1, -1.71)^T$ cm, one of the four points at which the interatrial conduction shall occur, the intracellular applied stimulus is given by

$$I^i_{app}(\mathbf{x}, t) = C\mathbf{1}_{\Omega_{app}}(\mathbf{x})\mathbf{1}_{[t^i,t^f]}(t),$$

with $C = 1$ mA, $\Omega_{app} = \{\mathbf{x} \in \Omega : (x - \bar{x})^2 + (y - \bar{y})^2 + (z - \bar{z})^2 \leq (0.5)^2\}$, $t^i = 0$ and $t^f = 5$.

The parameter ($n_\mu = 1$) consists in the electric intracellular conductivity in the longitudinal direction to the fibers, i.e., the conductivity tensor $\mathbf{D}_i(\mathbf{x}; \boldsymbol{\mu})$ takes the form

$$\mathbf{D}_i(\mathbf{x}; \mu) = \sigma^i_t I + (\mu - \sigma^i_t)\mathbf{f}_0 \otimes \mathbf{f}_0,$$

where the parameter space is $\mathcal{P} = 3.1 \cdot [10^{-4}, 10^{-2}]$ $\Omega^{-1}\text{cm}^{-1}$. The remaining intra- and extracellular conductivities are set equal to $\sigma^i_t = 2\times10^{-2}$ $\Omega^{-1}\text{cm}^{-1}$, $\sigma^e_l = 1.3\times10^{-4}$ $\Omega^{-1}\text{cm}^{-1}$ and $\sigma^e_t = 2\times10^{-3}$ $\Omega^{-1}\text{cm}^{-1}$. The parameters of the A-P ionic model (1.5) are given by $K = 8$, $a = 0.1$, $\epsilon_0 = 0.01$, $b = 0.1$, $c_2 = 0.3$ and $c_1 = 0.05$ [ten Tusscher, 2004].

For the training phase, we uniformly sample $N_t = 500$ time instances in the interval $(0, T)$ and consider $N_{train} = 21$ training-parameter instances uniformly distributed over $\mathcal{P}$. For the testing phase, $N_{test} = 20$ testing-parameter instances have been considered, each of them corresponding to the midpoint of two consecutive training-parameter instances. The maximum number of epochs is set to $N_{epochs} = 20000$, the batch size is $N_b = 40$ and, regarding

[1]The FOM and the POD-Galerkin ROM simulations are carried out on a full 64 GB node (20 Intel® Xeon® E5-2640 v4 2.4GHz cores) of a HPC cluster.

the early-stopping criterion, we stop the training if the loss function does not decrease along 1000 epochs.

In Figure 7.2 we show the FOM solution and the POD DL-ROM approximation, with $n = 2$ and $N = 256$, for the testing-parameter instance $\mu_{test} = 0.001055\ \Omega^{-1}\text{cm}^{-1}$ at $t = 67.4$ and $t = 116.6$. In Figure 7.3 we report instead the error $\boldsymbol{\epsilon}_k$ associated to the time instances considered. The error indicator $\epsilon_{rel}$ is equal to $2.1 \times 10^{-2}$.

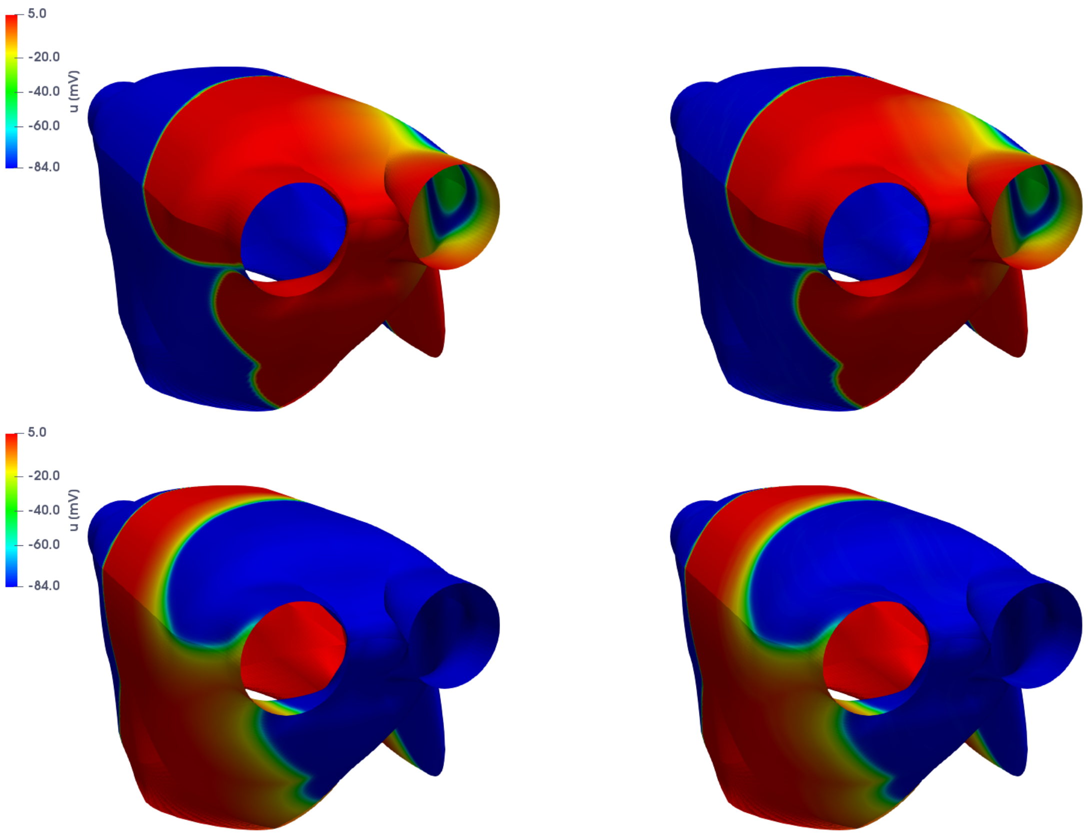


Figure 7.2: *Test 2*: FOM solution (left) and POD DL-ROM one (right), with $n = 2$ and $N = 256$, for the testing-parameter instance $\mu_{test} = 0.001055\ \Omega^{-1}\text{cm}^{-1}$ at $t = 67.4$ (top) and $t = 116.6$ (bottom).

In Table 7.2 we show the CPU computational time needed to solve the FOM by means of NURBS-based IGA on a 6-core platform[2] and the GPU POD DL-ROM total training and validation time, together with the testing time. The time needed to train the POD DL-ROM in this case is higher than the one of the previous example, even if the two problems share the same dimension $N$ and almost the same dimensions $N_h$ and $N_{train}$. This difference is then related to the complexity of the LA geometry; indeed we recall that the input of the POD DL-ROM, i.e. the FOM solution, is reshaped into a 2D matrix before entering the first layer of the neural network. In the case of LA, being the geometry highly complex, when the FOM solution is reshaped, the spatial correlation is completely lost; on the other hand, being the geometry of the LV, introduced in Section 7.1, more regular, the spatial distribution is partially preserved. However, we are able to achieve again the chance of solving the problem in several different scenarios, during the testing stage, in real-time. Indeed, here $T = 0.2$, which coincides with the computational time entailed by the POD DL-ROM.

[2]Numerical tests have been performed on a MacBook Pro Intel Core i7 6-core with 16 GB RAM.

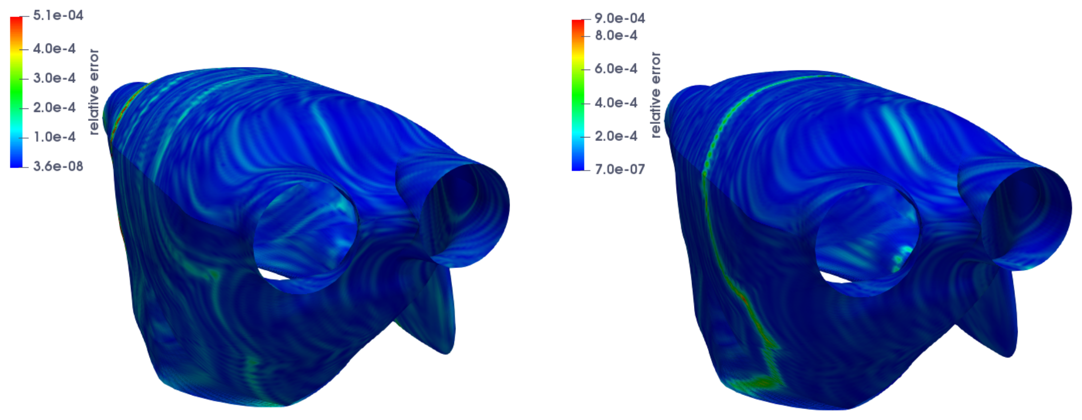


Figure 7.3: *Test 2*: Relative error $\boldsymbol{\epsilon}_k$ for the testing-parameter instance $\mu_{test} = 0.001055\ \Omega^{-1}\text{cm}^{-1}$ at $t = 67.4$ (left) and $t = 116.6$ (right).

| FOM | POD DL-ROM: train | POD DL-ROM: test |
|---|---|---|
| 2 h | 3.5 h | 0.2 s |

Table 7.2: *Test 2*: FOM and POD DL-ROM computational times.

## 7.3 Test 3: Left atrium surface geometry varying stimulation site

Here we focus on the computation of the solution of the Bidomain equations (1.1) coupled with the R-M model (1.6), thus considering a different ionic model with respect to the one taken into account in the previous Section, on an idealized LA surface geometry. The direction of the cardiac fibers is determined as in the previous example. The equations have been discretized in space by means of P2 NURBS basis functions, the majority with a global C1 continuity, with $N_h = 154036$. Time integration is performed over the interval $(0, T)$, with $T = 200$ ms and a time-step $\Delta t = 0.1$ ms.

The parameters ($n_\mu = 3$) consist in the coordinates of the center of the intracellular applied current, and belong to the two-dimensional subdomain, the fuchsia region shown in Figure 7.4, together with the portion of the domain affected by the stimulus, the pink region. The intracellular applied current is defined as

$$I^i_{app}(\mathbf{x}, t) = C\mathbf{1}_{\Omega_{app}(\boldsymbol{\mu})}(\mathbf{x})\mathbf{1}_{[t^i,t^f]}(t),$$

with $C = 100$ mA, $\Omega_{app}(\boldsymbol{\mu}) = \{\mathbf{x} \in \Omega : (x - \mu_1)^2 + (y - \mu_2)^2 + (z - \mu_3)^2 \leq (0.5)^2\}$, $t^i = 0$ ms and $t^f = 5$ ms.

For the training phase, we uniformly sample $N_t = 200$ time instances in the interval $(0, T)$ and consider $N_{train} = 18$ training-parameter instances randomly sampled from the parameter space. For the testing phase, $N_{test} = 14$ randomly sampled testing-parameter instances have been considered. The maximum number of epochs is $N_{epochs} = 40000$, the batch size is $N_b = 40$, the starting learning rate is $\eta = 2 \cdot 10^{-4}$ and, regarding the early-stopping criterion, we stop the training if the loss function does not decrease along 2000 epochs.

In order to point out the ability of the POD DL-ROM approximation to approach the FOM solution when computing outputs of interest we show, in Figure 7.5, the FOM and POD DL-ROM activation maps (ACs), together with the associated $\boldsymbol{\epsilon}_k$, for the testing-parameter instances $\boldsymbol{\mu}_{test} = (1.7168, -0.353198, -1.70097)$ cm and $\boldsymbol{\mu}_{test} = (1.43862, -0.803806, -1.43678)$ cm. We highlight the strong variability of the solution over the parameter space, shown by the different shape of the contour lines in Figure 7.5 (top) and (bottom), meaning that the

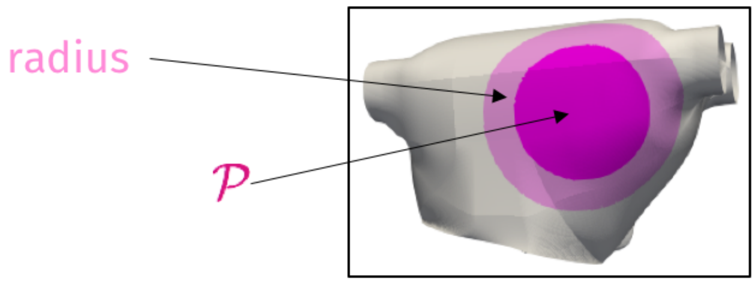


Figure 7.4: *Test 3*: Parameter space (fuchsia region) and portion of domain affected by the stimulus (pink region).

propagation direction of the front remarkably varies over $\mathcal{P}$, and the ability of the POD DL-ROM solution to capture it accurately.

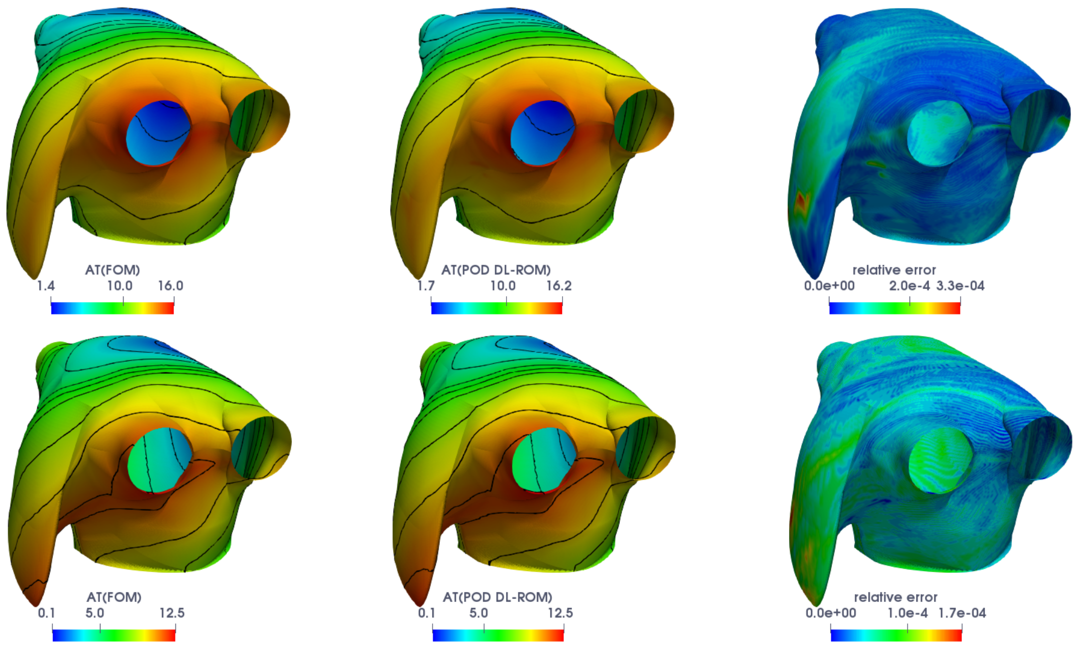


Figure 7.5: *Test 3*: FOM (left) and POD DL-ROM (center), with $n = 4$ and $N = 256$, ACs and relative error $\epsilon_k$ (right), for the testing-parameter instances $\boldsymbol{\mu}_{test} = (1.7168, -0.353198, -1.70097)$ cm (top) and $\boldsymbol{\mu}_{test} = (1.43862, -0.803806, -1.43678)$ cm (bottom).

Finally, in Table 7.3 we show the FOM CPU computational time[3] and the POD DL-ROM GPU training and testing times. We highlight that solving the FOM, for a single testing-parameter instance, requires 10 h, with respect to the POD DL-ROM total training and validation time which is equal to 5 h. Moreover, POD DL-ROM shows to be extremely efficient at testing time, by being able to provide results, once again, in real-time.

[3]Numerical tests have been carried out on a MacBook Pro Intel Core i7 6-core with 16 GB RAM.

| FOM | POD DL-ROM: train | POD DL-ROM: test |
|---|---|---|
| 10 h | 5 h | 0.2 s |

Table 7.3: *Test 3*: FOM and POD DL-ROM computational times.

## 7.4 Test 4: Figure of eight re-entry on left atrium surface

We now investigate the computation of the figure of eight re-entry on the idealized LA surface geometry in order to highlight the ability of the POD DL-ROM technique in tackling pathological cardiac EP problems. The set-up of the FOM is the one provided in Subsection 1.4.4, except for the final time equal to $T = 500$ ms.

The parameters ($n_\mu = 3$) consist in the coordinates of the center of the S2 intracellular applied currents and belong to the two-dimensional parameter space $\mathcal{P}$ highlighted in Figure 7.6. The choice of the parameter space is motivated by the fact that ectopic complexes usually arise in correspondence of pulmonary veins (PVs). We first apply a physiological stimulus in correspondence of the posterior septum and then the S2 stimulus, which takes the form

$$I_{app}^{i,2}(\mathbf{x},t) = C\mathbf{1}_{\Omega_2(\boldsymbol{\mu})}(\mathbf{x})\mathbf{1}_{[t_2^i,t_2^f]}(t),$$

with $C = 100$ mA, $\Omega_2(\boldsymbol{\mu}) = \{\mathbf{x} \in \Omega : (x-\mu_1)^2 + (y-\mu_2)^2 + (z-\mu_3)^2 \leq (0.5)^2\}$, $t_2^i = 210$ ms and $t_2^f = 215$ ms, is applied.

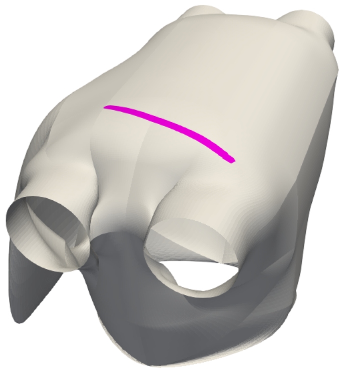


Figure 7.6: *Test 4*: Parameter space (fuchsia region).

This test case is a proof-of-concept of the strategy used in the clinical practice in order to identify possible re-entrant circuits, part of which may be latent, by conducting a virtual multi-site delivery of electrical stimuli from a number of atrial locations [Boyle et al., 2018, Arevalo et al., 2016, Prakosa et al., 2018].

We consider $N_t = 1000$ time instances in the interval $(300, 500)$ ms and randomly sample $N_{train} = 15$ training-parameter and $N_{test} = 5$ testing-parameter instances from the parameter space. The maximum number of epochs is $N_{epochs} = 30000$, the batch size is $N_b = 40$ and, regarding the early-stopping criterion, we stop the training if the loss function does not decrease along 2000 epochs.

We firstly set the rPOD dimension equal to $N = 256$ which results, over the testing set, in the projection error indicator $\epsilon_{rel}(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h) = 6.8 \times 10^{-2}$ and in the projection relative error $\boldsymbol{\epsilon}_k(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h)$ shown in Figure 7.7. Then, we do not expect the reconstruction error indicators being smaller than the previous values over the testing set.

In Figure 7.8 we compare the FOM and POD DL-ROM solutions, the latter with $n = 4$ and $N = 256$, together with $\boldsymbol{\epsilon}_k$, for the testing-parameter instance $\boldsymbol{\mu}_{test} = (0.2508, 0.7932, 1.66)$ cm at $t = 316.4$ ms. The error indicator $\epsilon_{rel}(\mathbf{u}_h, \tilde{\mathbf{u}}_h)$ is equal to $7.06 \times 10^{-2}$, meaning that

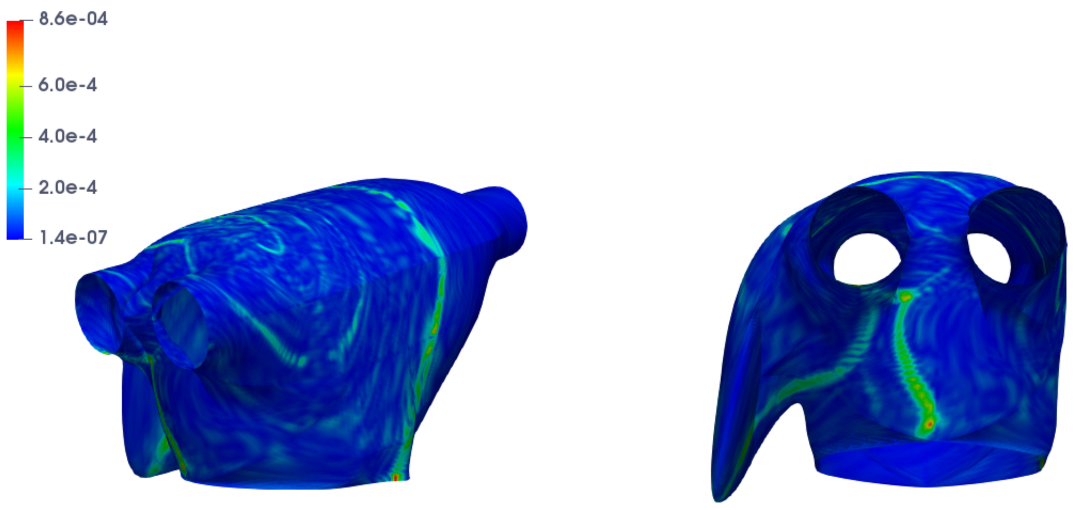


Figure 7.7: *Test 4*: Projection relative error $\boldsymbol{\epsilon}_k(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h)$ with $N = 256$.

the projection error limits $\epsilon_{rel}(\mathbf{u}_h, \tilde{\mathbf{u}}_h)$ over the testing set. However, the POD DL-ROM is able to completely capture the location and the shape of the re-entry, and the moving front; the error is related to the reconstruction of the steep fronts. Hence, we consider the results obtained completely satisfactory, given the extreme complexity of the problem at hand.

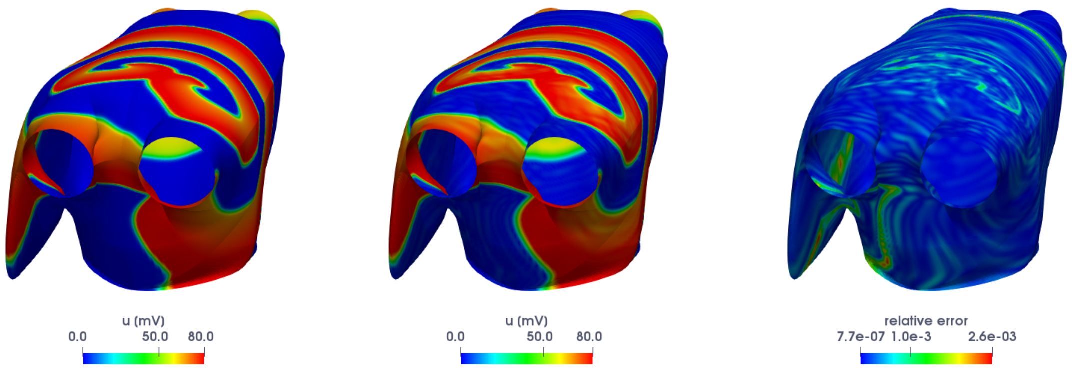


Figure 7.8: *Test 4*: FOM (left) and POD DL-ROM (center) solutions, the latter obtained with $n = 4$ and $N = 256$, together with $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instance $\boldsymbol{\mu}_{test} = (0.2508, 0.7932, 1.66)$ cm at $t = 316.4$ ms.

Then we investigate the impact of a higher value for the rPOD dimension; indeed, we set it equal to $N = 1024$. In this case, the projection error indicator $\epsilon_{rel}(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h)$ is equal to $2.84 \times 10^{-2}$ and the error indicator (3.13) becomes $\epsilon_{rel} = 5.4 \times 10^{-2}$. In Figure 7.9 we report the FOM solution and the POD DL-ROM approximation, with $n = 4$ and $N = 1024$, together with the relative error defined in (3.14), for the testing-parameter instance $\boldsymbol{\mu}_{test} = (0.2508, 0.7932, 1.66)$ cm at $t = 316.4$ ms. By comparing Figures 7.8 (right) and 7.9 (right), we can note how the use of a larger $N$ leads to slightly more accurate results.

In Table 7.4 we report the FOM CPU computational time on a 6-core platform[4] and the POD DL-ROM GPU total, i.e. training and validation time, and testing times, and the total number of epochs $n_e$, obtained on a Tesla V100 32GB GPU, by varying $N$. As expected, both the training and the testing times are larger for $N = 1024$ than $N = 256$, being the

[4]Numerical tests have been performed on a MacBook Pro Intel Core i7 6-core with 16 GB RAM.

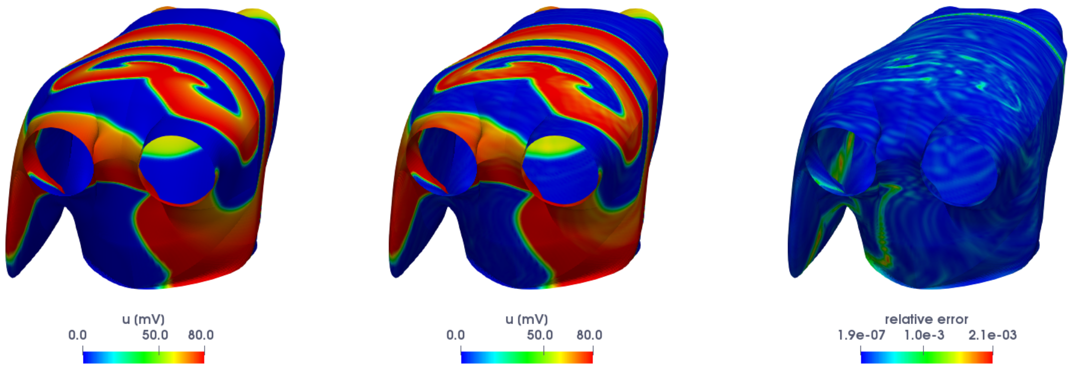


Figure 7.9: *Test 4*: FOM (left) and POD DL-ROM (center) solutions, the latter obtained with $n = 4$ and $N = 1024$, together with $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instance $\boldsymbol{\mu}_{test} = (0.2508, 0.7932, 1.66)$ cm at $t = 316.4$ ms.

number of parameters of the network higher in the former case. We highlight that, if we do not consider the time needed to assemble the snapshot matrix, the time required to train the POD DL-ROM over the parameter space, for $N = 256$, is smaller than performing a FOM simulation for a single parameter instance. We remark that we started from a learning rate equal to $\eta = 2 \cdot 10^{-4}$ for $N = 256$ and $\eta = 10^{-4}$ for $N = 1024$, the latter resulting in a longer total training and validation time; indeed, in this case training stops because of the maximum number of epochs achieved, however yielding a higher accuracy. At testing time, both the networks show to be extremely efficient.

| | FOM | POD DL-ROM: train | POD DL-ROM: test | $n_e$ |
|---|---|---|---|---|
| $N = 256$ | 7.2 h | 4.9 h | 0.32 s | 8849 |
| $N = 1024$ | 7.2 h | 20 h | 0.77 s | 30000 |

Table 7.4: *Test 4*: FOM and POD DL-ROM computational times.

As done in Test 4 of Section 4.3, we increase the complexity of the problem by enlarging the dimension of the parameter space, thus considering both re-entry and non re-entry dynamics. We randomly sample $N_{train} = 20 + 20 = 40$ training-parameter and $N_{test} = 10 + 10 = 20$ testing-parameter instances from the parameter space.

We set the rPOD dimension equal to $N = 1024$. In this case, the projection error indicator value is $\epsilon_{rel}(\mathbf{u}_h, \mathbf{V}\mathbf{V}^T\mathbf{u}_h) = 4.34\times10^{-2}$ and the reconstruction one is $\epsilon_{rel}(\mathbf{u}_h, \tilde{\mathbf{u}}_h) = 7.7\times10^{-2}$. We set the maximum number of epochs $N_{epochs}$ to 30000, by increasing this value it is possible to achieve a reconstruction error equal to the projection one. The parameter space is the one shown in Figure 7.10.

In Figure 7.11 we report the FOM and POD DL-ROM solutions, with $n = 4$ and $N = 1024$, along with $\boldsymbol{\epsilon}_k$, for the testing-parameter instances $\boldsymbol{\mu}_{test} = (0.3162, 0.8638, 0.6864)$ cm and $\boldsymbol{\mu}_{test} = (0.2508, 0.7932, 0.8895)$ cm at $t = 300.8$ ms. The POD DL-ROM thus shows to be able to reproduce the main features of the dynamics of the solution, and the error is mainly associated to the truncated POD modes.

## 7.5 Test 5: Re-entry break-up

Few attempts have been made in order to solve, by means of DL algorithms, problems featuring a chaotic and disorganized solution. In [Raissi, 2018] the Kuramoto-Sivashinsky equation, in a chaotic regime, is solved by means of physics-informed neural networks (PINNs), but

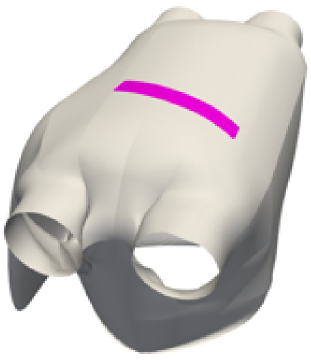


Figure 7.10: *Test 4*: Parameter space (fuchsia region).

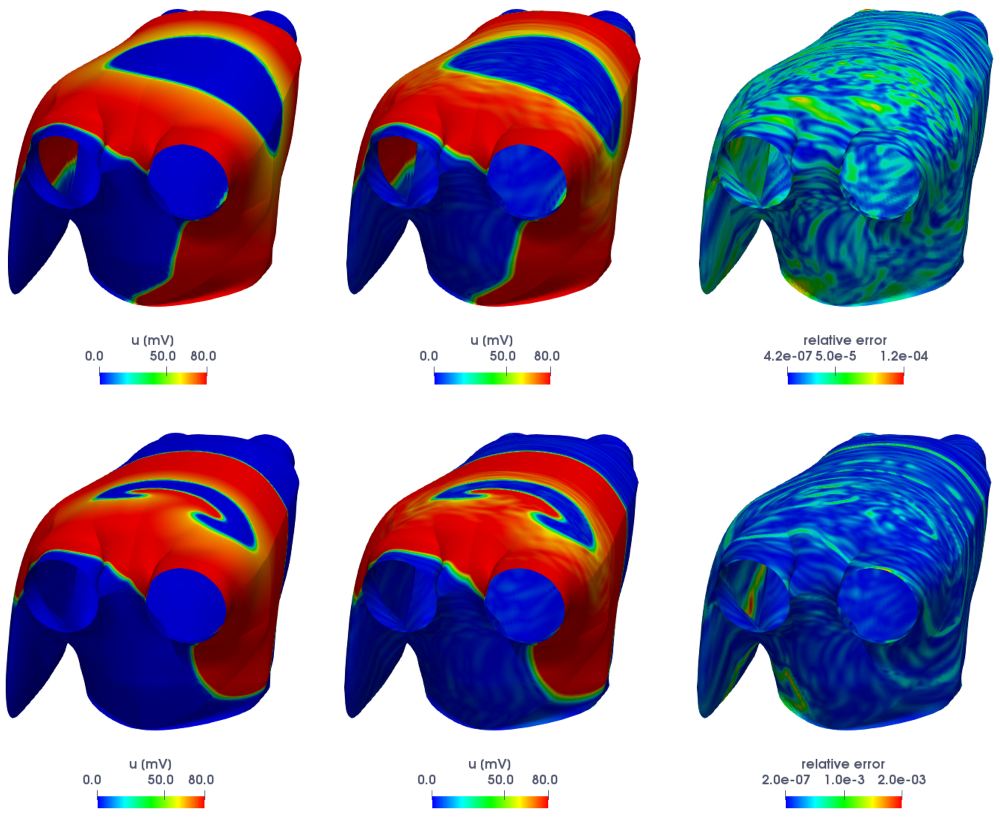


Figure 7.11: *Test 4*: FOM (left) and POD DL-ROM (center) solutions, the latter obtained with $n = 4$ and $N = 1024$, together with $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instances $\boldsymbol{\mu}_{test} = (0.3162, 0.8638, 0.6864)$ cm (top) and $\boldsymbol{\mu}_{test} = (0.2508, 0.7932, 0.8895)$ cm (bottom) at $t = 300.8$ ms.

the algorithm leads to not completely satisfactory results. In [Pathak et al., 2018a, Pathak et al., 2018b] a hybrid forecasting scheme based on reservoir computing in conjunction with knowledge-based models are successfully applied to prototype spatiotemporal chaotic systems. In [Yeo, 2017] a deep neural network for a model-free prediction of a chaotic dynamical system from noisy observations is presented. In that case, the proposed DL model aims to predict the conditional probability distribution of the state variable. Here we want to apply the techniques presented in this Thesis to the chaotic solution of the re-entry break-up problem.

In particular, we focus on the solution of the Monodomain equation (1.3) coupled with the A-P ionic model (1.5) over the domain $\Omega = (0, 4 \text{ cm})^2$ discretized by means of linear finite elements with $N_h = 128 \times 128 = 16384$ grid points. Time integration is performed over the interval $(0, T)$, with $T = 900$ ms and a time-step $\Delta t = 0.2/12.9$, by means of a one-step, semi-implicit, first order scheme. The fibers are parallel to the $x$-axis and the conductivities in the longitudinal and transversal directions to the fibers are $\sigma_l = 2 \times 10^{-3}$ cm$^2$/ms and $\sigma_t = 3.1 \times 10^{-4}$ cm$^2$/ms, respectively. We set the parameters of the A-P model equal to $K = 8$, $a = 0.1$, $\epsilon_0 = 0.01$, $b = 0.1$, $c_2 = 0.3$ and $c_1 = 0.05$ [ten Tusscher, 2004] and apply the cross-field stimulation protocol to generate the re-entry break-up.

The parameter (here $n_\mu = 1$) consists of the $x$-coordinate of the location of the S2 stimulus, which takes the form

$$I_{app}^{i,2}(\mathbf{x}, t) = C\mathbf{1}_{\Omega_2(\mu)}(\mathbf{x})\mathbf{1}_{[t_2^i, t_2^f]}(\tilde{t}),$$

where $C = 1$ mA, $\Omega_2(\mu) = \{\mathbf{x} \in \Omega : x \leq \mu\}$, $t_2^i = 125$ ms, $t_2^f = 130$ ms and $\mu \in \mathcal{P} = [0.5, 2.25]$ cm.

In order to investigate the feasibility of applying the POD DL-ROM technique on this challenging problem, we analyze the decay of the eigenvalues of the training snapshot matrix $\mathbf{S}^{train}$. In particular, we uniformly sample $N_{train} = 8, 15, 25, 35$ training-parameter instances from $\mathcal{P}$ and we compute the respective snapshot matrix. In Figure 7.12 we show the decay of the eigenvalues for $\epsilon_{POD} = 10^{-3}$, which results in a dimension of the linear trial subspace equal to $N = 2247, 3804, 5632, 6678$. The dimension $N$ remarkably increases with respect to $N_{train}$, meaning that it is not possible to reduce the problem over the parameter space by means of a linear ROM; indeed, a huge number of modes would be required to get an accurate approximation of the dynamics. Motivated by this fact, we decided to apply the DL-ROM technique to the problem under investigation.

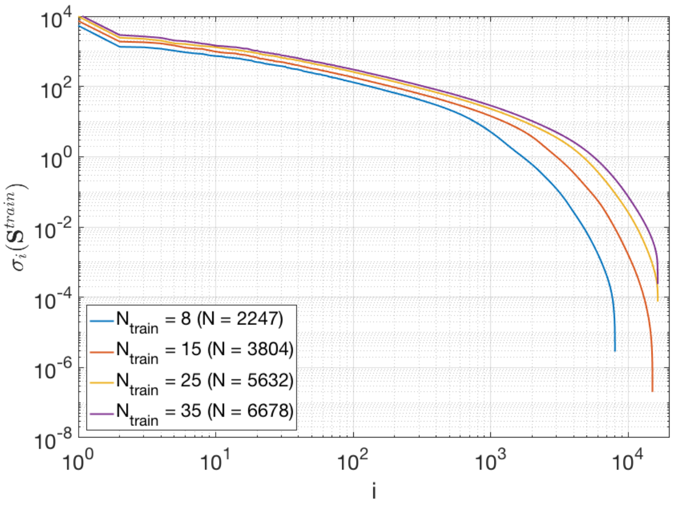


Figure 7.12: *Test 5*: Decay of $\sigma_i$ for different $N_{train}$.

We consider $N_t = 1000$ time instances uniformly distributed over the interval $(600, 900)$ ms, $N_{train} = 26$ training- and $N_{test} = 21$ testing-parameter instances, randomly sampled from the parameter space. The maximum number of epochs is set equal to $N_{epochs} = 30000$ and the batch size is $N_b = 40$. Regarding the early-stopping criterion, we stop the training if the loss does not decrease in 3000 epochs.

In Figure 7.13 we show the FOM and DL-ROM solutions, the latter with $n = 20$, along with the relative error $\boldsymbol{\epsilon}_k$, for the testing-parameter instances $\mu_{test} = 0.535$ cm and $\mu_{test} =$

0.7125 cm at $\tilde{t} = 688$ ms. In particular, $\mu_{test} = 0.535$ cm consists in the midpoint among two training-parameter instances whereas $\mu_{test} = 0.7125$ cm is a testing-parameter instance close to the training-parameter instance $\mu_{train} = 0.7$ cm; indeed in the last case the relative error $\boldsymbol{\epsilon}_k$ is smaller. We point out the impressive variability shown by the solution for different parameters values, at each single time. Even if the steep fronts of the FOM solution are not sharply reconstructed, the DL-ROM solution is able to capture all the multiple re-entries and the main dynamics, and to provide useful information like the location of rotors' cores.

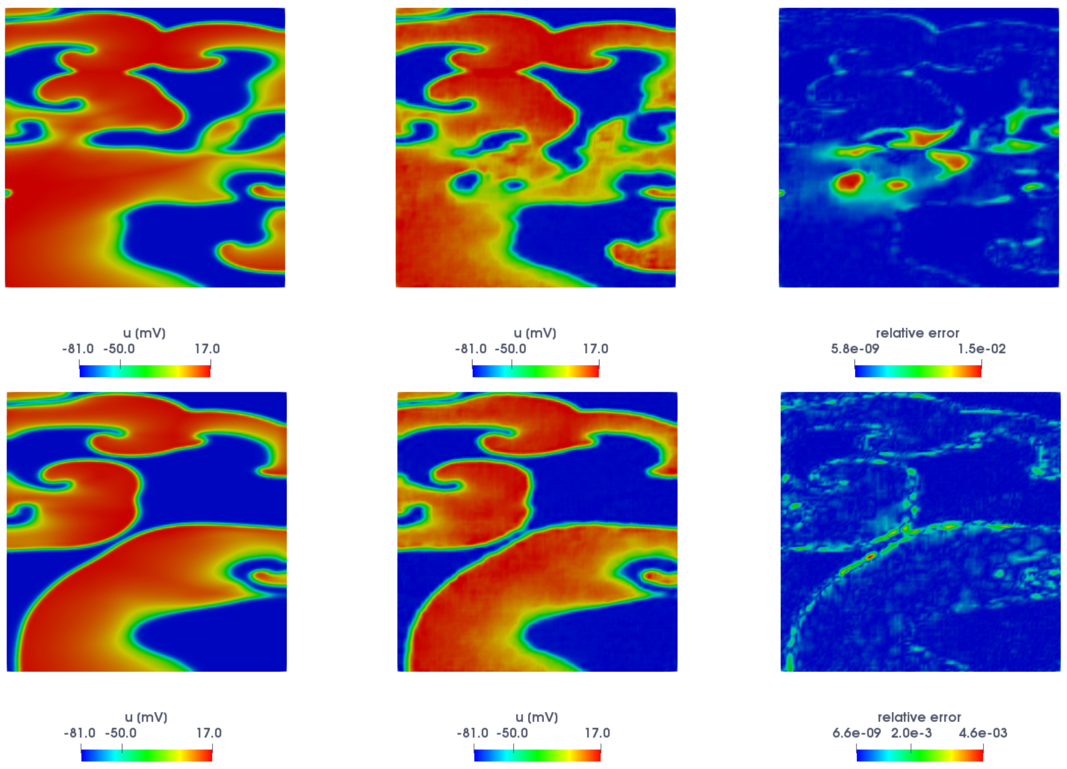


Figure 7.13: *Test 5*: FOM (left) and DL-ROM (right), with $n = 10$, solutions, together with $\boldsymbol{\epsilon}_k$ (right), for the testing-parameter instances $\mu_{test} = 0.535$ cm (top) and $\mu_{test} = 0.7125$ cm (bottom) at $\tilde{t} = 688$ ms.

In the previous results we have considered a quite large parameter space, considered the strong variability of the solution of the re-entry break-up problem over $\mathcal{P}$, equal to almost half edge of the domain. We expect that, by reducing the dimension of the parameter space, a higher accuracy could be achieved.

Finally, the DL-ROM testing computational time, on a GTX 1070 8 GB GPU, is 0.35 s, thus generating a speed-up, with respect to the FOM CPU computational time[5], equal to $1.48 \times 10^3$.

At the best of our knowledge, this is the first trie in which chaotic regimes featuring so many scales and depending on physical parameters are accurately reproduced by reduced order modeling techniques.

[5]Numerical tests have been performed on a MacBook Pro Intel Core i7 6-core with 16 GB RAM.

## 7.6 Discussion

In this Chapter, the performance of the (POD) DL-ROM technique has been verified on relevant test cases in cardiac EP on realistic geometries, both in physiological and pathological conditions, and by considering both FE method and NURBS-based IGA spatially discretized snapshots. The proposed POD DL-ROM framework can efficiently perform the training phase, thus drastically decreasing the computational complexity of DL-ROMs at training time, and provide real-time solutions to parametrized EP problems at testing time. The accuracy and the efficiency obtained by the (POD) DL-ROM approximations make them amenable, in the clinical setting, to replace the FOM solutions when computing quantities of interest, such as ACs and APDs.

# Conclusions

In this Thesis we have introduced a new generation of non-intrusive, nonlinear reduced order models, based on deep learning algorithms, to efficiently and accurately deal with nonlinear time-dependent parametrized PDEs featuring coherent structures which propagate over time. We considered transport, wave, or convection-dominated phenomena, on which conventional linear reduced order models reveal inefficient, with a focus on the solution of parametrized problems arising in cardiac electrophysiology in pathological scenarios, such as atrial tachycardia and atrial fibrillation.

Cardiac electrophysiology systems fit into both *(i)* a multi-query context, since repetitive evaluations of the input-output map are required in order to perform multi-scenario analysis, to deal with inter- and intra-subject variability and to consider specific pathological scenarios, and into *(ii)* a real-time context, due to the need, in a clinical setting, to compute outputs of interest in a very limited amount of time. Performing the numerical approximation of cardiac electrophysiology problems in these contexts, by means of traditional full order models, such as the finite element method or NURBS-based Isogeometric Analysis, is prohibitive because of the huge computational costs associated to the solution of the equations. Indeed, small time-step sizes must be selected to ensure stability; small mesh sizes are required in order to capture the steep fronts and preserve accuracy.

We proposed a novel technique to build low-dimensional reduced order models by exploiting deep learning algorithms to overcome typical computational bottlenecks shown by classical, linear projection-based reduced order models techniques (such as the reduced basis method relying on proper orthogonal decomposition) when dealing with problems featuring coherent structures propagating over time. The DL-ROM technique allows to approximate the solution manifold of a given parametrized nonlinear, time-dependent PDE by means of a low-dimensional, nonlinear trial manifold, and the nonlinear dynamics of the generalized coordinates on such reduced manifold, as a function of the time coordinate and the parameters. Both the nonlinear trial manifold and the reduced dynamics are learnt in a non-intrusive way, thus avoiding to query the arrays related to the full order model. The former is learnt by means of the decoder function of a convolutional autoencoder neural network, the latter through a deep feedforward neural network, and the encoder function of the convolutional autoencoder. The numerical results reported in this Thesis show that the proposed DL-ROM technique provides accurate solutions to parametrized PDEs, involving a low-dimensional reduced manifold whose dimension is equal to (or slightly larger than) the dimension of the solution manifold. Our numerical tests have shown that employing deep learning techniques to build reduced order models for parametrized nonlinear PDEs is indeed a feasible way, both in terms of numerical accuracy and computational efficiency.

Through the use of the DL-ROM, it is possible to boost the solution of parametrized problems in cardiac electrophysiology, both in physiological and pathological conditions, thus overcoming the main computational bottlenecks shown by POD-Galerkin ROMs in this context, such as *(i)* the linear superimposition of modes which linear reduced order models are based on, *(ii)* the need to account for the gating variables when solving the reduced dynamics, even if not required, *(iii)* the necessity to use hyper-reduction techniques to deal with terms that depend nonlinearly on either the transmembrane potential or the input parameters, and *(iv)* the need to solve a dynamical system until the desired time (the DL-ROM approximation can be queried at any time instance). Moreover, outputs of clinical interest, such as activation maps and action potentials, can be more efficiently evaluated by the DL-ROM technique than by a full order model while maintaining a high level of accuracy.

A key aspect in the setting of DL-ROMs concerns computational efficiency during the offline (or training) stage, which is also related with the curse of dimensionality. In this respect, we developed a strategy to enhance DL-ROMs in order to make the offline training stage dramatically faster. This strategy, which we refer to as POD DL-ROM, exploits *(i)* dimensionality reduction of full order model snapshots through randomized proper orthogonal decomposition and *(ii)* a suitable multi-fidelity pretraining stage exploiting snapshots computed through lower-fidelity models to initialize the parameters of neural networks in a sequential procedure. The performance of the POD DL-ROM technique has been verified on benchmark problems and challenging test cases on realistic geometries, both in physiological and pathological cardiac electrophysiology. The proposed POD DL-ROM framework can efficiently perform the training phase and provide real-time solutions to parametrized cardiac electrophysiology problems at testing time. Not only, we assessed computational performance, numerical accuracy and robustness of the POD DL-ROM technique on different problems, including advection-diffusion-reaction equations, nonlinear structural mechanics, and fluid dynamics. In all these cases, POD DL-ROMs are able to match the intrinsic dimension of the problems investigated, to overcome the main computational bottlenecks shown by conventional projection-based methods and to make the training phase extremely fast.

Due to their data-driven and non-intrusive nature, DL-ROMs and POD DL-ROMs might be considered as turn-key reduced order modeling techniques to handle applications of interest; depending on the complexity of the problem at hand it is also possible, in principle, to switch between these two strategies. In particular, the proposed reduced order modeling framework has been successfully applied to parametrized physiological and pathological problems in cardiac electrophysiology, considering both finite element and Isogeometric Analysis. At the best of our knowledge, the results reported in the last Chapter of this Thesis represent the first attempt at reducing the computational complexity associated to the re-entry and the re-entry break-up problems, this opening, virtually, a new path towards the model personalization in real-time, even when dealing with extremely challenging, and computationally involved, scenarios. The possibility to perform real-time numerical simulations, in cardiac electrophysiology, can be seen as the first step towards the translation of computational methods in the clinical practice, aiming at complementing clinical data, in view of supporting decisions, quantifying risks related to cardiac pathologies, planning therapies and interventions.

We envision several future extensions of the work presented in this Thesis, such as, for instance:

- enhancing the description of spatial heterogeneity by providing clinical data under the form of medical images as input to the (POD) DL-ROM framework, thus replacing the deep feedforward neural network, which now describes the reduced dynamics onto the non linear trial manifold, with a convolutional neural network;
- enforcing physics-based laws in the (POD) DL-ROM framework, for instance by relying on physics-informed neural networks;

- replacing intrusive and expensive hyper-reduction techniques, such as the discrete empirical interpolation method, with the POD DL-ROM technique;
- extending the POD DL-ROM framework to multi-physics problems (such as, e.g. electromechanics);
- exploiting the POD DL-ROM technique to enhance the evaluation of quantities of interest, to address sensitivity analysis and uncertainty quantification tasks efficiently;
- embedding POD DL-ROM within the clinical practice, to compute real-time outputs accounting for inter- and intra-subject variability.

# Bibliography

[zyg, 2014] (2014). Zygote solid 3D heart generation II developement report. Technical report, Zygote Media Group Inc.

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). *TensorFlow: A system for large-scale machine learning.*

[Abgrall et al., 2016] Abgrall, R., Amsallem, D., and Crisovan, R. (2016). Robust model reduction by $L^1$-norm minimization and approximation via dictionaries: application to nonlinear hyperbolic problems. *Advanced Modeling and Simulation in Engineering Sciences*, 3(1).

[Agarap, 2018] Agarap, A. F. (2018). Deep learning using rectified linear units (ReLU). *arXiv preprint arXiv:1803.08375.*

[Aliev and Panfilov, 1996] Aliev, R. R. and Panfilov, A. V. (1996). A simple two-variable model of cardiac excitation. *Chaos Solitons Fractals*, 7(3):293–301.

[Allessie and de Groot, 2014] Allessie, M. and de Groot, N. (2014). Crosstalk opposing view: Rotors have not been demonstrated to be the drivers of atrial fibrillation. *Journal of Physiology*, 592(15):3167–3170.

[Alonso and Bengtson, 2014] Alonso, A. and Bengtson, L. G. S. (2014). A rising tide: the global epidemic of atrial fibrillation. *Circulation*, 129(8):829–830.

[Altman and Dittmer, 1971] Altman, P. and Dittmer, D. (1971). Respiration and circulation. *Technical Report, Federation of American Societies for Experimental Biology Bethesda MD.*

[Amsallem et al., 2009] Amsallem, D., Cortial, J., Carlberg, K., and Farhat, C. (2009). A method for interpolating on manifolds structural dynamics reduced-order models. *International Journal for Numerical Methods in Engineering*, 80(9):1241–1258.

[Amsallem and Haasdonk, 2016] Amsallem, D. and Haasdonk, B. (2016). PEBL-ROM: Projection-error based local reduced-order models. *Advanced Modeling and Simulation in Engineering Sciences*, 3(1):6.

[Amsallem et al., 2012] Amsallem, D., Zahr, M. J., and Farhat, C. (2012). Nonlinear model order reduction based on local reduced-order bases. *International Journal for Numerical Methods in Engineering*, 92(10):891–916.

[Amsallem et al., 2015] Amsallem, D., Zahr, M. J., and Washabaugh, K. (2015). Fast local reduced basis updates for the efficient reduction of nonlinear systems with hyper-reduction. *Advances in Computational Mathematics*, 41(5):1187–1230.

[Andreotti et al., 2017] Andreotti, F., Carr, O., Pimentel, M. A. F., Mahdi, A., and De Vos, M. (2017). Comparing feature-based classifiers and convolutional neural networks to detect arrhythmia from short segments of ECG. *2017 Computing in Cardiology (CinC)*, pages 1–4.

[Antipov et al., 2017] Antipov, G., Baccouche, M., and Dugelay, J. (2017). Face aging with conditional generative adversarial networks. *Proceedings of the IEEE International Conference on Image Processing (ICIP)*.

[Arevalo et al., 2016] Arevalo, H., Vadakkumpadan, F., Guallar, E., Jebb, A., Malamas, P., Wu, K., and Trayanova, N. (2016). Arrhythmia risk stratification of patients after myocardial infarction using personalized heart models. *Nature Communications*, 7.

[Ayed et al., 2019] Ayed, I., Cedilnik, N., Gallinari, P., and Sermesant, M. (2019). EP-Net: Learning cardiac electrophysiology models for physiology-based constraints in data-driven predictions. *FIMH 2019 - 10th International Conference on Functional Imaging of the Hearth*, pages 55–63.

[Ballarin et al., 2016] Ballarin, F., Faggiano, E., Ippolito, S., Manzoni, A., Quarteroni, A., Rozza, G., and Scrofani, R. (2016). Fast simulations of patient-specific haemodynamics of coronary artery bypass grafts based on a POD-Galerkin method and a vascular shape parametrization. *Journal of Computational Physics*, 315:609–628.

[Ballarin et al., 2015] Ballarin, F., Manzoni, A., Quarteroni, A., and Rozza, G. (2015). Supremizer stabilization of POD-Galerkin approximation of parametrized steady incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Engineering*, 102(5):1136–1161.

[Barrault et al., 2004] Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T. (2004). An ‘empirical interpolation’ method: Application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathématique de l’Académie des Sciences*, 339(9):667–672.

[Bartezzaghi, 2014] Bartezzaghi, A. (2014). isoGlib: an Isogeometric Analysis library for the solution of high-order partial differential equations on surfaces. *PDESoft*.

[Baydin et al., 2018] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, B. A. (2018). Automatic differentiation in machine learning: a survey. *arXiv preprint arXiv:1502.05767*.

[Bendahmane et al., 2010] Bendahmane, M., Bürger, R., and Ruiz-Baier, R. (2010). A multiresolution space-time adaptive scheme for the bidomain model in electrocardiology. *Numerical Methods for Partial Differential Equations*, 26(6):1377–1404.

[Benner et al., 2017] Benner, P., Cohen, A., Ohlberger, M., and Willcox, K. (2017). *Model reduction and approximation: Theory and algorithms*. SIAM.

[Benner et al., 2015] Benner, P., Gugercin, S., and Willcox, K. (2015). A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review 57*, 4:483–531.

[Bērziņš et al., 2020] Bērziņš, A., Helmig, J., Key, F., and Elgeti, S. (2020). Standardized non-intrusive reduced order modeling using different regression models with application to complex flow problems. *arXiv preprint arXiv:2006.13706v1*.

[Bhattacharya et al., 2020] Bhattacharya, K., Hosseini, B., Kovachki, N. B., and Stuart, A. (2020). Model reduction and neural networks for parametric PDEs. *arXiv preprint arXiv:2005.03180*.

[Bonomi et al., 2017] Bonomi, D., Manzoni, A., and Quarteroni, A. (2017). A matrix DEIM technique for model reduction of nonlinear parametrized problems in cardiac mechanics. *Computer Methods in Applied Mechanics and Engineering*, 324:300–326.

[Bourlard and Kamp, 1998] Bourlard, H. and Kamp, Y. (1998). Auto-association by multi-layer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294.

[Bourlard and Wellekens, 1989] Bourlard, H. and Wellekens, C. (1989). Speech pattern discrimination and multi-layered perceptrons. *Computer Speech and Language*, 3:1–19.

[Boyle et al., 2018] Boyle, P., Hakim, J. B., Zahid, S., Franceschi, W. H., Murphy, M. J., Prakosa, A., Aronis, K., Zghaib, t., Balouch, M., Ipek, E. G., Chrispin, J., Berger, R., Ashikaga, H., Marine, J., Calkins, H., Nazarian, S., Spragg, D., and Trayanova, N. (2018). The fibrotic substrate in persistent atrial fibrillation patients: comparison between predictions from computational modeling and measurements from focal impulse and rotor mapping. *Frontiers in Physiology*, 9.

[Brooks and Lu, 1972] Brooks, C. and Lu, H. (1972). *The sinoatrial pacemaker of the heart*. Charles C. Thomas Publisher.

[Bucelli et al., 2020] Bucelli, M., Salvador, M., and Dedè, L. (2020). Multipatch isogeometric analysis for electrophysiology: simulation in a human heart. *MOX-Report No. 46/2020*.

[Bueno-Orovio et al., 2008] Bueno-Orovio, A., Cherry, E., and Fenton, F. (2008). Minimal model for human ventricular action potentials in tissue. *Journal of Theoretical Biology*, 253(3):544–560.

[Buffa et al., 2012] Buffa, A., Maday, Y., Patera, A. T., Prud'homme, C., and Turinici, G. (2012). A priori convergence of the greedy algorithm for the parametrized reduced basis method. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 46(3):595–603.

[Bui-Thanh et al., 2004] Bui-Thanh, T., Damodaran, M., and Willcox, K. (2004). Aerodynamicdata reconstruction and inverse design using proper orthogonal decomposition. *AIAA Journal*, 42(8):1505–1516.

[Bui-Thanh et al., 2008] Bui-Thanh, T., Willcox, K., and Ghattas, O. (2008). Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications. *AIAA Journal*, 46(10):2520–2529.

[Burman and Ern, 2003] Burman, E. and Ern, A. (2003). The discrete maximum principle for stabilized finite element methods. *Numerical Mathematics and Advanced Applications*, pages 557–566.

[Cagniart et al., 2019] Cagniart, N., Maday, Y., and Stamm, B. (2019). *Model order reduction for problems with large convection effects*, volume 47 of *Computational Methods in Applied Sciences*. Springer, Cham.

[Carlberg et al., 2013] Carlberg, K., Farhat, C., Cortial, J., and Amsallem, D. (2013). The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647.

[Carlberg et al., 2019] Carlberg, K. T., Jameson, A., Kochenderfer, M., Morton, J., Peng, L., and Witherden, F. (2019). Recovering missing CFD data for high-order discretizations using deep neural networks and dynamics learning. *arXiv preprint arXiv:1812.01177.*

[Cervi and Spiteri, 2019] Cervi, J. and Spiteri, R. J. (2019). A comparison of fourth-order operator splitting methods for cardiac simulations. *Applied Numerical Mathematics*, 145:227–235.

[Chatterjee, 2000] Chatterjee, A. (2000). An introduction to the proper orthogonal decomposition. *Current Science*, 78(7):808–817.

[Chaturantabut and Sorensen, 2010] Chaturantabut, S. and Sorensen, D. C. (2010). Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764.

[Chen et al., 2020] Chen, W., Wang, Q., Hesthaven, J. S., and Zhang, C. (2020). Physics-informed machine learning for reduced-order modeling of nonlinear problems. *Preprint.*

[Chung and Hulbert, 1993] Chung, J. and Hulbert, G. M. (1993). A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized-$\alpha$ method. *Journal of Applied Mechanics*, 60(2):371–375.

[Chung et al., 2017] Chung, J. S., Senior, A. W., Vinyals, O., and Zisserman, S. (2017). Lip reading sentences in the wild. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3444–3453.

[Clayton et al., 2011] Clayton, R., Bernus, O., Cherry, E., Dierckx, H., Fenton, F., Mirabella, L., Panfilov, A., Sachse, F., Seemann, G., and Zhang, H. (2011). Models of cardiac tissue electrophysiology: Progress, challenges and open questions. *Progress in Biophysics and Molecular Biology*, 104(1):22–48. Cardiac physiome project: Mathematical and modelling foundations.

[Clayton and Taggart, 2005] Clayton, R. and Taggart, P. (2005). Regional differences in apd restitution can initiate wavebreak and re-entry in cardiac tissue: A computational study. *BioMedical Engineering OnLine*, 4(54).

[Clayton et al., 2020] Clayton, R. H., Aboelkassem, Y., Cantwell, C. D., Corrado, C., Delhaas, T., Huberts, W., Lei, C. L., Ni, H., Panfilov, A. V., Roney, C., and dos Santos, R. W. (2020). An audit of uncertainty in multi-scale cardiac electrophysiology models. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2173):20190335.

[Clevert et al., 2015] Clevert, D., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289.*

[Clifford et al., 2017] Clifford, G. D., Liu, C., Moody, B., Lehman, L. H., Silva, I., Li, Q., Johnson, A. E., and Mark, R. G. (2017). AF classification from a short single lead ECG recording: The PhysioNet/computing in cardiology challenge 2017. pages 1–4.

[Colciago et al., 2014] Colciago, C. M., Deparis, S., and Quarteroni, A. (2014). Comparisons between reduced order models and full 3D models for fluid-structure interaction problems in haemodynamics. *Journal of Computational and Applied Mathematics*, 265:120–138. Current Trends and Progresses in Scientific Computation.

[Colli Franzone et al., 2005] Colli Franzone, P., Pavarino, L., and Taccardi, B. (2005). Simulating patterns of excitation, repolarization and action potential duration with cardiac bidomain and monodomain models. *Mathematical Biosciences*, 197(1):35–66.

[Colli Franzone and Pavarino, 2004] Colli Franzone, P. and Pavarino, L. F. (2004). A parallel solver for reaction–diffusion systems in computational electrocardiology. *Mathematical Models and Methods in Applied Sciences*, 14(06):883–911.

[Colli Franzone et al., 2006] Colli Franzone, P., Pavarino, L. F., and Savaré, G. (2006). *Computational electrocardiology: Mathematical and numerical modeling*. Springer Milan.

[Colli Franzone et al., 2014] Colli Franzone, P., Pavarino, L. F., and Scacchi, S. (2014). *Mathematical cardiac electrophysiology*, volume 13 of *Modeling, Simulation & Applications*. Springer.

[Cottrell et al., 2009] Cottrell, J. A., Hughes, T. J. R., and Bazilevs, Y. (2009). *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley Publishing, 1st edition.

[Courtemanche et al., 1998] Courtemanche, M., Ramirez, R. J., and Nattel, S. (1998). Ionic mechanisms underlying human atrial action potential properties: Insights from a mathematical model. *American Journal of Physiology-Heart and Circulatory Physiology*, 275(1):301–321.

[Coveney et al., 2020] Coveney, S., Corrado, C., Roney, C. H., O'Hare, D., Williams, S. E., O'Neill, M. D., Niederer, S. A., Clayton, R. H., Oakley, J. E., and Wilkinson, R. D. (2020). Gaussian process manifold interpolation for probabilistic atrial activation maps and uncertain conduction velocity. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2173):20190345.

[Dal Santo et al., 2019] Dal Santo, N., Deparis, S., Manzoni, A., and Quarteroni, A. (2019). An algebraic least squares reduced basis method for the solution of nonaffinely parametrized Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 344:186–208.

[Dal Santo et al., 2020] Dal Santo, N., Deparis, S., and Pegolotti, L. (2020). Data driven approximation of parametrized PDEs by reduced basis and neural networks. *Journal of Computational Physics*, 416:109550.

[de Boor, 1972] de Boor, C. (1972). On calculating with B-splines. *Journal of Approximation Theory*, 6(1):50–62.

[Dedè et al., 2015] Dedè, L., Jäggli, C., and Quarteroni, A. (2015). Isogeometric numerical dispersion analysis for two-dimensional elastic wave propagation. *Computer Methods in Applied Mechanics and Engineering*, 284:320–348. Isogeometric Analysis Special Issue.

[Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Dhamala et al., 2018] Dhamala, J., Arevalo, H. J., Sapp, J., Horácek, B. M., Wu, K. C., Trayanova, N. A., and Wang, L. (2018). Quantifying the uncertainty in model parameters using Gaussian process-based Markov chain Monte Carlo in cardiac electrophysiology. *Medical Image Analysis*, 48:43–57.

[Drineas et al., 2006] Drineas, P., Kannan, R., and Mahoney, M. W. (2006). Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36:158–183.

[Dzialowski and Crossley, 2015] Dzialowski, E. M. and Crossley, D. A. (2015). *The Cardiovascular System*. Academic Press, San Diego, sixth edition.

[Farhat et al., 2020] Farhat, C., Grimberg, S., Manzoni, A., and Quarteroni, A. (2020). Computational bottlenecks for PROMs: Pre-computation and hyperreduction.

[Feeny et al., 2020] Feeny, A. K., Chung, M. K., Madabhushi, A., Attia, Z. I., Cikes, M., Firouznia, M., Friedman, P. A., Kalscheur, M. M., Kapa, S., Narayan, S. M., Noseworthy, P. A., Passman, R. S., Perez, M. V., Peters, N. S., Piccini, J. P., Tarakji, K. G., Thomas, S. A., Trayanova, N. A., Turakhia, M. P., and Wang, P. J. (2020). Artificial intelligence and machine learning in arrhythmias and cardiac electrophysiology. *Circulation: Arrhythmia and Electrophysiology*, 13(8):e007952.

[Fink and Rheinboldt, 1984] Fink, J. P. and Rheinboldt, W. C. (1984). Solution manifolds and submanifolds of parametrized equations and their discretization errors. *Numerische Mathematik*, 45(3):323–343.

[FitzHugh, 1961] FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1(6):445–466.

[Forti and Dedè, 2015] Forti, D. and Dedè, L. (2015). Semi-implicit BDF time discretization of the navier-stokes equations with VMS-LES modeling in a high performance computing framework. *Computers & Fluids*, 117(0):168–182.

[Freno and Carlberg, 2018] Freno, B. A. and Carlberg, K. T. (2018). Machine-learning error models for approximate solutions to parameterized systems of nonlinear equations. *arXiv preprint arXiv:1808.02097*.

[Fresca et al., 2020a] Fresca, S., Dedè, L., and Manzoni, A. (2020a). A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. *arXiv preprint arXiv:2001.04001*.

[Fresca and Manzoni, 2021] Fresca, S. and Manzoni, A. (2021). POD DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. *arXiv preprint arXiv:2101.11845*.

[Fresca et al., 2020b] Fresca, S., Manzoni, A., Dedè, L., and Quarteroni, A. (2020b). Deep learning-based reduced order models in cardiac electrophysiology. *PLOS ONE*, 15(10):1–32.

[García-Cosío et al., 2012] García-Cosío, F., Pastor Fuentes, A., and Núñez Angulo, A. (2012). Clinical approach to atrial tachycardia and atrial flutter from an understanding of the mechanisms. electrophysiology based on anatomy. *Revista Española de Cardiología (English Edition)*, 65(4):363–375.

[Garotta et al., 2020] Garotta, F., Demo, N., Tezzele, M., Carraturo, M., Reali, A., and Rozza, G. (2020). *Reduced order isogeometric analysis approach for PDEs in parametrized domains*. Springer International Publishing.

[Gerardo-Giorda, 2007] Gerardo-Giorda, L. (2007). Modeling and numerical simulation of action potential patterns in human atrial tissues. *Preprint hal-00132706*.

[Gerbeau and Lombardi, 2014] Gerbeau, J. and Lombardi, D. (2014). Approximated Lax pairs for the reduced order integration of nonlinear evolution equations. *Journal of Computational Physics*, 265:246–269.

[Geselowitz and Miller III, 1983] Geselowitz, D. B. and Miller III, W. T. (1983). A bidomain model for anisotropic cardiac muscle. *Annals of Biomedical Engineering*, 11(3-4):191–206.

[Glorot and Bengio, 2010] Glorot, S. and Bengio, B. (2010). Understanding the difficulty of training deep feedforward neural networks. *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics.*

[González and Balajewicz, 2018] González, F. J. and Balajewicz, M. (2018). Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. *arXiv preprint arXiv:1808.01346.*

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning.* MIT Press.

[Grepl and Patera, 2010] Grepl, M. A. and Patera, A. T. (2010). A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 39(1):157–181.

[Guo and Hesthaven, 2018] Guo, M. and Hesthaven, J. S. (2018). Reduced order modeling for nonlinear structural analysis using Gaussian process regression. *Computer Methods in Applied Mechanics and Engineering*, 341:807–826.

[Guo and Hesthaven, 2019] Guo, M. and Hesthaven, J. S. (2019). Data-driven reduced order modeling for time-dependent problems. *Computer Methods in Applied Mechanics and Engineering*, 345:75–99.

[Gurtin, 1982] Gurtin, M. E. (1982). *An introduction to continuum mechanics*, volume 158. Academic press.

[Göktepe et al., 2010] Göktepe, S., Wong, J., and Kuhl, E. (2010). Atrial and ventricular fibrillation: Computational simulation of spiral waves in cardiac tissue. *Archive of Applied Mechanics*, 80:569–580.

[Haghighat et al., 2020] Haghighat, E., Raissi, M., Moure, A., Gomez, H., and Juanes, R. (2020). A deep learning framework for solution and discovery in solid mechanics. *arXiv preprint arXiv:2003.02751.*

[Halko et al., 2011] Halko, N., Martinsson, P., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53:217–288.

[Han et al., 2017] Han, J., Jentzen, A., and E, W. (2017). Solving high-dimensional partial differential equations using deep learning. *arXiv preprint arXiv:1707.02568.*

[Hastie et al., 2001] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The elements of statistical learning.* Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.

[Hesthaven and Ubbiali, 2018] Hesthaven, J. and Ubbiali, S. (2018). Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78.

[Hinton and Zemel, 1994] Hinton, G. E. and Zemel, R. S. (1994). Autoencoders, minimum description length, and Helmholtz free energy. *Proceedings of the 6th International Conference on Neural Information Processing Systems (NIPS'1993).*

[Hu et al., 2013] Hu, Y., Gurev, V., Constantino, J., Bayer, J., and Trayanova, N. (2013). Effects of mechano-electric feedback on scroll wave stability in human ventricular fibrillation. *PLOS ONE*, 8.

[Hurtado et al., 2017] Hurtado, D., Castro, S., and Madrid, P. (2017). Uncertainty quantification of two models of cardiac electromechanics. *International Journal for Numerical Methods in Biomedical Engineering*, 33(12):e2894.

[Iollo and Lombardi, 2014] Iollo, A. and Lombardi, D. (2014). Advection modes by optimal mass transfer. *Physical Review E*, 89:022923.

[Itchhaporia et al., 1996] Itchhaporia, D., Snow, P. B., Almassy, R. J., and Oetgen, W. J. (1996). Artificial neural networks: Current status in cardiovascular medicine. *Journal of the American College of Cardiology*, 28(2):515–521.

[Iwasaki et al., 2011] Iwasaki, Y., Nishida, K., Kato, T., and Nattel, S. (2011). Atrial fibrillation pathophysiology. *Circulation*, 124(20):2264–2274.

[Jarvik, 2004] Jarvik, R. (2004). The total artificial heart. *Scientific American*, 244(1):74–81.

[Johnston et al., 2018] Johnston, B. M., Coveney, S., Chang, E. T., Johnston, P. R., and Clayton, R. H. (2018). Quantifying the effect of uncertainty in input parameters in a simplified bidomain model of partial thickness ischaemia. *Medical & Biological Engineering & Computing*, 56(5):761–780.

[Johnstone et al., 2016] Johnstone, R. H., Chang, E. T., Bardenet, R., de Boer, T. P., Gavaghan, D. J., Pathmanathan, P., Clayton, R. H., and Mirams, G. R. (2016). Uncertainty and variability in models of the cardiac action potential: Can we build trustworthy models? *Journal of Molecular and Cellular Cardiology*, 96:49–62. Special Issue: Computational Modelling of the Heart.

[Kani and Elsheikh, 2017] Kani, J. N. and Elsheikh, A. H. (2017). DR-RNN: A deep residual recurrent neural network for model reduction. *arXiv preprint arXiv:1709.00939*.

[Kani and Elsheikh, 2018] Kani, J. N. and Elsheikh, A. H. (2018). Reduced-order modeling of subsurface multi-phase flow models using deep residual recurrent neural networks. *Transport in Porous Media*, 126(3):713–741.

[Kannel et al., 1982] Kannel, W. B., Abbott, R. D., Savage, D. D., and McNamara, P. M. (1982). Epidemiologic features of chronic atrial fibrillation. *New England Journal of Medicine*, 306(17):1018–1022.

[Karma, 1993] Karma, A. (1993). Spiral breakup in model equations of action potential propagation in cardiac tissue. *Physical review letters*, 71(7):1103–1106.

[Karma, 2013] Karma, A. (2013). Physics of cardiac arrhythmogenesis. *Annual Review of Condensed Matter Physics*, 4(1):313–337.

[Kast et al., 2020] Kast, M., Guo, M., and Hesthaven, J. S. (2020). A non-intrusive multifidelity method for the reduced order modeling of nonlinear problems. *Computer Methods in Applied Mechanics and Engineering*, 364:112947.

[Kim et al., 2020] Kim, Y., Choi, Y., WIDEMANN, D., and ZOHDI, T. (2020). A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *arXiv preprint arXiv:2009.11990*.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). *Adam: A method for stochastic optimization.* International Conference on Learning Representations (ICLR).

[Klabunde, 2011] Klabunde, R. (2011). *Cardiovascular Physiology Concepts.* Wolters Kluwer Health/Lippincott Williams & Wilkins.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'2012)*, 1:1097–1105.

[Kutyniok et al., 2019] Kutyniok, G., Petersen, P., Raslan, M., and Schneider, R. (2019). A theoretical analysis of deep neural networks and parametric PDEs. *arXiv preprint arXiv:1904.00377*.

[Laakmann and Petersen, 2020] Laakmann, F. and Petersen, P. (2020). Efficient approximation of solutions of parametric linear transport equations by ReLU DNNs. *arXiv preprint arXiv:2001.11441*.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient based learning applied to document recognition. *Proceedings of the IEEE*, pages 533–536.

[Lee and Carlberg, 2020] Lee, K. and Carlberg, K. (2020). *Journal of Computational Physics*, 404:108973.

[Lei et al., 2020] Lei, C. L., Ghosh, S., Whittaker, D. G., Aboelkassem, Y., Beattie, K. A., Cantwell, C. D., Delhaas, T., Houston, C., Novaes, G. M., Panfilov, A. V., Pathmanathan, P., Riabiz, M., dos Santos, R. W., Walmsley, J., Worden, K., Mirams, G. R., and Wilkinson, R. D. (2020). Considering discrepancy when calibrating a mechanistic electrophysiology model. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2173):20190349.

[Levrero-Florencio et al., 2020] Levrero-Florencio, F., Margara, F., Zacur, E., Bueno-Orovio, A., Wang, Z., Santiago, A., Aguado-Sierra, J., Houzeaux, G., Grau, V., Kay, D., Vázquez, M., Ruiz-Baier, R., and Rodriguez, B. (2020). Sensitivity analysis of a strongly-coupled human-based electromechanical cardiac model: Effect of mechanical parameters on physiologically relevant biomarkers. *Computer Methods in Applied Mechanics and Engineering*, 361:112762.

[Likas et al., 2003] Likas, A., Vlassis, N., and Verbeek, J. J. (2003). The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461.

[Longobardi et al., 2020] Longobardi, S., Lewalle, A., Coveney, S., Sjaastad, I., Espe, E. K. S., Louch, W. E., Musante, C. J., Sher, A., and Niederer, S. A. (2020). Predicting left ventricular contractile function via gaussian process emulation in aortic-banded rats. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2173):20190334.

[Lusch et al., 2018] Lusch, B., Kutz, J. N., and Brunton, S. (2018). Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9.

[Maday et al., 2008] Maday, Y., Nguyen, N., Patera, A., and Pau, G. (2008). A general multipurpose interpolation procedure: The magic points. *Communications on Pure and Applied Analysis*, 8:383–404.

[Manzoni et al., 2016] Manzoni, A., Pagani, S., and Lassila, T. (2016). Accurate solution of Bayesian inverse uncertainty quantification problems combining reduced basis methods and reduction error models. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):380–412.

[Manzoni et al., 2012] Manzoni, A., Quarteroni, A., and Rozza, G. (2012). Model reduction techniques for fast blood flow simulation in parametrized geometries. *International Journal for Numerical Methods in Biomedical Engineering*, 28(6-7):604–625.

[Manzoni et al., 2015] Manzoni, A., Salmoiraghi, F., and Heltai, L. (2015). Reduced basis isogeometric methods (RB-IGA) for the real-time simulation of potential flows about parametrized NACA airfoils. *Computer Methods in Applied Mechanics and Engineering*, 284:1147–1180.

[Mirams et al., 2016] Mirams, G., Pathmanathan, P., Gray, R., Challenor, P., and Clayton, R. (2016). Uncertainty and variability in computational and mathematical models of cardiac physiology. *The Journal of Physiology*, 594.23:6833–6847.

[Mitchell and Schaeffer, 2003] Mitchell, C. C. and Schaeffer, D. G. (2003). A two-current model for the dynamics of cardiac membrane. *Bulletin of Mathematical Biology*, 65(5):767–793.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

[Mohan and Gaitonde, 2018] Mohan, A. and Gaitonde, D. V. (2018). A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks. *arXiv preprint arXiv:1804.0926*.

[Morillo et al., 2017] Morillo, C., Banerjee, A., Perel, P., Wood, D., and Jouven, X. (2017). Atrial fibrillation: The current epidemic. *Journal of Geriatric Cardiology*, 14:195–203.

[Nagaiah et al., 2013] Nagaiah, C., Kunisch, K., and Plank, G. (2013). Optimal control approach to termination of re-entry waves in cardiac electrophysiology. *Journal of Mathematical Biology*, 67(2):359–388.

[Nagumo et al., 1962] Nagumo, J., Arimoto, S., and Yoshizawa, S. (1962). An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070.

[Narayan and Jalife, 2014] Narayan, S. M. and Jalife, J. (2014). Crosstalk proposal: Rotors have been demonstrated to drive human atrial fibrillation. *The Journal of Physiology*, 592(15):3163–3166.

[Narayan et al., 2012a] Narayan, S. M., Krummen, D. E., Shivkumar, K., Clopton, P., Rappel, W.-J., and Miller, J. M. (2012a). Treatment of atrial fibrillation by the ablation of localized sources confirm (conventional ablation for atrial fibrillation with or without focal impulse and rotor modulation) trial. *Journal of the American College of Cardiology*, 60(7):628–636.

[Narayan et al., 2012b] Narayan, S. M., Patel, J., Mulpuru, S., and Krummen, D. E. (2012b). Focal impulse and rotor modulation ablation of sustaining rotors abruptly terminates persistent atrial fibrillation to sinus rhythm with elimination on follow-up: A video case study. *Heart Rythym*, 9(9):1436–1439.

[Nash and Panfilov, 2004] Nash, M. P. and Panfilov, A. V. (2004). Electromechanical model of excitable tissue to study reentrant cardiac arrhythmias. *Progress in Biophysics and Molecular Biology*, 85:501–522.

[Nattel, 2002] Nattel, S. (2002). New ideas about atrial fibrillation 50 years on. *Nature*, 415:219–226.

[Negri et al., 2015] Negri, F., Manzoni, A., and Amsallem, D. (2015). Efficient model reduction of parametrized systems by matrix discrete empirical interpolation. *Journal of Computational Physics*, 303:431 – 454.

[Neic et al., 2017] Neic, A., Campos, F. O., Prassl, A. J., Niederer, S. A., Bishop, M. J., Vigmond, E. J., and Plank, G. (2017). Efficient computation of electrograms and ecgs in human whole heart simulations using a reaction-eikonal model. *Journal of Computational Physics*, 346:191–211.

[Niederer et al., 2011] Niederer, S., Mitchell, L., Smith, N., and Plank, G. (2011). Simulating human cardiac electrophysiology on clinical time-scales. *Frontiers in Physiology*, 2.

[Niederer et al., 2020] Niederer, S. A., Aboelkassem, Y., Cantwell, C. D., Corrado, C., Coveney, S., Cherry, E. M., Delhaas, T., Fenton, F. H., Panfilov, A. V., Pathmanathan, P., Plank, G., Riabiz, M., Roney, C. H., dos Santos, R. W., and Wang, L. (2020). Creation and application of virtual patient cohorts of heart models. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2173):20190558.

[Niederer et al., 2009] Niederer, S. A., Fink, M., Noble, D., and Smith, N. P. (2009). A meta-analysis of cardiac electrophysiology computational models. *Experimental Physiology*, 94(5):486–495.

[Nishida and Nattel, 2014] Nishida, K. and Nattel, S. (2014). Atrial fibrillation compendium. *Circulation Research*, 114(9):1447–1452.

[Ogawa et al., 2018] Ogawa, H., An, Y., Ikeda, S., Aono, Y., Doi, K., Ishii, M., Iguchi, M., Masunaga, N., Esato, M., Tsuji, H., Wada, H., Hasegawa, K., Abe, M., Lip, G. Y., Akao, M., and null null (2018). Progression from paroxysmal to sustained atrial fibrillation is associated with increased adverse events. *Stroke*, 49(10):2301–2308.

[Ogden, 1997] Ogden, R. W. (1997). *Non-linear elastic deformations*. Courier Corporation.

[Ohlberger and Rave, 2016] Ohlberger, M. and Rave, S. (2016). Reduced basis methods: Success, limitations and future challenges. *Proceedings of ALGORITMY*, pages 1–12.

[Opie, 2004] Opie, L. (2004). *Heart physiology: from cell to circulation*. Lippincott Williams & Wilkins.

[Opschoor et al., 2020] Opschoor, J. A. A., Petersen, P. C., and Schwab, C. (2020). Deep ReLU networks and high-order finite element methods. *Analysis and Applications*, 18(05):715–770.

[Pagani et al., 2019] Pagani, S., Manzoni, A., and Carlberg, K. (2019). Statistical closure modeling for reduced-order models of stationary systems by the ROMES method. *arXiv preprint arXiv:1901.02792*.

[Pagani et al., 2018] Pagani, S., Manzoni, A., and Quarteroni, A. (2018). Numerical approximation of parametrized problems in cardiac electrophysiology by a local reduced basis method. *Computer Methods in Applied Mechanics and Engineering*, 340:530–558.

[Panfilov and Holden, 1990] Panfilov, A. and Holden, A. (1990). Self-generation of turbulent vortices in a two-dimensional model of cardiac tissue. *Physics Letters A*, 151(1):23 – 26.

[Parish and Carlberg, 2019] Parish, E. and Carlberg, K. (2019). Time-series machine-learning error models for approximate solutions to parameterized dynamical systems. *arXiv preprint arXiv:1907.11822*.

[Patelli et al., 2017] Patelli, A. S., Dedè, L., Lassila, T., Bartezzaghi, A., and Quarteroni, A. (2017). Isogeometric approximation of cardiac electrophysiology models on surfaces: An accuracy study with application to the human left atrium. *Computer Methods in Applied Mechanics and Engineering*, 317:248–273.

[Pathak et al., 2018a] Pathak, J., Hunt, B., Girvan, M., Lu, Z., and Ott, E. (2018a). Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach. *Physical Review Letters*, 120:024102.

[Pathak et al., 2018b] Pathak, J., Wikner, A., Fussell, R., Chandra, S., Hunt, B., Girvan, M., and Ott, E. (2018b). Hybrid forecasting of chaotic processes: using machine learning in conjunction with a knowledge-based model. *Chaos*, 28(4):041101.

[Pathmanathan et al., 2012] Pathmanathan, P., Bernabeu, M., Niederer, S., Gavaghan, D., and Kay, D. (2012). Computational modelling of cardiac electrophysiology: Explanation of the variability of results from different numerical solvers. *International Journal for Numerical Methods in Biomedical Engineering*, 28(8):890–903.

[Pathmanathan et al., 2010] Pathmanathan, P., Chapman, S., Gavaghan, D., and Whiteley, J. (2010). Cardiac electromechanics: the effect of contraction model on the mathematical problem and accuracy of the numerical scheme. *The Quarterly Journal of Mechanics and Applied Mathematics*, 27(11):1751–1770.

[Pathmanathan et al., 2019] Pathmanathan, P., Cordeiro, J. M., and Gray, R. A. (2019). Comprehensive uncertainty quantification and sensitivity analysis for cardiac action potential models. *Frontiers in Physiology*, 10.

[Pegolotti et al., 2019] Pegolotti, L., Dedè, L., and Quarteroni, A. (2019). Isogeometric analysis of the electrophysiology in the human heart: Numerical simulation of the bidomain equations on the atria. *Computer Methods in Applied Mechanics and Engineering*, 343:52–73.

[Peherstorfer, 2018] Peherstorfer, B. (2018). Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling. *arXiv preprint arXiv:1812.02094*.

[Peherstorfer et al., 2014] Peherstorfer, B., Butnaru, D., Willcox, K., and Bungartz, H. (2014). Localized discrete empirical interpolation method. *SIAM Journal of Scientific Computing*, 36.

[Petersen and Voigtlaender, 2018] Petersen, P. and Voigtlaender, F. (2018). Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks*, 108:296–330.

[Plank et al., 2008] Plank, G., Zhou, L., Greenstein, J., Cortassa, S., Winslow, R., O'Rourke, B., and Trayanova, N. A. (2008). From mitochondrial ion channels to arrhythmias in the heart: Computational techniques to bridge the spatio-temporal scales. *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences*, 366:3381–409.

[Plonsey and Barr, 1988] Plonsey, R. and Barr, R. C. (1988). *Bioelectricity, A Quantitative Approach.*

[Prakosa et al., 2018] Prakosa, A., Arevalo, H. J., Deng, D., Boyle, P. M., Nikolov, P. P., Ashikaga, H., Blauer, J. J. E., Ghafoori, E., Park, C. J., Blake, R. C., Han, F. T., MacLeod, R. S., Halperin, H. R., Callans, D. J., Ranjan, R., Chrispin, J., Nazarian, S., and Trayanova, N. A. (2018). Personalized virtual-heart technology for guiding the ablation of infarct-related ventricular tachycardia. *Nature Biomedical Engineering*, 2(10):732–740.

[Prud'homme et al., 2001] Prud'homme, C., Rovas, D. V., Veroy, K., Machiels, L., Maday, Y., Patera, A. T., and Turinici, G. (2001). Reliable real-time solution of parametrized partial differential equations: reduced-basis output bound methods. *Journal of Fluids Engineering*, 124(1):70–80.

[Pulch and Youssef, 2020] Pulch, R. and Youssef, M. (2020). Machine learning for trajectories of parametric nonlinear dynamical systems. *Journal of Machine Learning for Modeling and Computing*, 1(1):75–95.

[Pyakillya et al., 2017] Pyakillya, B., Kazachenko, N., and Mikhailovsky, N. (2017). Deep learning for ECG classification. *Journal of Physics: Conference Series.*

[Qu et al., 1999] Qu, Z., Weiss, J. N., and Garfinkel, A. (1999). Cardiac electrical restitution properties and stability of reentrant spiral waves: A simulation study. *American Journal of Physiology-Heart and Circulatory Physiology*, 276(1):269–283.

[Qu et al., 2010] Qu, Z., Xie, Y., Garfinkel, A., and Weiss, J. (2010). T-wave alternans and arrhythmogenesis in cardiac diseases. *Frontiers in Physiology*, 1.

[Quaglino et al., 2018] Quaglino, A., Pezzuto, S., Koutsourelakis, P.-S., Auricchio, A., and Krause, R. (2018). Fast uncertainty quantification of activation sequences in patient-specific cardiac electrophysiology meeting clinical time constraints. *International Journal for Numerical Methods in Biomedical Engineering*, 34(7):e2985.

[Quarteroni, 2013] Quarteroni, A. (2013). *Numerical Models for Differential Problems.* Springer Publishing Company, Incorporated, 2nd edition.

[Quarteroni et al., 2019] Quarteroni, A., Dedè, L., Manzoni, A., and Vergara, C. (2019). *Mathematical modelling of the human cardiovascular system: Data, numerical approximation, clinical applications.* Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.

[Quarteroni et al., 2016] Quarteroni, A., Manzoni, A., and Negri, F. (2016). *Reduced basis methods for partial differential equations: An introduction*, volume 92. Springer.

[Quarteroni et al., 2017] Quarteroni, A., Manzoni, A., and Vergara, C. (2017). The cardiovascular system: Mathematical modeling, numerical algorithms, clinical applications. *Acta Numerica*, 26:365–590.

[Quarteroni et al., 2008] Quarteroni, A., Sacco, R., and Saleri, F. (2008). *Matematica Numerica.* Springer Milan.

[Quarteroni and Valli, 1994] Quarteroni, A. and Valli, A. (1994). *Numerical approximation of partial differential equations*, volume 23. Springer.

[Raissi, 2018] Raissi, M. (2018). Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, 19:1–24.

[Raissi and Karniadakis, 2018] Raissi, M. and Karniadakis, G. E. (2018). Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141.

[Raissi et al., 2017a] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017a). Physics informed deep learning (Part I): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561.*

[Raissi et al., 2017b] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017b). Physics informed deep learning (Part II): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566.*

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

[Rajpurkar et al., 2017] Rajpurkar, P., Hannun, A. Y., Haghpanahi, M., Bourn, C., and Ng, A. Y. (2017). Cardiologist-level arrhythmia detection with convolutional neural networks. *arXiv preprint arXiv:1707.01836.*

[Regazzoni et al., 2019] Regazzoni, F., Dedè, L., and Quarteroni, A. (2019). Machine learning for fast and reliable solution of time-dependent differential equations. *Journal of Computational Physics*, 397.

[Reiss et al., 2018] Reiss, J., Schulze, P., Sesterhenn, J., and Mehrmann, V. (2018). The shifted proper orthogonal decomposition: A mode decomposition for multiple transport phenomena. *SIAM Journal on Scientific Computing*, 40(3):1322–1344.

[Rinaldi et al., 2015] Rinaldi, M., Dedè, L., Quarteroni, A., and Negri, F. (2015). Reduced Basis method for Isogeometric Analysis: Application to structural problems. *Master Thesis.*

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407.

[Rogers and McCulloch, 1994] Rogers, J. M. and McCulloch, A. D. (1994). A collocation-galerkin finite element model of cardiac action potential propagation. *IEEE Transactions on Biomedical Engineering*, 41(8):743–757.

[Rossi et al., 2014] Rossi, S., Lassila, T., Ruiz Baier, R., Sequeira, A., and Quarteroni, A. (2014). Thermodynamically consistent orthotropic activation model capturing ventricular systolic wall thickening in cardiac electromechanics. *European Journal of Mechanics - A/Solids*, 48:129–142.

[Rozza et al., 2013] Rozza, G., Huynh, D., and Manzoni, A. (2013). Reduced basis approximation and a posteriori error estimation for Stokes flows in parametrized geometries: Roles of the inf-sup stability constants. *Numerische Mathematik*, 125(1):115–152.

[Rozza et al., 2008] Rozza, G., Huynh, D., and Patera, A. (2008). Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *ARCME-Archives of Computional Methods in Engineering*, 15(3):229–275.

[Rozza and Veroy, 2007] Rozza, G. and Veroy, K. (2007). On the stability of the reduced basis method for stokes equations in parametrized domains. *Computer Methods in Applied Mechanics and Engineering*, 196(7):1244–1260.

[Rumelhart et al., 1986] Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, pages 533–536.

[Russell and Norvig, 2009] Russell, S. and Norvig, P. (2009). *Artificial intelligence: A modern approach.* Prentice Hall Press, USA, 3rd edition.

[Sachetto Oliveira et al., 2018] Sachetto Oliveira, R., Martins Rocha, B., Burgarelli, D., Meira Jr., W., Constantinides, C., and Weber Dosa Santos, R. (2018). Performance evaluation of gpu parallelization, space-time adaptive algorithms, and their combination for simulating cardiac electrophysiology. *International Journal for Numerical Methods in Biomedical Engineering*, 34(2).

[Saha et al., 2018] Saha, M., Roney, C. H., Bayer, J. D., Meo, M., Cochet, H., Dubois, R., and Vigmond, E. J. (2018). Wavelength and fibrosis affect phase singularity locations during atrial fibrillation. *Frontiers in Physiology*, 9:1207.

[Sahli Costabal et al., 2020] Sahli Costabal, F., Yang, Y., Perdikaris, P., Hurtado, D. E., and Kuhl, E. (2020). Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8:42.

[Sakamoto et al., 2005] Sakamoto, S.-I., Nitta, T., Ishii, Y., Miyagi, Y., Ohmori, H., and Shimizu, K. (2005). Interatrial electrical connections: the precise location and preferential conduction. *Journal of Cardiovascular Electrophysiology*, 16(10):1077–1086.

[Salmoiraghi et al., 2016] Salmoiraghi, F., Ballarin, F., Heltai, L., and Rozza, G. (2016). Isogeometric analysis-based reduced order modelling for incompressible linear viscous flows in parametrized shapes. *Advanced Modeling and Simulation in Engineering Sciences*, 3(1).

[San and Maulik, 2018] San, O. and Maulik, R. (2018). Neural network closures for nonlinear model order reduction. *Advances in Computational Mathematics*, 44(6):1717–1750.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

[Stella et al., 2020] Stella, S., Vergara, C., Maines, M., Catanzariti, D., Africa, P., Demattè, C., Centonze, M., Nobile, F., Del Greco, M., and Quarteroni, A. (2020). Integration of maps of activation times in computational cardiac electrophysiology. *MOX-Report No. 19/2020.*

[Strocchi et al., 2020] Strocchi, M., Augustin, C. M., Gsell, M. A. F., Karabelas, E., Neic, A., Gillette, K., Razeghi, O., Prassl, A. J., Vigmond, E. J., Behar, J. M., Gould, J., Sidhu, B., Rinaldi, C. A., Bishop, M. J., Plank, G., and Niederer, S. A. (2020). A publicly available virtual cohort of four-chamber heart meshes for cardiac electro-mechanics simulations. *PLOS ONE*, 15(6):e0235145.

[Sundnes et al., 2009] Sundnes, J., Artebrant, R., Skavhaug, O., and Tveito, A. (2009). A second-order algorithm for solving dynamic cell membrane equations. *IEEE Transactions on Biomedical Engineering*, 56(10):2546–2548.

[Sundnes et al., 2006] Sundnes, J., Lines, G. T., Cai, X., Nielsen, B. F., Mardal, K., and Tveito, A. (2006). *Computing the Electrical Activity in the Heart.* Springer Berlin Heidelberg.

[Sundnes et al., 2007] Sundnes, J., Lines, G. T., Cai, X., Nielsen, B. F., Mardal, K., and Tveito, A. (2007). *Computing the electrical activity in the heart*, volume 1. Springer Science & Business Media.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'2014)*, 2:3104–3112.

[Szlam et al., 2014] Szlam, A., Kluger, Y., and Tygert, M. (2014). An implementation of a randomized algorithm for principal component analysis. *arXiv preprint arXiv:1412.3510v1.*

[Takeishi et al., 2017] Takeishi, N., Kawahara, Y., and Yairi, T. (2017). Learning Koopman invariant subspaces for dynamic mode decomposition. *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'2017)*, page 1130–1140.

[Taylor and Stone, 2009] Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685.

[ten Tusscher, 2004] ten Tusscher, K. (2004). Spiral wave dynamics and ventricular arrhythmias. *PhD Thesis.*

[ten Tusscher and Panfilov, 2006] ten Tusscher, K. H. W. J. and Panfilov, A. V. (2006). Alternans and spiral breakup in a human ventricular tissue model. *American Journal of Physiology-Heart and Circulatory Physiology*, 291(3):1088–1100.

[Tencer and Potter, 2020] Tencer, J. and Potter, K. (2020). Enabling nonlinear manifold projection reduced-order models by extending convolutional neural networks to unstructured data. *arXiv preprint arXiv:2006.06154v1.*

[Tomek et al., 2019] Tomek, J., Bueno-Orovio, A., Passini, E., Zhou, X., Minchole, A., Britton, O., Bartolucci, C., Severi, S., Shrier, A., Virag, L., Varro, A., and Rodriguez, B. (2019). Development, calibration, and validation of a novel human ventricular myocyte model in health, disease, and drug block. *eLife*, 8:e48890.

[Trayanova, 2011] Trayanova, N. A. (2011). Whole-heart modeling applications to cardiac electrophysiology and electromechanics. *Circulation Research*, 108:113–28.

[Trehan et al., 2017] Trehan, S., Carlberg, K. T., and Durlofsky, L. J. (2017). Error modeling for surrogates of dynamical systems using machine learning. *International Journal for Numerical Methods in Engineering*, 112(12):1801–1827.

[Veroy and Patera, 2005] Veroy, K. and Patera, A. T. (2005). Certified real-time solution of the parametrized steady incompressible navier-stokes equations: rigorous reduced-basis a posteriori error bounds. *International Journal for Numerical Methods in Fluids*, 47:773–788.

[Vigmond et al., 2008] Vigmond, E., Dos Santos, R. W., Prassl, A., Deo, M., and Plank, G. (2008). Solvers for the cardiac bidomain equations. *Progress in Biophysics and Molecular Biology*, 96(1):3–18.

[Vigmond et al., 2002] Vigmond, E. J., Aguel, F., and Trayanova, N. A. (2002). Computational techniques for solving the bidomain equations in three dimensions. *IEEE Transactions on Biomedical Engineering*, 49(11):1260–1269.

[Wan et al., 2018] Wan, Z., Vlachas, P., Koumoutsakos, P., and Sapsis, T. (2018). Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PLOS ONE*, 13.

[Wang et al., 2019] Wang, Q., Hesthaven, J. S., and Ray, D. (2019). Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem. *Journal of Computational Physics*, 384:289–307.

[Wang et al., 2020] Wang, Q., Ripamonti, N., and Hesthaven, J. S. (2020). Recurrent neural network closure of parametric POD-galerkin reduced-order models based on the mori-zwanzig formalism. *Journal of Computational Physics*, 410:109402.

[Willcox, 2006] Willcox, K. (2006). Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. *Computers & Fluids*, 35(2):208–226.

[Willcox and Peraire, 2002] Willcox, K. and Peraire, J. (2002). Balanced model reduction via the proper orthogonal decomposition. *American Institute of Aeronautics and Astronautics Journal*, 40(11):2323–2330.

[Xia et al., 2018] Xia, Y., Wulan, N., Wang, K., and Zhang, H. (2018). Detecting atrial fibrillation by deep convolutional neural networks. *Computers in Biology and Medicine*, 93:84–92.

[Xiong et al., 2017] Xiong, Z., Stiles, M. K., and Zhao, J. (2017). Robust ECG signal classification for detection of atrial fibrillation using a novel neural network. *2017 Computing in Cardiology (CinC)*, pages 1–4.

[Yang and Perdikaris, 2018] Yang, Y. and Perdikaris, P. (2018). Physics-informed deep generative models. *arXiv preprint arXiv:11812.0351.*

[Yeo, 2017] Yeo, K. (2017). Model-free prediction of noisy chaotic time series by deep learning. *arXiv preprint arXiv:1710.01693*.

[Yosinski et al., 2014] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems 27*, 27:3320–3328.

[Zhu et al., 2017a] Zhu, S., Dedè, L., and Quarteroni, A. (2017a). Isogeometric analysis and proper orthogonal decomposition for parabolic problems. *Numerische Mathematik*, 135(2):333–370.

[Zhu et al., 2017b] Zhu, S., Dedè, L., and Quarteroni, A. (2017b). Isogeometric analysis and proper orthogonal decomposition for the acoustic wave equation. *Esaim-Mathematical Modelling And Numerical Analysis-Modelisation Mathematique Et Analyse Numerique*, 51(4):1197–1221.

[Zihlmann et al., 2017] Zihlmann, M., Perekrestenko, D., and Tschannen, M. (2017). Convolutional recurrent neural networks for electrocardiogram classification. *arXiv preprint arXiv:1710.06122*.