MOX-Report No. 07/2019

Hyper-reduced order models for parametrized unsteady Navier-Stokes equations on domains with variable shape

Dal Santo, N.; Manzoni, A.

Hyper-reduced order models for parametrized unsteady Navier-Stokes equations on domains with variable shape

Niccolò Dal Santo[1], Andrea Manzoni[2]

[1] CMCS, École Polytechnique Fédérale de Lausanne (EPFL),
Station 8, 1015 Lausanne, Switzerland.
[2]Mox-Laboratory for Modeling and Scientific Computing, Department of Mathematics,
Politecnico di Milano, P.za Leonardo da Vinci 32, 20133 Milano, Italy.

Abstract

In this work we set up a new, general and computationally efficient way to tackle parametrized fluid flows modeled through unsteady Navier-Stokes equations defined on domains with variable shape, when relying on the reduced basis method. We easily describe a domain by flexible boundary parametrizations, and generate domain (and mesh) deformations by means of a solid extension, obtained by solving an harmonic extension or a linear elasticity problem. The proposed procedure is built over a two-stages reduction: (i) first, we construct a reduced basis approximation for the mesh motion problem, irrespectively of the fluid flow problem we focus on; (ii) then, we generate a reduced basis approximation of the unsteady Navier-Stokes problem, relying on finite element snapshots evaluated over a set of reduced deformed configuration, and approximating both velocity and pressure fields simultaneously. To deal with unavoidable nonaffine parametric dependencies arising in both the mesh motion and the state problem, we apply a matrix version of the discrete empirical interpolation method, allowing to treat geometrical deformations in a non-intrusive, efficient and purely algebraic way. The same strategy is used to perform hyper-reduction of nonlinear terms. To assess the numerical performances of the proposed technique, we address the solution of parametrized fluid flows where the parameters describe both the shape of the domain, and relevant physical features. Complex flow patterns such as the ones appearing in a patient specific carotid bifurcation are accurately approximated, as well as derived quantities of potential clinical interest.

1 Introduction

The efficient numerical simulation of fluid flows is of paramount importance in several engineering fields. Blood dynamics in arterial vessels and aerodynamics are just two examples of contexts where scientific computing can provide quantitative indication about the physical behavior of a system, in view of its better understanding, control, forecasting. Solving these problems entails the numerical approximation of unsteady Navier-Stokes equations in three-dimensional domains, requiring fine computational meshes if complex flow patterns must be recovered, and ultimately yielding large-scale systems of equations to be solved.

Very often, such a problem depends on a set of input parameters, describing physical and/or geometrical features; the resulting system must then be solved for several different parameter values, each one corresponding to a different scenario. This is the case, for instance, of blood flow simulations, where outputs of clinical interest often must be evaluated for different flow conditions and in different geometrical configurations. Dealing with domains with variable shapes is indeed of interest, in this context, to *(i)* take into account natural intra-patients variability in vessel morphology, as well as to *(ii)* describe the evolution of diseases involving a narrowing of the vessel lumen, such as plaques, occlusions or stenosis. Characterizing several *virtual* scenarios may thus require several input/output evaluations, each one corresponding to a single query to the numerical model. If quantitative outputs are meant to support clinicians and medical doctors in their decisions, each new numerical simulation should be carried out very rapidly (order of minutes, say) on deployed platforms rather than on huge parallel hardware architectures, possibly requiring limited data storage and memory capacity. Meeting all these requirements is a challenging task, with the result that traditional high-fidelity, or full-order, techniques – e.g. the finite element (FE) method – are ill-suited, despite the constant growth of computer resources available.

Reduced-order models (ROMs) are emerging methodologies aimed at reducing the computational complexity and costs entailed by the repeated solution of PDE problems [42]. In the case of parametrized PDEs (i.e. PDEs depending on a vector of parameters), the reduced basis (RB) method is a remarkable example of a ROM that enables dramatic reduction of the dimension of the discrete problems arising from numerical approximation from millions to hundreds, or thousands at most, of variables.

Several works have addressed the construction of rapid and reliable ROMs for Navier-Stokes equations in the last two decades, mainly relying on the reduced basis (RB) method [55] and exploiting either proper orthogonal decomposition (POD) [27, 30, 25, 6, 58, 12, 5, 18] or greedy algorithms [43, 15, 29, 33, 61] for the construction of reduced order spaces. Moreover, we remark that other possibilities have been investigated, e.g. in [16], where proper generalized decomposition is applied to Stokes equations in two-dimensional parametrized geometries. The RB approximation of parametrized Navier-Stokes equations is an involved task because of the need of treating efficiently nonlinearities and parameter dependencies, approximating both velocity and pressure, and keeping error propagation in time under control. Last, but not least, describing complex geometrical variations when dealing with fluid flows defined over domains undergoing shape changes is not straightforward. Indeed, this feature highly impacts on computational efficiency, since: *(i)* parametrizing a set of shape is usually a complex, highly problem-dependent, task, and *(ii)* geometrical parametrizations imply strongly nonaffine parametric dependencies.

Despite all these aspects have been considered, separately, in different works, a flexible and computationally cheap way to treat the RB approximation of Navier-Stokes equations accounting for all these aspects is still missing. A new, efficient (and rather general) way to deal with fluid flows in domains with varying shapes addressing all the aspects mentioned above is proposed in this paper. We extend the state-of-the-art framework of RB methods for the treatment of the unsteady Navier-Stokes equations in nonaffinely parametrized geometries by employing a mesh motion technique to tackle the domain deformation and a waterfall of ROMs to deal at first with the computation of the domain displacement and then with the fluid flow. In order to gain the maximum efficiency, an hyper-reduction strategy to treat the nonaffine and nonlinear convective terms appearing in the NS equations is also devised, and applied for the first time to complex three-dimensional flows.

In particular, we describe domain deformations by flexible boundary parametrizations, and generate domain (and mesh) deformations by means of a solid extension, obtained by solving an harmonic extension (alternatively, a linear elasticity) problem, thus yielding a solid extension mesh moving technique. We highlight that this is a new approach to deal with PDEs defined on varying domains in the RB context, initially explored in [34] for the case of scalar, linear problems defined on simple two-dimensional domains. We point out that previous alternatives, such as volume-based parametrizations (induced by, e.g., free-form deformations or interpolations relying on radial basis functions) have been used to deal with relatively simple geometrical configurations [36, 35, 40] and recently extended to address complex shapes such as bifurcating arteries and multiple by-pass configurations [3, 4], nevertheless entailing a huge and troublesome work on the continuous formulation of the problem, as well as intrusive changes to its high-fidelity implementation. Moreover, selecting the number of control points, their position and admissible displacements is far from being trivial when dealing with complex three-dimensional shapes [47].

Extending the procedure introduced in [34], we propose a two-stages reduction: *(a)* first, we construct a RB approximation for the mesh motion problem, irrespectively of the fluid flow problem we focus on; *(b)* then, we generate a reduced basis approximation of the unsteady Navier-Stokes problem, exploiting a suitable enrichment of the velocity space to be able to approximate both velocity and pressure fields at the same time. To deal with the complex nonaffine parametric dependencies arising in both the mesh motion and the fluid flow problem, as well as with nonlinearities, we rely on a matrix version of the (discrete) empirical interpolation method (DEIM) allowing us to perform inexpensive evaluations of the online matrix operators for both the deformation and the state problem. In particular, we first recover an (approximate) affine parametric dependence in the high-fidelity arrays appearing in both problems, by applying matrix DEIM (MDEIM) and DEIM for matrix and vector operators, respectively. This is performed in a purely algebraic, black-box way, in order to overcome the application of the EIM on the continuous formulation of the problem, which is usually highly demanding, see e.g. [36, 35, 3, 4]. We then perform the RB approximation of both the deformation and the state problem, relying on a Galerkin-POD technique. This results in an extremely efficient, almost automatic (and less intrusive) framework capable to tackle fluid flows in complex parametrized shapes.

The structure of the paper is as follows. We introduce the parametrized formulation of unsteady Navier-Stokes equations we deal with in Sect. 2, as well as the high-fidelity, full-order model (FOM) and its algebraic counterpart. In Sect. 3 we describe the ROM framework to treat parametrized unsteady Navier-Stokes equations in domains of variable shape, detailing how to build the RB spaces, treat the pressure ensuring the ROM stability, and enhance computational efficiency by a suitable combination of DEIM and MDEIM to assemble nonlinear and/or parameter-dependent arrays. In Sect. 4 we provide a self-contained description of the solid extension mesh moving technique we employ to address shape deformations and their parametrization, as well as a description of the entire reduction workflow. In Sect. 5 we report our numerical results obtained with the proposed framework. A detailed analysis is first carried out on a simpler Navier-Stokes flow in a parametrized cylinder, showing the interplay between all the reduction stages; then, a problem of applied interest is considered, namely blood flows in carotid bifurcations, showing the capability of the proposed reduction scheme to compute, in an efficient way, accurate outputs of clinical interest related with the wall shear stress exerted by blood on the arterial wall.

2 Parametrized Navier-Stokes equations

In this section we introduce the Navier-Stokes (NS) equations for a viscous Newtonian incompressible fluid. Throughout the paper, $\boldsymbol{\mu} \in \mathcal{D}$ denotes a parameter vector, whose components might represent physical and/or geometrical features of interest; $\mathcal{D} \subset \mathbb{R}^p$ denotes the corresponding parameter space. Given an open bounded and $\boldsymbol{\mu}$-dependent domain $\Omega(\boldsymbol{\mu}) \subset \mathbb{R}^d$, $d = 2, 3$, such that $\partial\Omega(\boldsymbol{\mu}) = \Gamma_{out}(\boldsymbol{\mu}) \cup \Gamma_{in}(\boldsymbol{\mu}) \cup \Gamma_w(\boldsymbol{\mu})$ and $\mathring{\Gamma}_{out}(\boldsymbol{\mu}) \cap \mathring{\Gamma}_{in}(\boldsymbol{\mu}) = \mathring{\Gamma}_w(\boldsymbol{\mu}) \cap \mathring{\Gamma}_{in}(\boldsymbol{\mu}) = \mathring{\Gamma}_{out}(\boldsymbol{\mu}) \cap \mathring{\Gamma}_w(\boldsymbol{\mu}) = \emptyset$, and a final time $T > 0$, unsteady NS equations read as follows:

$$\begin{cases} \dfrac{\partial \vec{u}(\boldsymbol{\mu})}{\partial t} + \vec{u}(\boldsymbol{\mu}) \cdot \nabla \vec{u}(\boldsymbol{\mu}) - \nabla \cdot \boldsymbol{\sigma}\big(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu})\big) + \nabla p(\boldsymbol{\mu}) = \vec{0} & \text{in } \Omega(\boldsymbol{\mu}) \times (0,T) \\ \nabla \cdot \vec{u}(\boldsymbol{\mu}) = 0 & \text{in } \Omega(\boldsymbol{\mu}) \times (0,T) \\ \vec{u}(\boldsymbol{\mu}) = \vec{0} & \text{on } \Gamma_w(\boldsymbol{\mu}) \times (0,T) \\ \vec{u}(\boldsymbol{\mu}) = \vec{g}_{NS}(\boldsymbol{\mu}) & \text{on } \Gamma_{in}(\boldsymbol{\mu}) \times (0,T) \\ \boldsymbol{\sigma}\big(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu})\big)\vec{n}(\boldsymbol{\mu}) = \vec{0} & \text{on } \Gamma_{out}(\boldsymbol{\mu}) \times (0,T) \\ \vec{u}(\boldsymbol{\mu}) = \vec{u}_0 & \text{in } \Omega(\boldsymbol{\mu}) \times \{t = 0\}. \end{cases} \tag{1}$$

Here $\vec{u}(\boldsymbol{\mu})$ and $p(\boldsymbol{\mu})$ are the velocity and the pressure of the fluid and $\boldsymbol{\sigma}\big(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu})\big) = -p(\boldsymbol{\mu})\mathbf{I} + 2\nu\boldsymbol{\varepsilon}\big(\vec{u}(\boldsymbol{\mu})\big)$ denotes the stress tensor. Here $\nu = \nu(\boldsymbol{\mu})$ denotes the (possibly parameter-dependent) kinematic viscosity, while $\boldsymbol{\varepsilon}\big(\vec{u}(\boldsymbol{\mu})\big) = \frac{1}{2}\left(\nabla\vec{u}(\boldsymbol{\mu}) + \nabla\vec{u}(\boldsymbol{\mu})^T\right)$ is the strain tensor. Moreover, we assume that time dependence of Dirichlet boundary data can be expressed by separating time and $\boldsymbol{\mu}$, that is,

$$\vec{g}_{NS}(\boldsymbol{\mu}) = \vec{g}_{NS}(t; \boldsymbol{\mu}) = w(t) g_D(\boldsymbol{\mu}). \tag{2}$$

We define the Reynolds number $Re = L\bar{U}/\nu$ as the non-dimensional ratio of convection to diffusion, where L and $\bar{U}$ are the characteristic length of the domain and velocity of the flow, respectively; here we deal with laminar flows, featuring $Re \in [1, 10^3]$.

In order to introduce the variational formulation, let us denote by

$$V_D = \{\vec{v} \in H^1(\Omega(\boldsymbol{\mu}))^d : \vec{v}\big|_{\Gamma_{in}(\boldsymbol{\mu})} = \vec{g}_{NS}(\boldsymbol{\mu}),\, \vec{v}\big|_{\Gamma_w(\boldsymbol{\mu})} = \vec{0}\}, \qquad Q = L^2(\Omega(\boldsymbol{\mu})), \tag{3}$$

the functional spaces for velocity and pressure, respectively, and

$$V = \{\vec{v} \in H^1(\Omega(\boldsymbol{\mu}))^d : \vec{v}\big|_{\Gamma_{in}(\boldsymbol{\mu}) \cup \Gamma_w(\boldsymbol{\mu})} = \vec{0}\}.$$

We highlight that the functional spaces depend on the parameter $\boldsymbol{\mu}$, that is $V = V(\boldsymbol{\mu})$ and $Q = Q(\boldsymbol{\mu})$, and similarly the FE spaces which will be introduced below; the $\boldsymbol{\mu}$-dependence will be however omitted for the sake of clarity. The variational formulation of the parametrized unsteady NS equations reads: for any $t \in (0,T)$, find $(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu})) \in V_D \times Q$ such that

$$\begin{aligned} \left(\frac{\partial \vec{u}(\boldsymbol{\mu})}{\partial t}, \vec{v}\right) + d(\vec{u}(\boldsymbol{\mu}), \vec{v}; \boldsymbol{\mu}) + b(\vec{v}, p(\boldsymbol{\mu}); \boldsymbol{\mu}) + c(\vec{u}(\boldsymbol{\mu}), \vec{u}(\boldsymbol{\mu}), \vec{v}; \boldsymbol{\mu}) \\ + b(q, \vec{u}(\boldsymbol{\mu}); \boldsymbol{\mu}) = 0 \quad \forall (\vec{v}, q) \in V \times Q \end{aligned} \tag{4}$$

with $\vec{u}(\boldsymbol{\mu}) = \vec{u}_0$ as initial condition at $t = 0$ and, for any $\vec{u}, \vec{v}, \vec{w} \in H^1(\Omega(\boldsymbol{\mu}))^d$ and $q \in Q$,

$$d(\vec{u}, \vec{v}; \boldsymbol{\mu}) = \int_{\Omega(\boldsymbol{\mu})} \nu(\boldsymbol{\mu})(\nabla\vec{u} + \nabla\vec{u}^T) : \nabla\vec{v}\, d\Omega(\boldsymbol{\mu}), \qquad b(q, \vec{v}; \boldsymbol{\mu}) = -\int_{\Omega(\boldsymbol{\mu})} q \nabla \cdot \vec{v}\; d\Omega(\boldsymbol{\mu})$$

$$c(\vec{u}, \vec{v}, \vec{w}; \boldsymbol{\mu}) = \int_{\Omega(\boldsymbol{\mu})} (\vec{v} \cdot \nabla)\vec{u} \cdot \vec{w}\; d\Omega(\boldsymbol{\mu}).$$

2.1 FE discretization and BDF time integration

Problem (4) is first discretized in space by means of the FE method, and in time with a backward differentiation formula (BDF) scheme. Given two finite dimensional spaces $V_h \subset V$, $Q_h \subset Q$ with dimensions N_h^u, N_h^p, respectively, such that $N_h^u + N_h^p = N_h$, the semi-discretized problem reads: given $\boldsymbol{\mu} \in \mathcal{D}$, for any $t \in (0, T)$, find $(\vec{u}_h(\boldsymbol{\mu}), p_h(\boldsymbol{\mu})) \in V_h \times Q_h$ such that

$$\begin{aligned}\left(\frac{\partial \vec{u}_h(\boldsymbol{\mu})}{\partial t}, \vec{v}_h\right) + d(\vec{u}_h(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu}) + b(\vec{v}_h, p_h(\boldsymbol{\mu}); \boldsymbol{\mu}) + c(\vec{u}_h(\boldsymbol{\mu}), \vec{u}_h(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu}) \\ + b(q, \vec{u}_h(\boldsymbol{\mu}); \boldsymbol{\mu}) = F_1(t, \vec{v}_h; \boldsymbol{\mu}) + F_2(t, q_h; \boldsymbol{\mu}) \quad \forall(\vec{v}_h, q_h) \in V_h \times Q_h.\end{aligned} \tag{5}$$

Here $F_1(t, \vec{v}; \boldsymbol{\mu})$, $F_2(t, q; \boldsymbol{\mu})$ are linear $(t, \boldsymbol{\mu})$-dependent forms encoding the action of the non homogeneous DIrichlet condition $\vec{u}_h(\boldsymbol{\mu})\big|_{\Gamma_{in}(\boldsymbol{\mu})} = \vec{g}_{NS}(\boldsymbol{\mu})$. Indeed, a lifting approach is used to deal with non homogeneous Dirichlet boundary conditions, as detailed in Sect. 4.

A fully-discretized problem is finally obtained from (5) by using the BDF scheme of order σ. To this aim, let us introduce a partition of the interval $[0, T]$ in N_t subintervals of equal size $\Delta t = T/N_t$, such that $t_n = n\Delta t$, and approximate the time derivative as

$$\frac{d\vec{u}_h(\boldsymbol{\mu})}{dt} \approx \frac{\alpha_1 \vec{u}_h^{n+1}(\boldsymbol{\mu}) - \vec{u}_h^{n,\sigma}(\boldsymbol{\mu})}{\Delta t}. \tag{6}$$

In the numerical examples presented, we will limit ourselves to the case $\sigma = \{1, 2\}$, for which

$$\vec{u}_h^{n,\sigma}(\boldsymbol{\mu}) = \begin{cases} \vec{u}_h^n(\boldsymbol{\mu}), & n \geq 0 \text{ and } \sigma = 1 \\ 2\vec{u}_h^n(\boldsymbol{\mu}) - \dfrac{1}{2}\vec{u}_h^{n-1}(\boldsymbol{\mu}), & n \geq 1 \text{ and } \sigma = 2 \end{cases} \tag{7}$$

and $\alpha_1 = 1,\ 3/2$ for $\sigma = 1, 2$, respectively. Here $(\vec{u}_h^n(\boldsymbol{\mu}), p_h^n(\boldsymbol{\mu}))$ denotes the FE solution at time n. The fully-discretized problem reads: given $\boldsymbol{\mu} \in \mathcal{D}$, $\vec{u}_h^n(\boldsymbol{\mu}), \ldots, \vec{u}_h^{n+1-\sigma}(\boldsymbol{\mu})$, for $n \geq \sigma - 1$ find $(\vec{u}_h^{n+1}(\boldsymbol{\mu}), p_h^{n+1}(\boldsymbol{\mu})) \in V_h \times Q_h$ such that $\vec{u}_h^0(\boldsymbol{\mu}) = \vec{u}_0$ and

$$\begin{aligned}\left(\frac{\alpha_1 \vec{u}_h^{n+1}(\boldsymbol{\mu}) - \vec{u}_h^{n,\sigma}(\boldsymbol{\mu})}{\Delta t}, \vec{v}_h\right) + d(\vec{u}_h^{n+1}(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu}) + b(\vec{v}_h, p_h^{n+1}(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ + c(\vec{u}_h^{n+1}(\boldsymbol{\mu}), \vec{u}_h^{n+1}(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu}) + b(q, \vec{u}_h^{n+1}(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ = F_1(t, \vec{v}_h; \boldsymbol{\mu}) + F_2(t, q_h; \boldsymbol{\mu}) \quad \forall(\vec{v}_h, q_h) \in V_h \times Q_h,\end{aligned} \tag{8}$$

Following this strategy, the fully discrete formulation of problem (8) would consist in a nonlinear problem to be solved at each time-step, e.g. with a Newton method. While a fully implicit approach yields in general a stable time discretization scheme, the associated computational costs may be remarkably high due to the repeated assembly of the residual vector and Jacobian matrix, and the solution of the resulting linear system. To reduce the cost entailed by the use of a fully implicit BDF approach, we consider instead a semi-implicit BDF scheme, where the nonlinear term $c(\vec{u}_h^{n+1}(\boldsymbol{\mu}), \vec{u}_h^{n+1}(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu})$ is extrapolated by means of the Newton-Gregory backward polynomials (see, e.g., [22, 21]). To this aim, we consider the following extrapolations of order $\sigma = 1, 2$ for the velocity at the discrete time t_{n+1}:

$$\vec{u}_h^{n,*}(\boldsymbol{\mu}) = \begin{cases} \vec{u}_h^n(\boldsymbol{\mu}) & \text{if } \sigma = 1 \\ 2\vec{u}_h^n(\boldsymbol{\mu}) - \vec{u}_h^{n-1}(\boldsymbol{\mu}) & \text{if } \sigma = 2, \end{cases}$$

and replace it into the nonlinear term, obtaining the fully discrete linearized semi-implicit formulation of problem (8): given $\boldsymbol{\mu} \in \mathcal{D}$, $\vec{u}_h^n(\boldsymbol{\mu}), \dots, \vec{u}_h^{n+1-\sigma}(\boldsymbol{\mu})$, for $n \geq \sigma - 1$ find $(\vec{u}_h^{n+1}, p_h^{n+1}) \in V_h \times Q_h$ such that $\vec{u}_h^0(\boldsymbol{\mu}) = \vec{u}_0$ and

$$\begin{aligned}\left(\frac{\alpha_1 \vec{u}_h^{n+1}(\boldsymbol{\mu}) - \vec{u}_h^{n,\sigma}(\boldsymbol{\mu})}{\Delta t}, \vec{v}_h\right) &+ d(\vec{u}_h^{n+1}(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu}) + b(\vec{v}_h, p_h^{n+1}(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ &+ c(\vec{u}_h^{n+1}(\boldsymbol{\mu}), \vec{u}_h^{n,*}(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu}) + b(q, \vec{u}_h^{n+1}(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ &\quad = F_1(t, \vec{v}_h; \boldsymbol{\mu}) + F_2(t, q_h; \boldsymbol{\mu}) \quad \forall (\vec{v}_h, q_h) \in V_h \times Q_h.\end{aligned} \tag{9}$$

The fully discrete semi-implicit formulation (9) yields a linear problem in the variables $\vec{u}_h^{n+1}(\boldsymbol{\mu})$ and $p_h^{n+1}(\boldsymbol{\mu})$ to be solved only once at each time t^n, $n = 1, \dots, N_t$. Our high-fidelity, full order model (FOM) will consist of problem (9), where a FE approximation in space and a BDF2 scheme in time are employed.

2.2 Algebraic formulation

Problem (9) leads to a sequence in time of parametrized linear systems of the form

$$\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) \begin{bmatrix} \mathbf{u}^{n+1}(\boldsymbol{\mu}) \\ \mathbf{p}^{n+1}(\boldsymbol{\mu}) \end{bmatrix} = \mathbf{g}^{n+1}(\boldsymbol{\mu}) \qquad n = 0, \dots, N_t - 1, \tag{10}$$

where $\mathbf{u}^n(\boldsymbol{\mu}), \mathbf{u}^{n,*}(\boldsymbol{\mu}), \mathbf{u}^{n,\sigma}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^u}$ and $\mathbf{p}^n(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^p}$ denote the FE vector representation of the FE functions $\vec{u}_h^n(\boldsymbol{\mu}), \vec{u}_h^{n,*}(\boldsymbol{\mu}), \vec{u}_h^{n,\sigma}(\boldsymbol{\mu})$ and $p_h^n(\boldsymbol{\mu})$, respectively, and $\mathbf{u}^0(\boldsymbol{\mu}) = \mathbf{u}_0 \in \mathbb{R}^{N_h^u}$ is the initial condition. $\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ and $\mathbf{g}^{n+1}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ are given by

$$\begin{aligned}\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) &= \begin{bmatrix} \frac{\alpha_1}{\Delta t}\mathbf{M}^u(\boldsymbol{\mu}) + \mathbf{D}(\boldsymbol{\mu}) + \mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) & \mathbf{B}^T(\boldsymbol{\mu}) \\ \mathbf{B}(\boldsymbol{\mu}) & \mathbf{0} \end{bmatrix}, \\ \mathbf{g}^{n+1}(\boldsymbol{\mu}) &= \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}^u(\boldsymbol{\mu})\mathbf{u}^{n,\sigma}(\boldsymbol{\mu}) + \mathbf{f}_1^{n+1}(\boldsymbol{\mu}) \\ \mathbf{f}_2^{n+1}(\boldsymbol{\mu}) \end{bmatrix}.\end{aligned} \tag{11}$$

Here $\mathbf{M}^u(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^u \times N_h^u}$ is the velocity mass matrix, that is

$$\big(\mathbf{M}^u(\boldsymbol{\mu})\big)_{ij} = (\phi_j^u, \phi_i^u)_{L^2(\Omega(\boldsymbol{\mu}))} \qquad i, j = 1, \dots, N_h^u,$$

$\mathbf{D}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^u \times N_h^u}$ and $\mathbf{B}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^p \times N_h^u}$ are the velocity stiffness and the divergence operator, respectively, defined as

$$\begin{aligned}\big(\mathbf{D}(\boldsymbol{\mu})\big)_{ij} &= d(\phi_j^{\vec{u}}(\boldsymbol{\mu}), \phi_i^{\vec{u}}(\boldsymbol{\mu}); \boldsymbol{\mu}) \qquad \forall i, j = 1, \dots, N_h^u \\ \big(\mathbf{B}(\boldsymbol{\mu})\big)_{ij} &= b(\phi_j^{\vec{u}}(\boldsymbol{\mu}), \phi_i^p(\boldsymbol{\mu}); \boldsymbol{\mu}) \qquad \forall i = 1, \dots, N_h^p,\ j = 1, \dots, N_h^u.\end{aligned} \tag{12}$$

The matrix $\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) \in \mathbb{R}^{N_h^u \times N_h^u}$ arises from the linearization of the nonlinear term,

$$\big(\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})\big)_{ij} = c(\phi_j^u, \vec{u}_h^{n,*}(\boldsymbol{\mu}), \phi_i^u; \boldsymbol{\mu}) \qquad i, j = 1, \dots, N_h^u. \tag{13}$$

Remark 2.1. *To guarantee the well-posedness of the algebraic problem* (10)*, the velocity and pressure FE spaces V_h and Q_h must yield a divergence matrix $\mathbf{B}(\boldsymbol{\mu})$ that fulfills the following inf-sup condition: there exists $\beta_p > 0$ such that*

$$\beta_{hp}^{\boldsymbol{\mu}} = \inf_{\mathbf{q} \in \mathbb{R}^{N_h^p}} \sup_{\mathbf{v} \in \mathbb{R}^{N_h^u}} \frac{\mathbf{v}^T \mathbf{B}(\boldsymbol{\mu}) \mathbf{q}}{\|\mathbf{v}\|_{\mathbf{X}_u(\boldsymbol{\mu})} \|\mathbf{q}\|_{\mathbf{X}_p(\boldsymbol{\mu})}} \geq \beta_p \qquad \forall \boldsymbol{\mu} \in \mathcal{D}; \tag{14}$$

A possible choice, which is the one used in the numerical experiments, consists in employing Taylor-Hood FE spaces, that is $\mathbb{P}^2$ and $\mathbb{P}^1$ basis functions for velocity and pressure, respectively.

The efficient solution of the sequence of linear systems defined in (10) calls into play suitable numerical techniques, among which we mention, e.g., multilevel and domain decomposition methods [20, 23, 54, 59] and block-preconditioning strategies [19, 14, 57], such as the least-squares commutator [17, 37] and the pressure-convection-diffusion [28, 48] preconditioners. A relevant preconditioner strategy employed for the solution of the NS equations is the SIMPLE preconditioner [46, 56], which is based on the following factorization

$$\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) = \mathcal{L}_{bt}(\boldsymbol{\mu})\mathcal{U}(\boldsymbol{\mu}) \tag{15}$$

with

$$\mathcal{L}_{bt}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{F}(\boldsymbol{\mu}) & 0 \\ \mathbf{B}(\boldsymbol{\mu}) & \mathbf{S}(\boldsymbol{\mu}) \end{bmatrix}, \qquad \mathcal{U}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{I}_{N_h^u} & \mathbf{F}^{-1}(\boldsymbol{\mu})\mathbf{B}^T(\boldsymbol{\mu}) \\ 0 & \mathbf{I}_{N_h^p} \end{bmatrix}. \tag{16}$$

Here $\mathbf{S}(\boldsymbol{\mu}) = -\mathbf{B}(\boldsymbol{\mu})\mathbf{F}^{-1}(\boldsymbol{\mu})\mathbf{B}^T(\boldsymbol{\mu})$ denotes the Schur complement matrix and $\mathbf{F}(\boldsymbol{\mu})$ is the (1,1)-block of the matrix $\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$ defined in (11). The application of the SIMPLE preconditioner, which we denote by $\mathbf{P}_{\text{SIMPLE}}(\boldsymbol{\mu})$, to the Krylov basis is carried out according to Algorithm 1 (which is equivalent to a single iteration of the classical SIMPLE method), where $\mathbf{F}(\boldsymbol{\mu})$ is substituted by its diagonal in the definition of $\mathbf{S}(\boldsymbol{\mu})$ and when performing the update step. Moreover, in our implementation, steps 1-2 are replaced by inner GMRES iterations. In our numerical experiments, the linearized linear system (10) will be solved with the flexible GMRES (FGMRES) [45] method, employing a SIMPLE preconditioner.

3 A ROM framework for parametrized unsteady NS equations in domains of variable shape

In this section we present a ROM technique to reduce the cost needed to solve the FE system (10), by providing an algebraic, black-box, way to treat the NS equations parametrized geometry and the nonaffine parametric dependence entailed by the nonlinear term. In the numerical examples used to illustrate the method, we will put more emphasis on geometrical parameters, however the construction is general and accounts for physical parameters as well. For ease of notation, we will carry out the whole construction form an algebraic standpoint.

The RB approximation of velocity and pressure fields at time t_n is expressed as a linear combination of the RB basis functions,

$$\mathbf{u}^n(\boldsymbol{\mu}) \approx \mathbf{V}_u\mathbf{u}_N^n(\boldsymbol{\mu}), \qquad \mathbf{p}^n(\boldsymbol{\mu}) \approx \mathbf{V}_p\mathbf{p}_N^n(\boldsymbol{\mu}) \tag{17}$$

where $\mathbf{V}_u \in \mathbb{R}^{N_h^u \times N_u}$ and $\mathbf{V}_p \in \mathbb{R}^{N_h^p \times N_p}$ denote the matrices whose columns are the vectors of degrees of freedom of the basis functions for the velocity and the pressure RB spaces, respectively. The construction of these spaces will be detailed in the following section.

Algorithm 1 Computation of $\mathbf{z} = \mathbf{P}_{\text{SIMPLE}}^{-1}\mathbf{v}$ ($\boldsymbol{\mu}$ is omitted)

```
1: solve $\mathbf{F}\mathbf{z}_u = \mathbf{v}_u$
2: solve $\mathbf{S}\mathbf{z}_p = \mathbf{v}_p - \mathbf{B}\mathbf{z}_u$
3: update $\mathbf{z}_u = \mathbf{z}_u - \mathbf{F}^{-1}\mathbf{B}^T\mathbf{z}_p$
```

Substituting (17) into (10) and performing a Galerkin projection, we obtain the following Galerkin-RB problem: given $\boldsymbol{\mu} \in \mathcal{D}$, $\mathbf{u}_N^n, \dots, \mathbf{u}_N^{n+1-\sigma}$, for $n \geq \sigma - 1$ find $(\mathbf{u}_N^{n+1}(\boldsymbol{\mu}), \mathbf{p}_N^{n+1}(\boldsymbol{\mu})) \in \mathbb{R}^{N_u} \times \mathbb{R}^{N_p}$ such that $\mathbf{u}_N^0(\boldsymbol{\mu}) = \mathbf{u}_{N,0}$ and

$$\mathbf{N}_N(\mathbf{V}_u \mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) \begin{bmatrix} \mathbf{u}_N^{n+1}(\boldsymbol{\mu}) \\ \mathbf{p}_N^{n+1}(\boldsymbol{\mu}) \end{bmatrix} = \mathbf{g}_N^{n+1}(\boldsymbol{\mu}). \tag{18}$$

The RB arrays $\mathbf{N}_N(\mathbf{V}_u \mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) \in \mathbb{R}^{N \times N}$ and $\mathbf{g}_N^{n+1}(\boldsymbol{\mu}) \in \mathbb{R}^N$ are obtained by projecting onto the RB spaces $\mathbf{V}_u$ and $\mathbf{V}_p$ the corresponding blocks defined in (11); in other words, they can be obtained as

$$\begin{aligned} \mathbf{N}_N(\mathbf{V}_u \mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) &= \begin{bmatrix} \frac{\alpha_1}{\Delta t} \mathbf{M}_N^u(\boldsymbol{\mu}) + \mathbf{D}_N(\boldsymbol{\mu}) + \mathbf{C}_N(\mathbf{V}_u \mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) & \mathbf{B}_N^T(\boldsymbol{\mu}) \\ \mathbf{B}_N(\boldsymbol{\mu}) & 0 \end{bmatrix}, \\ \mathbf{g}_N^{n+1}(\boldsymbol{\mu}) &= \begin{bmatrix} \frac{1}{\Delta t} \mathbf{M}_N^u(\boldsymbol{\mu}) \mathbf{u}_N^{n,\sigma}(\boldsymbol{\mu}) + \mathbf{f}_{N1}^{n+1}(\boldsymbol{\mu}) \\ \mathbf{f}_{N2}^{n+1}(\boldsymbol{\mu}) \end{bmatrix}, \end{aligned} \tag{19}$$

where

$$\begin{aligned} \mathbf{D}_N(\boldsymbol{\mu}) = \mathbf{V}_u^T \mathbf{D}(\boldsymbol{\mu}) \mathbf{V}_u, \qquad \mathbf{M}_N^u(\boldsymbol{\mu}) = \mathbf{V}_u^T \mathbf{M}^u(\boldsymbol{\mu}) \mathbf{V}_u, \qquad \mathbf{B}_N(\boldsymbol{\mu}) = \mathbf{V}_p^T \mathbf{B}(\boldsymbol{\mu}) \mathbf{V}_u, \\ \mathbf{f}_{N1}^{n+1}(\boldsymbol{\mu}) = \mathbf{V}_u^T \mathbf{f}_1^{n+1}(\boldsymbol{\mu}), \qquad \mathbf{f}_{N2}^{n+1}(\boldsymbol{\mu}) = \mathbf{V}_p^T \mathbf{f}_2^{n+1}(\boldsymbol{\mu}), \qquad \mathbf{u}_{N,0} = \mathbf{V}_u^T \mathbf{u}_0. \end{aligned} \tag{20}$$

Finally, the linearized term $\mathbf{C}_N(\mathbf{V}_u \mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$ is obtained by projecting its FE element counterpart evaluated at the RB approximation, that is

$$\mathbf{C}_N(\mathbf{V}_u \mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) = \mathbf{V}_u^T \mathbf{C}(\mathbf{V}_u \mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) \mathbf{V}_u. \tag{21}$$

Remark 3.1. *An alternative to the Galerkin-RB formulation would consist in a Petrov-Galerkin method. Such option is particularly convenient when dealing with turbulent flows, since it allows to obtain a properly well-posed ROM in terms of long-term stability for highly nonlinear dynamical systems, see e.g. [10]; however, for the regimes we are interested in, there is not such an issue, and a Galerkin approach represents a reliable, and cheaper, option.*

Remark 3.2. *Throughout this chapter, we will use the matrices $\mathbf{X}_u \in \mathbb{R}^{N_h^u \times N_h^u}$ and $\mathbf{X}_p \in \mathbb{R}^{N_h^p \times N_h^p}$, which algebraically encode the scalar products $(\cdot,\cdot)_{V_h}$ and $(\cdot,\cdot)_{Q_h}$ over the velocity and pressure space, respectively, and define*

$$\mathbf{X}_h(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{X}_u(\boldsymbol{\mu}) & 0 \\ 0 & \mathbf{X}_p(\boldsymbol{\mu}) \end{bmatrix}.$$

3.1 Basis construction: double POD strategy

To construct the reduced basis matrices $\mathbf{V}_u$ and $\mathbf{V}_p$ we rely on POD. This requires to collect snapshots of the FOM solution for a sample of selected parameter values $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$ by computing, for $n = 0, \dots, N_t - 1$, the solution of the high-fidelity linear system (10); then, we perform POD separately on velocity and pressure snapshots. This procedure would in principle lead to either an SVD of very large snapshot matrices (of size $n_s N_t \times N_h^u$ and $n_s N_t \times N_h^p$, respectively), or an eigenproblem for two correlation matrices of size $n_s N_t \times n_s N_t$. To make the paper self-contained, we report a brief explanation of POD in the Appendix, Sect. A.1.

When the FOM dimension N_h is sufficiently large, both these options would entail an overwhelming computational burden. To avoid such a cost, given the parameter values $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$, we rather build the POD basis sequentially by performing at first *(i)* POD with respect to the time trajectory (for a fixed $\boldsymbol{\mu}$) and, after collecting this information, *(ii)* POD with respect to the parametric dependence. This procedure is carried out in the following steps:

1. *POD in time.* For each $\boldsymbol{\mu}_i$, $i = 1, \dots, n_s$, we compute the solution of (10) for $n = 0, \dots, N_t - 1$ and collect snapshots $[\mathbf{u}^1(\boldsymbol{\mu}_i), \dots, \mathbf{u}^{N_t}(\boldsymbol{\mu}_i)]$ (resp. $[\mathbf{p}^1(\boldsymbol{\mu}_i), \dots, \mathbf{p}^{N_t}(\boldsymbol{\mu}_i)]$) in a matrix $\mathbf{S}^i_{\vec{u}}$ (resp. $\mathbf{S}^i_p$). Then, we perform SVD on the time trajectory, that is, for any $i = 1, \dots, n_s$, we compute
$$\mathbf{S}^i_{\vec{u}} = [\mathbf{u}^1(\boldsymbol{\mu}_i), \dots, \mathbf{u}^{N_t}(\boldsymbol{\mu}_i)], \qquad \mathbf{V}^i_u = \mathrm{POD}(\mathbf{S}^i_{\vec{u}}, \mathbf{X}_u, \varepsilon_t),$$
$$\mathbf{S}^i_p = [\mathbf{p}^1(\boldsymbol{\mu}_i), \dots, \mathbf{p}^{N_t}(\boldsymbol{\mu}_i)], \qquad \mathbf{V}^i_p = \mathrm{POD}(\mathbf{S}^i_p, \mathbf{X}_p, \varepsilon_t);$$

2. *POD in parameter.* After collecting all the basis functions obtained by the n_s POD in time, we perform a final POD of the matrix collecting the retained basis functions,
$$\mathbf{S}_{\vec{u}} = [\mathbf{V}^1_u, \dots, \mathbf{V}^{n_s}_u], \qquad \mathbf{V}_u = \mathrm{POD}(\mathbf{S}_{\vec{u}}, \mathbf{X}_u, \varepsilon_\mu),$$
$$\mathbf{S}_p = [\mathbf{V}^1_p, \dots, \mathbf{V}^{n_s}_p], \qquad \mathbf{V}_p = \mathrm{POD}(\mathbf{S}_p, \mathbf{X}_p, \varepsilon_\mu).$$

The tolerances $\varepsilon_t \geq \varepsilon_\mu > 0$ are used to set suitable stopping criteria, based as usual on the (sum of) discarded singular values for POD in time and parameter, respectively; requiring that $\varepsilon_t \geq \varepsilon_\mu$ ensures that POD in parameter is based on a proper sampling in time.

3.2 ROM Stability

Performing, as in (18), a Galerkin projection onto the RB space built through the POD procedure above, unfortunately, does not automatically ensure the stability of the resulting RB problem (in the sense of the fulfillment of an *inf-sup* condition at the reduced level), thus yielding a potentially singular matrix $\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$. Several strategies can be employed to overcome this issue; here we augment the velocity space by means of a set of *enriching* basis functions computed through the pressure supremizing operator, which depends on the divergence term. This yields a RB problem with additional degrees of freedom for the velocity field (as many as the pressure variable), see [44, 5] for the details.

To this aim, we introduce the pressure supremizing operator, such that, for any given $q_h \in Q_h$, $T_p(q_h; \boldsymbol{\mu})$ returns the solution of the following variational problem
$$(T_p(q_h; \boldsymbol{\mu}), \vec{v}_h)_{V(\boldsymbol{\mu})} = b(\vec{v}_h, q_h; \boldsymbol{\mu}) \qquad \forall \vec{v}_h \in V_h. \tag{22}$$
We essentially extend the procedure explored in [5] to the time-dependent case, so that in practice the enriching velocity functions are constructed as follows:

1. for each $i = 1, \dots, n_s$ and for each $n = 1, \dots, N_t$ we compute the supremizers, by solving
$$\mathbf{X}_u(\boldsymbol{\mu}_i)\mathbf{t}^n_p(\boldsymbol{\mu}_i) = \mathbf{B}^T(\boldsymbol{\mu}_i)\mathbf{p}^n(\boldsymbol{\mu}_i), \tag{23}$$
we collect the supremizer snapshots in the matrix $\mathbf{S}^i_{\vec{t}} \in \mathbb{R}^{N^u_h \times n_s}$ and compress them by performing POD in time
$$\mathbf{S}^i_{\vec{t}} = [\mathbf{t}^1_p(\boldsymbol{\mu}_i), \dots, \mathbf{t}^{N_t}_p(\boldsymbol{\mu}_i)] \qquad \mathbf{V}^i_s = \mathrm{POD}(\mathbf{S}^i_{\vec{t}}, \mathbf{X}_u, \varepsilon_t);$$

2. we generate a global snapshot matrix and perform a POD in parameter to obtain an enriching basis $\mathbf{V}_s \in \mathbb{R}^{N^u_h \times N_s}$
$$\mathbf{S}_{\vec{t}} = [\mathbf{V}^1_s, \dots, \mathbf{V}^{n_s}_s] \qquad \mathbf{V}_s = \mathrm{POD}(\mathbf{S}_{\vec{t}}, \mathbf{X}_u, \varepsilon_\mu);$$

3. we finally perform a Gram-Schmidt orthonormalization to merge the supremizer basis functions with the columns of $\mathbf{V}_u$ and obtain the basis matrix for the velocity space,

$$\mathbf{V}_u = \text{G-S}([\mathbf{V}_u, \mathbf{V}_s], \mathbf{X}_u).$$

In particular, an *approximate supremizer* option is pursued, in order the velocity space not to be $\boldsymbol{\mu}$-dependent. Note that, for the case at hand, the supremizers must take into account also time dependence. In presence of $\boldsymbol{\mu}$-dependent domains, the supremizing operator is $\boldsymbol{\mu}$-dependent, too: in order to avoid the construction of the pressure supremizing operator online, for each $\boldsymbol{\mu}$, an offline enrichment is employed. This strategy leads to a RB problem which is *inf-sup* stable in practice, but whose well-posedness is not rigorously proven [44, 5]. Alternative strategies to ensure ROM stability would rely either on a Petrov-Galerkin (e.g., least squares, (LS)) RB method, or on the use of a stabilized FOM (like, e.g., a $\mathbb{P}^1 - \mathbb{P}^1$ Streamline Upwind Petrov-Galerkin (SUPG) finite element method).

In the former case, the resulting LS-RB method uses a test space which is obtained as the image of the trial RB space through a global supremizing operator involving both velocity and pressure fields, yielding an automatically stable RB problem. An in-depth analysis and possible ways to ensure the computational efficiency of such a strategy have been presented by the authors in [13]; further investigations regarding the Navier-Stokes case are still in progress.

In the latter case, the resulting Galerkin-RB approximation is well-posed; neither an enrichment of the velocity space, nor a LS-RB formulation, are necessary to ensure the stability of the corresponding RB system; see, e.g., [1, 38].

Galerkin projection would still provide a stable RB problem in the case pressure is treated independently from velocity, and is reconstructed by solving a Poisson equation [8]. This strategy, however, requires divergence-free velocity basis functions; for the case at hand, which involves non closed fluid configurations, geometrical deformations and a lifting approach for handling non homogeneous essential boundary conditions, this assumption is not met.

3.3 Enhancing efficiency by hyper-reduction

Because of the $\boldsymbol{\mu}$-dependence induced by the geometry deformation, all the matrices and vectors in the ROM (10) depend nonaffinely on the parameter $\boldsymbol{\mu}$; moreover, a critical issue is represented by the linearized term $\mathbf{C}_N(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$ appearing in (11). In order to assemble it, we should at first build $\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu})$, required to assemble the FOM matrix $\mathbf{C}(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$; this latter must then be projected as in (21). Unfortunately, these operations would make the assembling of ROM arrays very expensive, thus preventing an efficient offline/online decoupling. We highlight that such a difficulty arises because of the nonaffine parameter dependence entailed by the geometrical deformation; if, instead, an affine parametrization is considered, the quadratically nonlinear term $\mathbf{C}_N(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$ could be expressed as an sum of N_u affine components, where N_u is the dimension of the RB velocity space [33].

Hyper-reduction techniques aim at enhancing the efficiency of a ROM during the online stage, with the goal of pursuing an assembling phase independent of the very high-dimensional FE arrays; the resulting hyper-reduced order model (HROM) relies on a set of $\boldsymbol{\mu}$-independent quantities which can be stored and then combined for each new instance of the parameters to be queried online. With this aim, we employ here, for the first time in the case of nonlinear unsteady NS equations in parametrized geometries, the matrix version of the discrete empirical interpolation method (DEIM). Such a procedure requires the evaluation of a sample of system (vectors and matrices) snapshots, followed by a POD on vectors and vectorized matrices, then by a further selection of a set of well-chosen interpolation points.

To start with, MDEIM can be readily employed to compute an approximated affine decomposition $\{\mathbf{D}^q\}_{q=1}^{Q_d}$ of the diffusion matrix $\mathbf{D}(\boldsymbol{\mu})$, $\{\mathbf{B}^q\}_{q=1}^{Q_b}$ of the pressure-divergence matrix $\mathbf{B}(\boldsymbol{\mu})$, and $\{(\mathbf{M}^u)^q\}_{q=1}^{Q_b}$ of the velocity mass $\mathbf{M}^u(\boldsymbol{\mu})$ matrix, respectively. The corresponding ROM arrays $\mathbf{D}_N(\boldsymbol{\mu})$, $\mathbf{B}_N(\boldsymbol{\mu})$ and $\mathbf{M}_N^u(\boldsymbol{\mu})$ can then be approximated by

$$\widetilde{\mathbf{D}}_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_d} \theta_d^q(\boldsymbol{\mu})\mathbf{D}_N^q, \qquad \widetilde{\mathbf{B}}_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_b} \theta_b^q(\boldsymbol{\mu})\mathbf{B}_N^q, \qquad \widetilde{\mathbf{M}}_N^u(\boldsymbol{\mu}) = \sum_{q=1}^{Q_m} \theta_m^q(\boldsymbol{\mu})(\mathbf{M}_N^u)^q, \quad (24)$$

respectively, where the matrices $\mathbf{D}_N^q = \mathbf{V}_u^T\mathbf{D}^q\mathbf{V}_u$, $\mathbf{B}_N^q = \mathbf{V}_p^T\mathbf{B}^q\mathbf{V}_u$ and $(\mathbf{M}_N^u)^q = \mathbf{V}_u^T(\mathbf{M}^u)^q\mathbf{V}_u^T$ can be assembled and stored once for all. The $\boldsymbol{\mu}$-dependent coefficients appearing in each of the expansions above are determined by solving an interpolation problem for any new value of $\boldsymbol{\mu}$, as usually when dealing with empirical interpolation, see Sect. A.2 in the Appendix.

The assumption on the inlet condition outlined in (2) allows, on the other hand, to uncouple time and space-parameter contributions in the inlet Dirichlet condition. This is then reflected in the corresponding contribution at the right hand side of (10), which can be expressed as

$$\mathbf{f}_1^{n+1}(\boldsymbol{\mu}) = w(t_{n+1})\mathbf{f}_1(\boldsymbol{\mu}), \qquad \mathbf{f}_2^{n+1}(\boldsymbol{\mu}) = w(t_{n+1})\mathbf{f}_2(\boldsymbol{\mu}), \quad (25)$$

where $\mathbf{f}_i(\boldsymbol{\mu})$, $i = 1, 2$ are time-independent vectors, thus yielding

$$\mathbf{f}_{N1}^{n+1}(\boldsymbol{\mu}) = w(t_{n+1})\mathbf{V}_u^T\mathbf{f}_1(\boldsymbol{\mu}) = w(t_{n+1})\mathbf{f}_{N1}(\boldsymbol{\mu}), \quad \mathbf{f}_{N2}^{n+1}(\boldsymbol{\mu}) = w(t_{n+1})\mathbf{V}_u^T\mathbf{f}_2(\boldsymbol{\mu}) = w(t_{n+1})\mathbf{f}_{N2}(\boldsymbol{\mu}).$$

Thanks to (25), we can use DEIM to build an affine approximation $\{\mathbf{f}_1^q\}_{q=1}^{Q_g^1}$, $\{\mathbf{f}_2^q\}_{q=1}^{Q_g^2}$ of $\mathbf{f}_1(\boldsymbol{\mu})$ and $\mathbf{f}_2(\boldsymbol{\mu})$, respectively., which is then employed to precompute and store in the offline phase the affine approximations $\{\mathbf{f}_{N1}^q\}_{q=1}^{Q_g^1}$, $\{\mathbf{f}_{N2}^q\}_{q=1}^{Q_g^2}$ for $\mathbf{f}_{N1}(\boldsymbol{\mu})$ and $\mathbf{f}_{N2}(\boldsymbol{\mu})$, respectively, such that

$$\mathbf{f}_{N1}(\boldsymbol{\mu}) \approx \widetilde{\mathbf{f}}_{N1}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_g^1} \theta_{N1}(\boldsymbol{\mu})\mathbf{f}_{N1}^q, \qquad \mathbf{f}_{N2}(\boldsymbol{\mu}) \approx \widetilde{\mathbf{f}}_{N2}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_g^2} \theta_{N2}(\boldsymbol{\mu})\mathbf{f}_{N2}^q. \quad (26)$$

The linearized term $\mathbf{C}(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$ nonaffinely depends on the parameter $\boldsymbol{\mu}$, however an MDEIM-approximated affine decomposition is not readily computable, due to its dependence on $\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu})$; hence, a different strategy, which takes advantage of a sequential time-parameter POD approach is used in this respect. In particular, once the sequence of linear systems (10) is solved for the parameter instances $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$, the following steps are executed:

1. *MDEIM in time.* For each parameter $\boldsymbol{\mu}_i$, $i = 1, \dots, n_s$, vectorized matrix snapshots in time of the convective term are collected, and POD is applied to build a basis with respect to the time trajectory of the system

$$\mathbf{S}_C^i = [\mathrm{vec}(\mathbf{C}(\mathbf{u}^{0,*}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i)), \dots, \mathrm{vec}(\mathbf{C}(\mathbf{u}^{N_t-1,*}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i))], \quad \mathbf{V}_C^i = \mathrm{MDEIM}(\mathbf{S}_C^i, \varepsilon_C^{loc});$$

2. all the time matrix basis are gathered and a final approximated affine basis is constructed with respect to the parameter dependence

$$\mathbf{S}_C = [\mathbf{V}_C^1, \dots, \mathbf{V}_C^{n_s}] \qquad \mathbf{V}_C = \mathrm{MDEIM}(\mathbf{S}_C, \varepsilon_C);$$

3. the approximate affine decomposition $\{\mathbf{C}_N^q\}_{q=1}^{Q_c}$ of $\mathbf{C}_N(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$ is finally built, such that

$$\begin{aligned}\mathbf{C}_N(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) &\approx \widetilde{\mathbf{C}}_N(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) \\ &= \sum_{q=1}^{Q_c}\tilde{\Theta}_c^q(\boldsymbol{\mu})\mathbf{V}_u^T\mathbf{C}^q\mathbf{V}_u = \sum_{q=1}^{Q_c}\tilde{\Theta}_c^q(\boldsymbol{\mu})\mathbf{C}_N^q. \qquad (27)\end{aligned}$$

Here the matrices $\mathbf{C}^q \in \mathbb{R}^{N_h^u\times N_h^u}$, $q=1,\dots,Q_c$, are the "unvectorized" columns of $\mathbf{V}_C$ and constitute an approximated affine basis for $\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$. As a matter of fact, the $\boldsymbol{\mu}$-independent matrices $\mathbf{C}_N^q \in \mathbb{R}^{N\times N}$ can be precomputed and stored once for all.

In the procedure above, ε_C^{loc} and ε_C are the tolerances used to stop the modes selection when performing POD in time (for each $k=1,\dots,n_s$) and in parameter, respectively. The final algorithm involving the construction of the NS-HROM for the parametrized sequence of algebraic system (10) is outlined in Algorithm 2.

When considering a new parameter online, we solve the approximated RB system

$$\widetilde{\mathbf{N}}_N(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\begin{bmatrix}\mathbf{u}_N^{n+1}(\boldsymbol{\mu})\\ \mathbf{p}_N^{n+1}(\boldsymbol{\mu})\end{bmatrix} = \widetilde{\mathbf{g}}_N^{n+1}(\boldsymbol{\mu}), \qquad \text{(NS-HROM)}$$

where $\widetilde{\mathbf{N}}_N(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$ features the same saddle-point structure as the matrix in (19), but involves the approximated affine matrices $\widetilde{\mathbf{D}}_N(\boldsymbol{\mu})$, $\widetilde{\mathbf{B}}_N(\boldsymbol{\mu})$, $\widetilde{\mathbf{M}}_N^u(\boldsymbol{\mu})$ and $\widetilde{\mathbf{C}}_N(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$; similarly the DEIM-approximated RB vectors (26) are employed for the cheap assembly of the right hand side $\widetilde{\mathbf{g}}_N^{n+1}(\boldsymbol{\mu})$.

Algorithm 2 Offline construction NS-HROM

```
1: procedure NS-RB_OFFLINE($\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}, \varepsilon_t, \varepsilon_\mu, \varepsilon_C^{loc}, \varepsilon_C, \delta_{\text{mdeim}}, \delta_{\text{deim}}$)
2:     Use MDEIM to compute an affine decomposition of $\mathbf{D}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$, $\mathbf{M}^u(\boldsymbol{\mu})$
3:     Use DEIM to compute an affine decomposition of $\mathbf{f}_1(\boldsymbol{\mu})$, $\mathbf{f}_2(\boldsymbol{\mu})$
4:     for i = 1 : $n_s$ do
5:         Compute $\{\mathbf{u}^n(\boldsymbol{\mu}_i)\}_{n=1}^{N_t}$, $\{\mathbf{p}^n(\boldsymbol{\mu}_i)\}_{n=1}^{N_t}$, $\{\mathbf{t}_p^n(\boldsymbol{\mu}_i)\}_{n=1}^{N_t}$
6:         Set $\mathbf{S}_{\vec{u}}^i = [\mathbf{u}^1(\boldsymbol{\mu}_i),\dots,\mathbf{u}^{N_t}(\boldsymbol{\mu}_i)]$ and $\mathbf{V}_u^i = \text{POD}(\mathbf{S}_{\vec{u}}^i, \mathbf{X}_u, \varepsilon_t)$
7:         Set $\mathbf{S}_p^i = [\mathbf{p}^1(\boldsymbol{\mu}_i),\dots,\mathbf{p}^{N_t}(\boldsymbol{\mu}_i)]$ and $\mathbf{V}_p^i = \text{POD}(\mathbf{S}_p^i, \mathbf{X}_p, \varepsilon_t)$
8:         Set $\mathbf{S}_{\vec{t}}^i = [\mathbf{t}_p^1(\boldsymbol{\mu}_i),\dots,\mathbf{t}_p^{N_t}(\boldsymbol{\mu}_i)]$ and $\mathbf{V}_s^i = \text{POD}(\mathbf{S}_{\vec{t}}^i, \mathbf{X}_u, \varepsilon_t)$
9:         Set $\mathbf{S}_C^i = [\text{vec}(\mathbf{C}(\mathbf{u}^{0,*}(\boldsymbol{\mu}_i);\boldsymbol{\mu}_i)),\dots,\text{vec}(\mathbf{C}(\mathbf{u}^{N_t-1,*}(\boldsymbol{\mu}_i);\boldsymbol{\mu}_i))]$
10:            and $\mathbf{V}_C^i = \text{POD}(\mathbf{S}_C^i, \mathbf{I}_{(N_h^u)^2}, \varepsilon_C^{loc})$
11:    end for
12:    Set $\mathbf{S}_{\vec{u}} = [\mathbf{V}_u^1,\dots,\mathbf{V}_u^{n_s}]$ and $\mathbf{V}_u = \text{POD}(\mathbf{S}_{\vec{u}}, \mathbf{X}_u, \varepsilon_\mu)$
13:    Set $\mathbf{S}_p = [\mathbf{V}_p^1,\dots,\mathbf{V}_p^{n_s}]$ and $\mathbf{V}_u = \text{POD}(\mathbf{S}_p, \mathbf{X}_p, \varepsilon_\mu)$
14:    Set $\mathbf{S}_{\vec{t}} = [\mathbf{V}_s^1,\dots,\mathbf{V}_s^{n_s}]$ and $\mathbf{V}_s = \text{POD}(\mathbf{S}_{\vec{t}}, \mathbf{X}_u, \varepsilon_\mu)$
15:    Set $\mathbf{S}_C = [\mathbf{V}_C^1,\dots,\mathbf{V}_C^{n_s}]$ and $\mathbf{V}_C = \text{MDEIM}(\mathbf{S}_C, \varepsilon_C)$
16:    Orthonormalize: $\mathbf{V}_u = \text{G-S}(\mathbf{V}_u, \mathbf{V}_s, \mathbf{X}_u)$
17:    Precompute and store the (approximated) RB affine decompositions:
18:    $\{\mathbf{D}_N^q\}_{q=1}^{Q_d}$, $\{\mathbf{B}_N^q\}_{q=1}^{Q_b}$, $\{\mathbf{M}_N^q\}_{q=1}^{Q_m}$, $\{\mathbf{C}_N^q\}_{q=1}^{Q_c}$, $\{\mathbf{f}_{N1}^q\}_{q=1}^{Q_g^1}$, $\{\mathbf{f}_{N2}^q\}_{q=1}^{Q_g^2}$.
19: end procedure
```

Remark 3.3. *For the sake of computational efficiency, we compute snapshots of the nonlinear term $\mathbf{S}_C^i$, for each parameter $\boldsymbol{\mu}_i$, $i = 1, \dots, n_s$, by evaluating the entries of the matrix $\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i)$, $n = 1, \dots, N_t$, obtained by linearizing the convective term around the FOM solution $\mathbf{u}(\boldsymbol{\mu})$, rather than around the ROM approximation. Evaluating this latter option would indeed require the construction of a first ROM without taking into account hyper-reduction of nonlinear terms, to be used to generate the ROM approximation required to evaluate the snapshots of the nonlinear term. By avoiding this extra effort, we manage to evaluate all the required snapshots by running just n_s FOM simulations over the time interval (see lines 6-10 of Algorithm 2). Nevertheless, due to the quadratic nonlinearity, the additional error entailed by this approximation does not impact on the overall accuracy of the proposed ROM.*

4 Sequential ROMs for deformation and fluid flows

To deal with complex domains and their deformations in an extremely flexible way, we exploit a general mesh deformation technique, in which deformations result from an additional FE problem either describing the behavior of the structure with respect to given inputs or an harmonic extension of boundary data. Such a technique belongs to the class of so called *mesh-based variational methods* (see, e.g., [50]), which compute smooth harmonic [2], biharmonic [26] or elastic [53, 52, 51] deformations by solving Laplacian, bi-Laplacian or elasticity problems, respectively, and often are referred to as solid-extension mesh moving techniques (SEMMT).

Here we assume that domain deformations result from the harmonic extension of a boundary displacement; other options could be considered as well, without impacting too much on the designed reduction workflow. In all these cases, indeed, we can set

$$\Omega(\boldsymbol{\mu}) = \{\vec{x}(\boldsymbol{\mu}) \in \mathbb{R}^3 : \ \vec{x}(\boldsymbol{\mu}) = \vec{x} + \vec{d}(\boldsymbol{\mu}), \ \vec{x} \in \Omega^0\}, \tag{28}$$

where Ω^0 is a given, reference domain and $\vec{d}(\boldsymbol{\mu})$ is the solution of the following variational problem: given $\boldsymbol{\mu} \in \mathcal{D}$, find the displacement field $\vec{d}(\boldsymbol{\mu}) \in V_d$, such that

$$a_d(\vec{d}(\boldsymbol{\mu}), \vec{w}; \boldsymbol{\mu}) = f_d(\vec{w}, \boldsymbol{\mu}), \qquad \forall \vec{w} \in V_d \tag{29}$$

where V_d is a suitable Hilbert space. Problem (29) arises, for instance, when an harmonic or solid extension is considered to extend a boundary data to the whole fluid domain, see e.g. [50, 2, 52]. By relying, e.g., on the FE method, problem (29) yields a linear system to solve,

$$\mathbf{A}^d(\boldsymbol{\mu})\mathbf{d}(\boldsymbol{\mu}) = \mathbf{f}^d(\boldsymbol{\mu}), \tag{30}$$

where $\mathbf{A}^d(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^d \times N_h^d}$ is the FE matrix obtained[1] from $a_d(\cdot, \cdot; \boldsymbol{\mu})$ and the right hand side $\mathbf{f}^d(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^d}$ is obtained from $f_d(\cdot, \boldsymbol{\mu})$. Here $\mathbf{f}^d(\boldsymbol{\mu})$ encodes the action of the nonhomogeneous conditions imposed on the deformable portion of the boundary; in particular, we can prescribe either a boundary displacement, or a condition on the *stress load*, according to suitable Dirichlet or Neumann conditions, respectively. In this work we consider both these options, relying on the choice of suitable $\boldsymbol{\mu}$-dependent analytic functions to model deformations occurring on either a large or a small portion of the boundary.

[1]Note that, very often, the dependence of problem (30) on the parameters defining the family of boundary deformations is only through its right-hand side $\mathbf{f}^d(\boldsymbol{\mu})$; additional parameters could be employed to specify some features of the differential operator, e.g., when dealing with elastic deformations.

Remark 4.1. *The corresponding meshes are also taken as a deformation of a reference mesh, hence not affecting the topology of the degrees of freedom. Denoting by Ω_h^0 the computational mesh over which the state problem is solved, solving* (30) *yields a deformed volumetric mesh* $\Omega_h(\boldsymbol{\mu}) = \{\mathbf{x}_h \in \mathbb{R}^3 : \mathbf{x}_h(\boldsymbol{\mu}) = \mathbf{x}_h + \mathbf{d}(\boldsymbol{\mu}), \, \mathbf{x}_h \in \Omega_h^0\}$ *where the nodes position is modified (so that* $\Omega_h(\boldsymbol{\mu})$ *conforms to the updated boundary) while keeping fixed the mesh connectivity.*

Hence, when aiming at solving the NS system (10) for any new instance of the geometric parameters, the FE problem (30) describing the deformation must first be solved. Additionally, to handle non homogeneous Dirichlet boundary conditions in the NS system (9), a suitable lifting function must be determined. In nontrivial geometries, this task entails a third problem to be solved: given $\boldsymbol{\mu} \in \mathcal{D}$, find $\vec{l}(\boldsymbol{\mu}) \in V_l$ such that

$$a_l(\vec{l}(\boldsymbol{\mu}), \vec{w}; \boldsymbol{\mu}) = f_l(\vec{w}, \boldsymbol{\mu}), \qquad \forall \vec{w} \in V_l, \tag{31}$$

where $V_l = V_l(\boldsymbol{\mu})$ is a proper Hilbert space and $\vec{l}(\boldsymbol{\mu})$ is the lifting function. As for (29), problem (31) can be discretized by the FE element method, yielding the following linear system:

$$\mathbf{A}^l(\boldsymbol{\mu})\mathbf{l}(\boldsymbol{\mu}) = \mathbf{f}^l(\boldsymbol{\mu}), \tag{32}$$

with $\mathbf{l}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^l}$, $\mathbf{A}^l(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^l \times N_h^l}$ and $\mathbf{f}^l(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^l}$.

When performing the offline phase to build the HROM for the fluid problem (Algorithm 2), problems (30) and (32) must be solved for each $\boldsymbol{\mu}_i$, $i = 1, \dots, n_s$. The same operations are required during the online phase, when the NS-HROM is queried for new instances of the parameter; even though the NS problem is typically the most demanding one, solving problems (30) and (32) may hamper the overall efficiency of the method. To speed up their evaluation, we sequentially build two HROMs – again, by relying on the RB method – for the efficient approximation of the displacement $\mathbf{d}(\boldsymbol{\mu})$ and the lifting function $\mathbf{l}(\boldsymbol{\mu})$:

- we rely on POD to build a N_d-dimensional RB matrix $\mathbf{V}^d \in \mathbb{R}^{N_h^d \times N_d}$ for approximating the deformation, exploiting MDEIM and DEIM to construct an affine decomposition of $\mathbf{A}^d(\boldsymbol{\mu})$ and $\mathbf{f}^d(\boldsymbol{\mu})$, respectively. Then, $\mathbf{d}(\boldsymbol{\mu}) \approx \mathbf{V}^d \mathbf{d}_N(\boldsymbol{\mu})$, where $\mathbf{d}_N(\boldsymbol{\mu}) \in \mathbb{R}^{N_d}$ solves

  $$\widetilde{\mathbf{A}}_N^d(\boldsymbol{\mu})\mathbf{d}_N(\boldsymbol{\mu}) = \widetilde{\mathbf{f}}_N^d(\boldsymbol{\mu}); \tag{D-HROM}$$

  the arrays $\widetilde{\mathbf{A}}_N^d(\boldsymbol{\mu}) \in \mathbb{R}^{N_d \times N_d}$ and $\widetilde{\mathbf{f}}_N^d(\boldsymbol{\mu}) \in \mathbb{R}^{N_d}$ appearing in the RB problem (D-HROM) above are obtained by Galerkin projection of the corresponding FOM arrays onto the subspace spanned by the columns of $\mathbf{V}^d$;

- similarly, we rely on POD to build an N_l-dimensional RB projection matrix $\mathbf{V}^l$ for approximating the lifting function, exploiting MDEIM and DEIM to construct an affine decomposition of $\mathbf{A}^l(\boldsymbol{\mu})$ and $\mathbf{f}^l(\boldsymbol{\mu})$, respectively. The RB problem (D-HROM) is used in view of the computation of the snapshots $\{\mathbf{l}(\boldsymbol{\mu}_i^l)\}_{i=1}^{n_s^l}$ required by POD, and $\mathbf{A}^l(\boldsymbol{\mu}_i^l)$, $\mathbf{f}^l(\boldsymbol{\mu}_i^l)$ required by (M)DEIM. Finally, $\mathbf{l}(\boldsymbol{\mu}) \approx \mathbf{V}^l \mathbf{l}_N(\boldsymbol{\mu})$, where $\mathbf{l}_N(\boldsymbol{\mu}) \in \mathbb{R}^{N_l}$ solves

  $$\widetilde{\mathbf{A}}_N^l(\boldsymbol{\mu})\mathbf{l}_N(\boldsymbol{\mu}) = \widetilde{\mathbf{f}}_N^l(\boldsymbol{\mu}); \tag{L-HROM}$$

  as before, $\widetilde{\mathbf{A}}_N^l(\boldsymbol{\mu}) \in \mathbb{R}^{N_l \times N_l}$ and $\widetilde{\mathbf{f}}_N^l(\boldsymbol{\mu}) \in \mathbb{R}^{N_l}$ are obtained by Galerkin projection of the corresponding FOM arrays onto $\mathbf{V}^l$.

Problems (D-HROM) and (L-HROM) are then used in the construction of (NS-HROM) in Algorithm 2. The complete procedure is outlined in Figure 1.

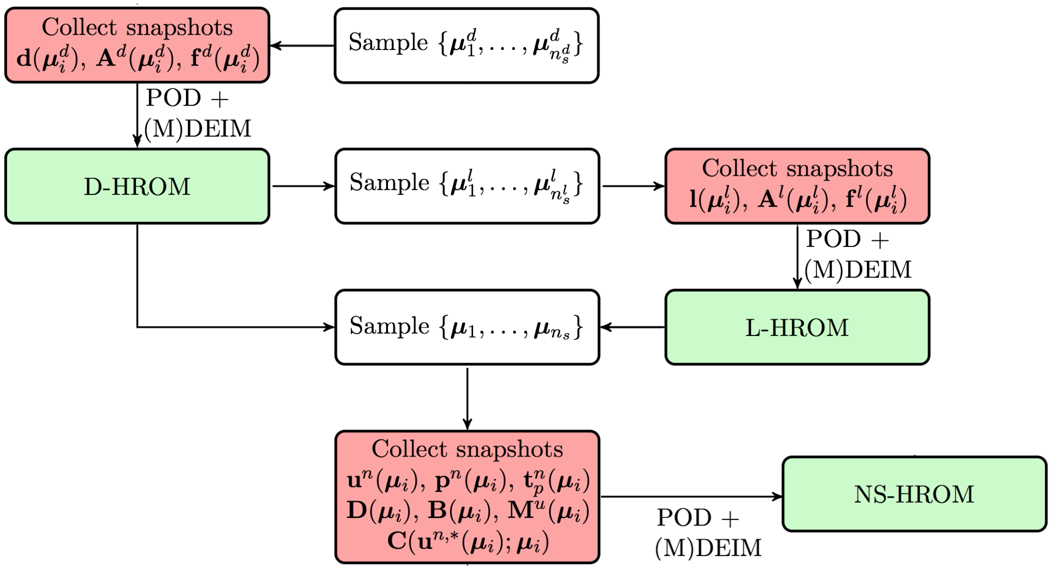

Collect snapshots
$\mathbf{d}(\boldsymbol{\mu}_i^d)$, $\mathbf{A}^d(\boldsymbol{\mu}_i^d)$, $\mathbf{f}^d(\boldsymbol{\mu}_i^d)$
Sample $\{\boldsymbol{\mu}_1^d, \dots, \boldsymbol{\mu}_{n_s^d}^d\}$
POD + (M)DEIM
D-HROM
Sample $\{\boldsymbol{\mu}_1^l, \dots, \boldsymbol{\mu}_{n_s^l}^l\}$
Collect snapshots
$\mathbf{l}(\boldsymbol{\mu}_i^l)$, $\mathbf{A}^l(\boldsymbol{\mu}_i^l)$, $\mathbf{f}^l(\boldsymbol{\mu}_i^l)$
POD + (M)DEIM
Sample $\{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{n_s}\}$
L-HROM
Collect snapshots
$\mathbf{u}^n(\boldsymbol{\mu}_i)$, $\mathbf{p}^n(\boldsymbol{\mu}_i)$, $\mathbf{t}_p^n(\boldsymbol{\mu}_i)$
$\mathbf{D}(\boldsymbol{\mu}_i)$, $\mathbf{B}(\boldsymbol{\mu}_i)$, $\mathbf{M}^u(\boldsymbol{\mu}_i)$
$\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i)$
POD + (M)DEIM
NS-HROM

Figure 1: Offline strategy to build NS-HROM. Red blocks highlight collection of solution, matrix and right hand side snapshots, while green blocks refer to HROMs construction.

Remark 4.2. *As a matter of fact, by replacing problem* (30) *with its reduced counterpart* (D-HROM) *we introduce an automatic,* low-dimensional *parametrization of domain deformations. For any* $\boldsymbol{\mu} \in \mathcal{D}$*, the corresponding volumetric mesh we actually deal with is given by* $\Omega_N(\boldsymbol{\mu}) = \{\mathbf{x}_N \in \mathbb{R}^3 : \mathbf{x}_N(\boldsymbol{\mu}) = \mathbf{x}_h + \mathbf{V}^d \mathbf{d}_N(\boldsymbol{\mu}),\ \mathbf{x}_h \in \Omega_h^0\}$*. The smaller the error between* $\mathbf{d}(\boldsymbol{\mu})$ *and* $\mathbf{V}^d\mathbf{d}_N(\boldsymbol{\mu})$ *– this is indeed ensured by the Galerkin-RB formulation of problem* (D-HROM) *– the more accurate the approximation* $\Omega_N(\boldsymbol{\mu})$ *of* $\Omega_h(\boldsymbol{\mu})$*.*

5 Numerical results for NS-HROM

We show in this section numerical results provided by the proposed ROM workflow on two different cases: the former deals with a simpler fluid flow in a parametrized cylinder, and allows to assess accuracy and efficiency of each ROM; the latter deals instead with a more involved blood flow in carotid bifurcation, showing the capability of the proposed framework to provide reliable evaluations of possible outputs of interest.

5.1 Test case I: NS flow in a parametrized cylinder

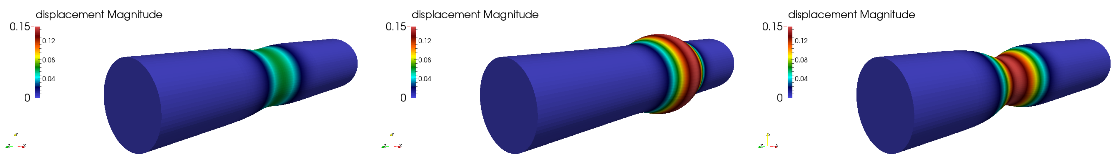
We first consider a fluid flow (viscosity $\nu = 0.01$) over the time interval $(0, 0.5)$ in a parametrized three-dimensional cylinder $\Omega(\boldsymbol{\mu}) \subset \mathbb{R}^3$. We start from a reference domain $\Omega^0 = \{\vec{x} \in \mathbb{R}^3 : x_1^2 + x_1^2 < 0.25,\ x_3 \in (0, 5)\}$ for which $\Gamma_{in} = \partial\Omega^0 \cap \{x_3 = 0\}$, $\Gamma_{out} = \partial\Omega^0 \cap \{x_3 = 3\}$, and $\Gamma_w = \partial\Omega \setminus (\Gamma_{in} \cup \Gamma_{out})$. The $\boldsymbol{\mu}$-dependent domain $\Omega(\boldsymbol{\mu})$ is generated by deforming Ω^0 as in (28), by a displacement $\vec{d}(\boldsymbol{\mu})$ obtained as the solution of the (vector) Laplace problem

$$\begin{cases} -\Delta \vec{d}(\boldsymbol{\mu}) = \vec{0} & \text{in } \Omega^0 \\ \vec{d}(\boldsymbol{\mu}) = \vec{h}(\boldsymbol{\mu}) & \text{on } \partial\Omega^0; \end{cases} \tag{33}$$

boundary portions $\Gamma_{in}(\boldsymbol{\mu})$, $\Gamma_{out}(\boldsymbol{\mu})$, $\Gamma_w(\boldsymbol{\mu})$ are obtained correspondingly. For the case at hand, displacement $\vec{d}(\boldsymbol{\mu})$ then results from the harmonic extension of a boundary deformation

$$\vec{h}(\boldsymbol{\mu}) = \begin{bmatrix} -x_1\mu_1 \exp\{-5(x_3-\mu_2)^2\} \\ -x_2\mu_1 \exp\{-5(x_3-\mu_2)^2\} \\ 0 \end{bmatrix},$$

entailing either a narrowing or an enlargement (according to the sign of μ_1) of the cylinder section at different positions along the coordinate x_3 (according to the value of μ_2). In our numerical experiments we take $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathcal{D} = [-0.3, 0.3] \times [2, 3]$. The numerical approximation $\vec{d}(\boldsymbol{\mu})_h$ of $\vec{d}(\boldsymbol{\mu})$, solution of problem (30), is built by employing the FE method (with $\mathbb{P}^2$ basis functions). Note that in the case (33), the FE matrix $\mathbf{A}^d(\boldsymbol{\mu})$ is $\boldsymbol{\mu}$-independent. In Figure 2, the deformation $\mathbf{d}_h(\boldsymbol{\mu})$ is reported for three different values of $\boldsymbol{\mu} \in \mathcal{D}$.



(a) $\boldsymbol{\mu} = (2.7, 0.12)$ (b) $\boldsymbol{\mu} = (2, -0.3)$ (c) $\boldsymbol{\mu} = (3, 0.3)$

Figure 2: Displacement for different values of $\boldsymbol{\mu}$.

Once the computational domain has been deformed, the lifting function $\vec{r}_{\vec{g}_D}(\boldsymbol{\mu})$ is computed similarly by solving the following problem

$$\begin{cases} -\Delta \vec{r}_{\vec{g}_D}(\boldsymbol{\mu}) = \vec{0} & \text{in } \Omega(\boldsymbol{\mu}) \\ \vec{r}_{\vec{g}_D}(\boldsymbol{\mu}) = \vec{g}_{NS}(\boldsymbol{\mu}) & \text{on } \Gamma_{in}(\boldsymbol{\mu}) \\ \vec{r}_{\vec{g}_D}(\boldsymbol{\mu}) = \vec{0} & \text{on } \Gamma_w(\boldsymbol{\mu}) \\ \partial_{\vec{n}(\boldsymbol{\mu})} \vec{r}_{\vec{g}_D}(\boldsymbol{\mu}) = \vec{0} & \text{on } \Gamma_{out}(\boldsymbol{\mu}), \end{cases} \tag{34}$$

which is an harmonic extension of the $\boldsymbol{\mu}$-dependent component $\vec{g}_{NS}(\boldsymbol{\mu})$ of the Dirichlet data appearing in (1) (see equation (2)). For the case at hand, $\vec{g}_{NS}(\boldsymbol{\mu})$ is taken as a parabolic profile vanishing on Γ_w and such that the flow rate at the inlet $\Gamma_{in}(\boldsymbol{\mu})$ is equal to 1; regarding the time dependent contribution, we take $w(t) = \sin(2\pi t)$. Problem (34), too, is discretized with the FE method with $\mathbb{P}^2$ basis functions, leading to the parametrized linear system (32). The FE solver (AMG-preconditioned GMRES) takes on average 0.55 and 0.41 seconds for the deformation and lifting problem, respectively (a stopping criterion of 10^{-9} on the FE residual rescaled with the Euclidean norm of the right hand side is used).

Finally, regarding the FOM for the NS system, we use a computational domain with 13'603 vertices and Taylor-Hood FE spaces, that is with $\mathbb{P}^2 - \mathbb{P}^1$ basis functions, leading to $N_h^u = 306'735$ and $N_h^p = 13'603$ degrees of freedom for velocity and pressure, respectively, and a total dimension $N_h = 320'338$ of the FOM. We employ the BDF2 method with $\Delta t = 0.01$ for the time discretization. The FE problem (10) is solved with FGMRES preconditioned with a SIMPLE preconditioner, where the solves (steps 1 and 2 in Algorithm 1) are carried out by inner iterations up to a tolerance of 10^{-5} using an Additive Schwarz preconditioner from the `Ifpack` package of `Trilinos`.

5.1.1 Offline phase

The offline phase consists of the subsequent construction of HROMs for the domain deformation, the lifting function and the fluid flow; below we detail the three stages.

(D-HROM) *construction.* The offline phase is carried out with $n_s^d = 30$ snapshots for POD and DEIM (MDEIM is not employed since the matrix $\mathbf{A}^d(\boldsymbol{\mu})$ is $\boldsymbol{\mu}$-independent); the singular value decompositions (SVDs) corresponding to the construction of the RB matrix $\mathbf{V}^d$ and the DEIM affine approximation are reported in Figure 3a. By setting a tolerance of 10^{-7} we come up with $N_d = 11$ RB function for the state approximation and $Q_f^d = 11$ DEIM basis functions for approximating the right hand side with $\widetilde{\mathbf{f}}_N^d(\boldsymbol{\mu})$; in Figure 3a the singular values corresponding to POD and DEIM are reported. As a matter of fact, only a few RB functions are necessary to accurately approximate the solution of (33). The offline phase takes $t_{\text{off}} = 51.17$ seconds; by testing online the HROM for the deformation on 50 parameter instances, we obtain an average FE residual $r_{\text{RB}} = 5.5 \cdot 10^{-7}$, with a solution computed in 0.026 seconds on average, see Table 1. This yields a computation about 10 times faster than the one entailed by solving the FE problem, which in this context represents a relevant boost, since the deformation problem is solved for each snapshot toward the construction of the lifting HROM (L-HROM) and the NS-HROM (see Figure 1).

Table 1: D-HROM: POD for state reduction and DEIM have been run with $\varepsilon_{\text{POD}} = \delta_{\text{deim}} = 10^{-7}$. Computational times are expressed in seconds.

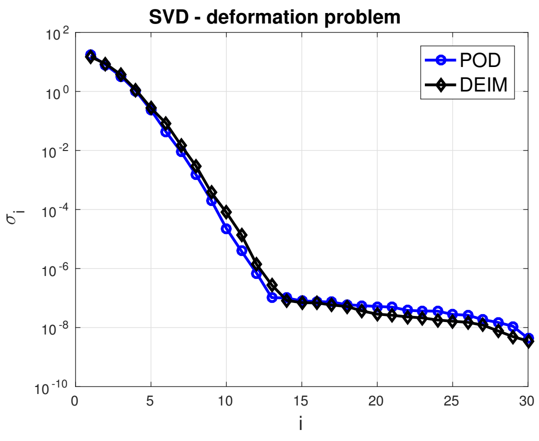
N_d	Q_f^d	Q_a^d	r_{RB}	$t_{\text{RB}}^{\text{onl}}$	t_{FE}	n_s^d	t_{off}
11	1	11	5.5e-7	0.026	0.25	30	25.53

(L-HROM) *construction.* The HROM for the lifting function is fed with the approximated deformation computed above. The offline phase is performed with $n_s^l = 150$ snapshots for POD, DEIM and MDEIM; the singular value decompositions (SVDs) corresponding to the construction of the RB matrix $\mathbf{V}^l$ and the (M)DEIM affine approximations are reported in Figure 3b. By setting a tolerance of $\varepsilon_{\text{POD}} = 10^{-7}$ we come up with $N_l = 42$ RB function for the state approximation, $Q_f^l = 22$ DEIM basis functions for approximating the right hand side and $Q_a^l = 44$ MDEIM basis matrices for $\mathbf{A}^l(\boldsymbol{\mu})$; the offline phase takes $t_{\text{off}} = 89.13$. By testing online the HROM for the lifting on 50 parameter instances, we obtain an average FE residual $r_{\text{RB}} = 7.3 \cdot 10^{-6}$, with a solution computed in 0.03 seconds on average, see Table 2. The resulting ROM yields a speed up of about 14 with respect to the solution of the FE linear system, similarly to the D-HROM.

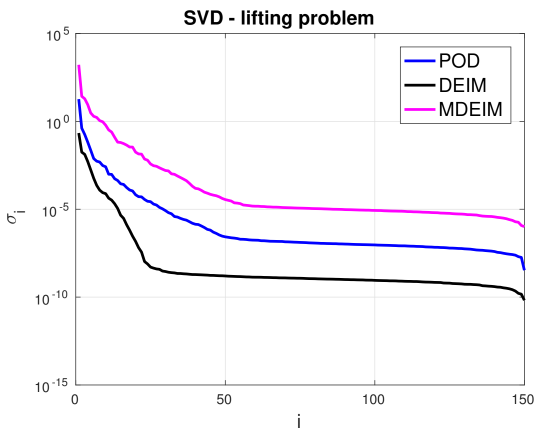
Table 2: L-HROM: POD for state reduction and (M)DEIM have been run with $\varepsilon_{\text{POD}} = \delta_{\text{deim}} = \delta_{\text{mdeim}} = 10^{-7}$. Computational times are expressed in seconds.

N_l	Q_f^l	Q_a^l	r_{RB}	$t_{\text{RB}}^{\text{onl}}$	t_{FE}	n_s^l	t_{off}
42	22	44	7.3e-6	0.030	0.41	50	89.13

(NS-HROM) *construction.* The most demanding stage of the offline phase is the construction of the HROM for the NS equations, on top of the previous two. To this aim, we employ $n_s = 50$ snapshots for the state reduction, keeping fixed the tolerance for the POD in time to $\varepsilon_t = 10^{-7}$, and varying the tolerance ε_μ, for building the final velocity and pressure RB spaces. We highlight that for the enriching RB matrices $\mathbf{V}_s^i$, $i = 1, \ldots, n_s$ and the final $\mathbf{V}_s$, we use a tolerance equal to $\varepsilon_t/10$ and $\varepsilon_\mu/10$, respectively. On average, the POD in time retains 20 basis functions for the velocity, 13 for the pressure and 20 for the enriching functions. Thus, the RB matrices $\mathbf{V}_u, \mathbf{V}_p, \mathbf{V}_s$ are built with POD starting from 1000, 1000 and 645 snapshots, respectively, and their dimensions depend on the value of ε_μ.



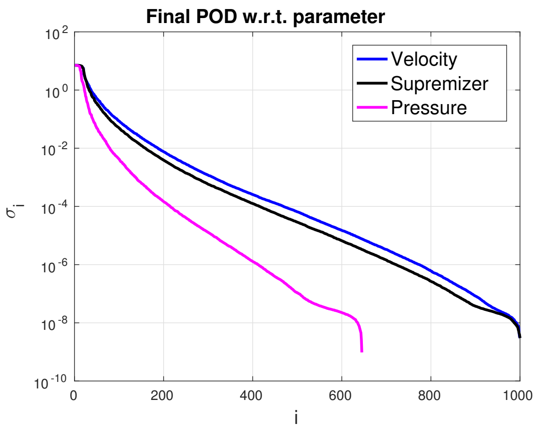
(a) SVD of POD (state reduction) and DEIM (system approximation) for building D-HROM.



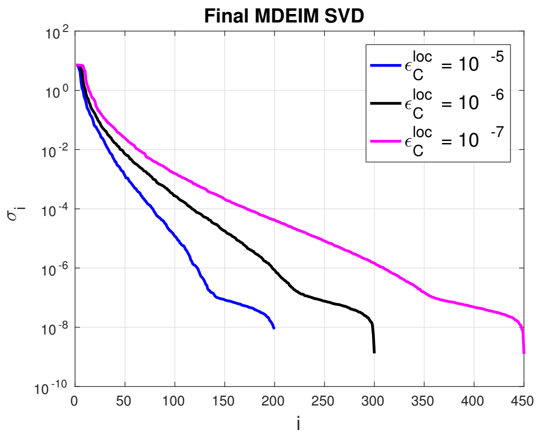
(b) [SVD of POD (state reduction) and (M)DEIM (system approximation) for building L-HROM.

Figure 3: SVDs for building D-HROM and L-HROM problems.

Regarding the system approximation, the construction of an approximate affine decomposition (through MDEIM) of $\mathbf{D}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$ and $\mathbf{M}^u(\boldsymbol{\mu})$, and an approximation of $\mathbf{f}_1(\boldsymbol{\mu})$ and $\mathbf{f}_2(\boldsymbol{\mu})$ (through DEIM) employs $n_s = 150$ snapshots, with tolerances δ_{mdeim} and δ_{deim} which are varied in the experiments. Similarly, the influence of ε_C^{loc} for computing the basis in time with MDEIM of the linearized term and the tolerance ε_C for the ultimate MDEIM is analyzed. A summary of the considered combinations of these tolerances is reported in Table 3; in general, we choose $\delta_{\text{mdeim}} = \delta_{\text{deim}} = \varepsilon_C^{loc} < \varepsilon_C$, where the latter inequality is motivated by the fact that a proper sampling in time must be carried out in order to obtain an accurate affine approximation of the term $\mathbf{C}(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$.



(a) SVD of final POD w.r.t. $\boldsymbol{\mu}$ to build the RB matrices $\mathbf{V}_u$, $\mathbf{V}_s$, $\mathbf{V}_s$.



(b) SVD final MDEIM for building $\mathbf{V}_C$.

Figure 4: SVDs for building final RB spaces (left) and final MDEIM of the linearized term (right). Notice that in the latter, according to the tolerance ε_C^{loc} of the MDEIMs in time, the number of snapshots and the decay of the resulting SVD for the final MDEIM change.

The results of the offline phase corresponding to the settings in Table 3 are reported in Table 4. In particular, N_u, N_s, N_p denote the number of RB functions retained by the PODs

Table 3: Chosen settings for numerical experiments of NS-HROM.

Setting	ε_t	ε_μ	$\delta_{\text{deim}} = \delta_{\text{mdeim}} = \varepsilon_C^{loc}$	ε_C
SET 1	10^{-7}	10^{-3}	10^{-5}	10^{-3}
SET 2	10^{-7}	10^{-3}	10^{-7}	10^{-3}
SET 3	10^{-7}	10^{-3}	10^{-7}	10^{-5}
SET 4	10^{-7}	10^{-5}	10^{-7}	10^{-3}

Table 4: Results of the offline phase for settings defined in Table 3. Computational times are reported in seconds.

Setting	N_u	N_s	N_p	Q_c^t	Q_c	t_{off}
SET 1	210	303	86	4	36	22596.8
SET 2	210	303	86	9	69	22733.7
SET 3	210	303	86	9	203	23610.0
SET 4	512	612	237	9	69	27941.1

for velocity, enriching velocities and pressure, respectively; the corresponding singular values are reported in Figure 4a, showing that the decay of pressure singular values is faster than the one obtained for velocity and supremizers; therefore, fewer modes are retained. The decay of velocity and supremizer snapshots is similar, however by using a smaller tolerance for the latter, $\mathbf{V}_s$ has a larger dimension than $\mathbf{V}_u$. These functions are then merged through a Gram-Schmidt orthonormalization procedure. The total number of RB functions finally ranges from 599 to 1361, depending on the chosen tolerance ε_μ. The number of matrix basis computed by MDEIM in time Q_c^t and the final MDEIM Q_c for the linearized term range from 4 to 9 in the former case, and from 36 to 203 in the latter case, respectively. Singular values decay for the approximation of the linearized term is shown in Figure 4b. By decreasing tolerances, the offline time increases accordingly, since it includes the computation of the RB affine basis for the matrices and right hand sides (see lines 17-18 of Algorithm 2); however, the most demanding stage remains the calculation of the snapshots for the NS problem.

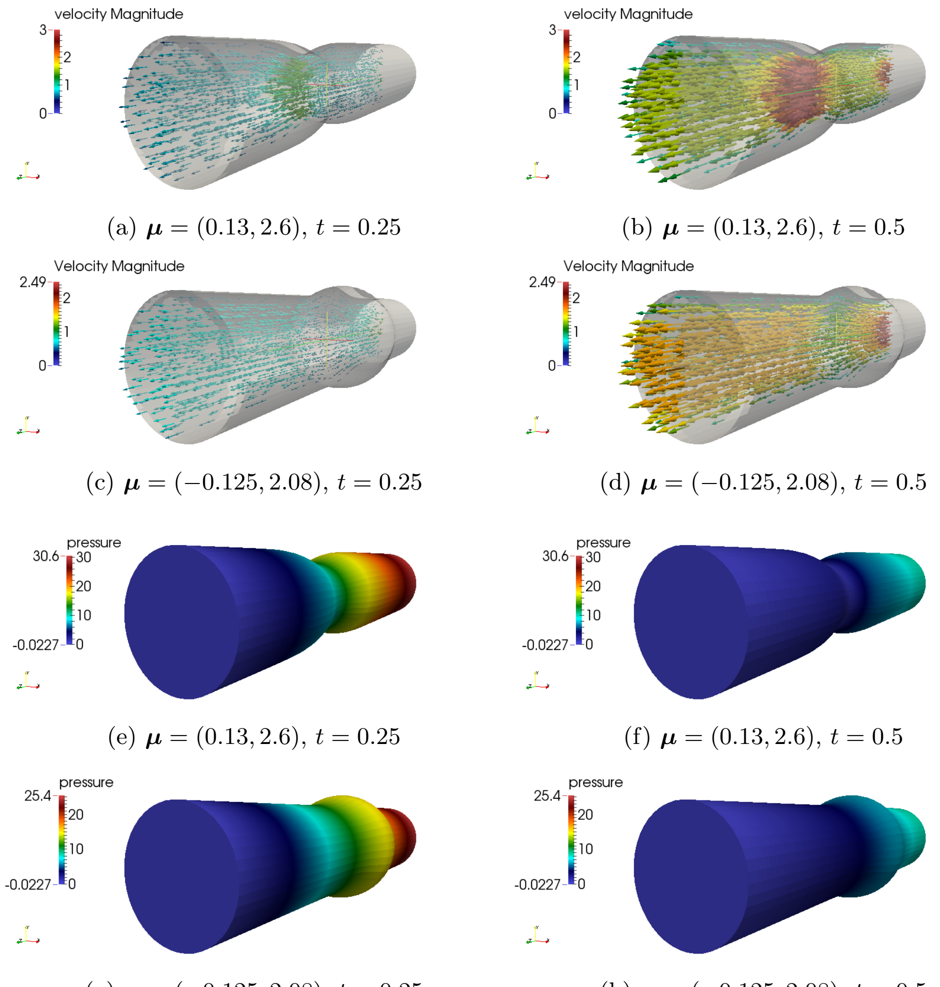
5.1.2 Online phase

During the online phase we assess the four ROMs defined in Table 3 on a test sample of 50 parameter vectors (not including those sampled offline). Examples of the solution of (10) for different values of the parameter and times are reported in Figure 5. In order to assess the quality of the RB approximation, we define

$$e_{\text{RB}}^u(\boldsymbol{\mu}) = \sqrt{\frac{\sum_{n=1}^{N_t} \|\mathbf{u}^n(\boldsymbol{\mu}) - \mathbf{V}_u \mathbf{u}_N^n\|^2_{\mathbf{X}_u}}{\sum_{n=1}^{N_t} \|\mathbf{u}^n(\boldsymbol{\mu})\|^2_{\mathbf{X}_u}}}, \qquad e_{\text{RB}}^p(\boldsymbol{\mu}) = \sqrt{\frac{\sum_{n=1}^{N_t} \|\mathbf{p}^n(\boldsymbol{\mu}) - \mathbf{V}_p \mathbf{p}_N^n\|^2_{\mathbf{X}_p}}{\sum_{n=1}^{N_t} \|\mathbf{p}^n(\boldsymbol{\mu})\|^2_{\mathbf{X}_p}}} \quad (35)$$

as the velocity and pressure relative errors, respectively, on the time interval $[0, T]$. We denote by $\bar{e}_{\text{RB}}^u, \bar{e}_{\text{RB}}^p$ the corresponding quantities averaged on the test sample; their values are reported in Table 5, together with the CPU time $t_{\text{RB}}^{\text{onl}}$ required to compute the RB solution in the online phase and the speedup obtained with respect to the FOM simulation.

We first consider SET 1 and SET 2, in which only the tolerance ε_C^{loc} varies; by decreasing it from 10^{-5} to 10^{-7}, relative errors (35) decrease of one order of magnitude. This is due to the fact that the time trajectory of the linearized term is not well approximated in the first case and the corresponding MDEIM basis in time is not accurate, yielding a poor final MDEIM



(a) $\boldsymbol{\mu} = (0.13, 2.6)$, $t = 0.25$

(b) $\boldsymbol{\mu} = (0.13, 2.6)$, $t = 0.5$

(c) $\boldsymbol{\mu} = (-0.125, 2.08)$, $t = 0.25$

(d) $\boldsymbol{\mu} = (-0.125, 2.08)$, $t = 0.5$

(e) $\boldsymbol{\mu} = (0.13, 2.6)$, $t = 0.25$

(f) $\boldsymbol{\mu} = (0.13, 2.6)$, $t = 0.5$

(g) $\boldsymbol{\mu} = (-0.125, 2.08)$, $t = 0.25$

(h) $\boldsymbol{\mu} = (-0.125, 2.08)$, $t = 0.5$

Figure 5: Velocity (lines 1,2) and pressure (lines 3,4) for different values of parameters and at time $t = 0.25$ (left) and $t = 0.5$ (right).

Table 5: Results averaged on 50 instances of the parameter considered online. Times are reported in seconds.

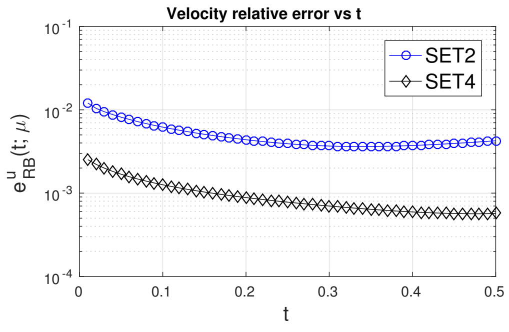
Setting	$\bar{e}^u_{\text{RB}}$	$\bar{e}^p_{\text{RB}}$	$t^{\text{onl}}_{\text{RB}}$	SPEEDUP
SET 1	3.07e-02	3.60e-02	4.48	71
SET 2	1.91e-03	3.80e-04	9.60	33
SET 3	1.88e-03	1.52e-04	10.96	29
SET 4	3.51e-04	1.04e-04	32.08	10

approximation. This fact can also be highlighted by the decay of singular values of MDEIM in parameter, in which the bases in time have been computed using $\varepsilon_C^{loc} = 10^{-5}, 10^{-6}$ and 10^{-7}; the decay of the singular values significantly changes by considering smaller values of ε_C^{loc}. As a matter of fact, the number of computed affine terms is 36 for SET 1 and 69 for SET 2, even though the same tolerance ε_C has been used (cf. Table 4). In the next experiments we consider $\varepsilon_C^{loc} = 10^{-7}$.

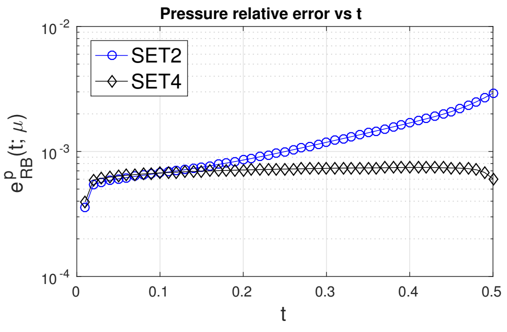
Let us now consider SET 3 and SET 4, where either ε_C or ε_μ is decreased to 10^{-5}. The former option has not a beneficial impact on the solution accuracy, since the error is almost constant: error convergence is indeed hampered by the too coarse state reduction. The latter option yields an improvement of the RB approximation accuracy, especially for the velocity. This fact is confirmed by considering the velocity and pressure relative errors

$$e^u_{\mathrm{RB}}(t_n;\boldsymbol{\mu}) = \frac{\|\mathbf{u}^n(\boldsymbol{\mu}) - \mathbf{V}_u\mathbf{u}^n_N\|^2_{\mathbf{X}_u}}{\|\mathbf{u}^{t_n}(\boldsymbol{\mu})\|^2_{\mathbf{X}_u}} \qquad e^p_{\mathrm{RB}}(t_n;\boldsymbol{\mu}) = \frac{\|\mathbf{p}^n(\boldsymbol{\mu}) - \mathbf{V}_p\mathbf{p}^n_N\|^2_{\mathbf{X}_u}}{\|\mathbf{p}^n(\boldsymbol{\mu})\|^2_{\mathbf{X}_u}}$$

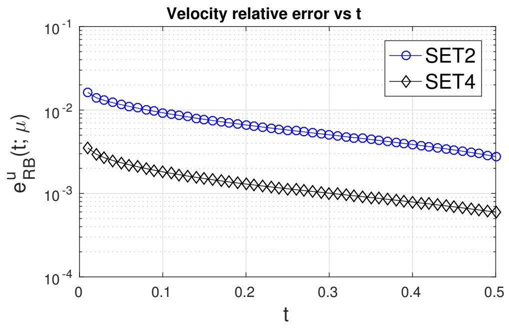
as function of the time step t_n, which are reported for two randomly selected values of the parameters in Figure 6 for both SET 2 and SET 4.



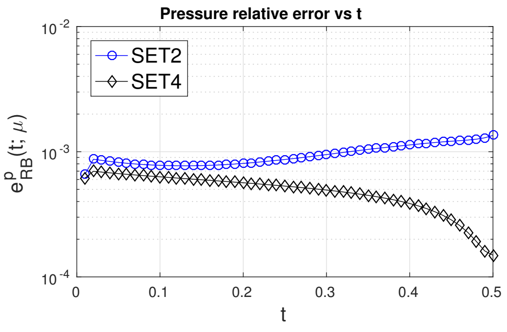
(a) Velocity relative error $\boldsymbol{\mu} = (0.27, 2.95)$.



(b) Pressure relative error $\boldsymbol{\mu} = (0.27, 2.95)$.



(c) Velocity relative error $\boldsymbol{\mu} = (-0.27, 2.05)$.



(d) Pressure relative error $\boldsymbol{\mu} = (-0.27, 2.05)$.

Figure 6: Velocity and pressure errors entailed by the NS-HROM as function of time for two values of the parameter.

As a matter of fact, the velocity is approximated at every time step with the same discrepancy by the HROM in the SET 2 and SET 4 settings; the pressure error follows instead different paths according to the employed setting.

Regarding the CPU time required to compute the RB approximation of NS equations, it ranges from about 4.5 to 32 seconds, with a speed up which varies from 10 to 71 times compared to the FOM solution, whose computational cost is on average 318.16 seconds. The HROM we have constructed therefore entails a significant speed up for the computation of an accurate solution of the unsteady NS problem. Furthermore, the new treatment of the nonlinear term shows to be a reliable option for the efficient assembling of the RB system in the NS case.

5.2 Blood flows in carotid bifurcations

The second test case we deal with is related with the efficient characterization of blood flows in a subject-specific three dimensional carotid bifurcation, where both the domain and inlet boundary conditions are parametrized, in terms of both flow variables and derived outputs of interest. The carotid bifurcation is located along the sides of the neck and furnishes the blood supply to the face and the brain [60]. Three branches can be distinguished: the common carotid artery (CCA) which then splits in the internal carotid artery (ICA) and the external carotid artery (ECA), see Figure 7a. In adult age, the carotid bifurcation may be subject to atherosclerosis, that is a narrowing of the artery in the bifurcation region, which might ultimately lead to stroke in most of the patients. The fluid dynamics of blood plays an important role in the development of such disease and CFD can be of help in the prediction of possible diseases. One of the main indicators employed in the risk analysis is distribution of the wall shear stresses (WSSs) occurring at the bifurcation [49], in this perspective, numerical simulations can play a relevant role in providing quantitative results able to support clinicians; recent results are reported, e.g., in [31, 24]. Here we exploit the proposed ROM workflow to investigate the behavior of blood flows when different physical and geometrical configurations described in terms of parameters are considered.

5.2.1 Test case setting

To start with, we define a family of parametrized geometrical configurations by deforming the reference domain shown in Figure 7a; deformation here is obtained as the harmonic extension of a Neumann boundary datum, by solving the following (vector) Laplace problem

$$\begin{cases} -\Delta \vec{d}(\boldsymbol{\mu}) = \vec{0} & \text{in } \Omega(\boldsymbol{\mu}) \\ \vec{d}(\boldsymbol{\mu}) = \vec{0} & \text{on } \Gamma_{in} \cup \Gamma_{out} \\ \dfrac{\partial \vec{d}(\boldsymbol{\mu})}{\partial \vec{n}} = \vec{h}(\boldsymbol{\mu}) & \text{on } \Gamma_w. \end{cases} \tag{36}$$

Here Γ_{in} represents the CCA inlet boundary portion, located at the bottom of the bifurcation in Figure 7a; Γ_{out} is given by the two ECA and ICA outflow boundaries, located on its top; finally, $\Gamma_w = \partial\Omega \setminus (\Gamma_{in} \cup \Gamma_{out})$. Here the parametrized datum $\vec{h}(\boldsymbol{\mu})$ represents a stress load entailing a deformation which narrows the two branches of the bifurcation,

$$\vec{h}(\boldsymbol{\mu}) = h(\vec{x};\boldsymbol{\mu}) = -\mu_1(1 - r^2(\vec{x}))\vec{n}\mathcal{X}_A(\vec{x}), \qquad \vec{x} \in \mathbb{R}^3,$$

where the portion A of the boundary where $\vec{h}(\boldsymbol{\mu})$ acts as a load is defined as

$$A = \left\{\vec{x} \in \mathbb{R}^3 : \, r^2 \leq R^2\right\} \cap \partial\Omega(\boldsymbol{\mu}), \qquad r^2 = r^2(\vec{x}) = x_1^2 + (x_2 - 2.5)^2 + x_3^2. \tag{37}$$

Here μ_1 is a parameter determining the magnitude of the load and $\mathcal{X}_A(\vec{x})$ is the indicator function equal to 1 on A and vanishing otherwise. The region identified by the set A is located at the separation of the CCA in the ECA and the ICA, see Figure 7b.

By following the setup employed in [31], at the CCA inlet boundary we prescribe a parametrized flow rate $Q_{\text{CCA}}(t;\boldsymbol{\mu})$, obtained as a suitable modification of the reference flow rate $Q^0_{\text{CCA}}(t)$, which has been acquired from echo-color Doppler and is reported in Figure 7c for a single heartbeat. The resulting inlet velocity $\vec{g}_{NS}(\boldsymbol{\mu})$ to be prescribed is the unique parabolic function, in the normal direction to $\Gamma_{in}(\boldsymbol{\mu})$ and vanishing in the tangential ones, such that

$$\int_{\Gamma_{in}} \vec{g}_{NS}(t;\boldsymbol{\mu}) \cdot \vec{n} \, \mathrm{d}\Gamma_{in} = Q_{\mathrm{CCA}}(t;\boldsymbol{\mu}) = \mu_2 \, Q^0_{\mathrm{CCA}}(t).$$

We highlight that assuming a parabolic profile at the inlet represents a proper choice when dealing with carotid bifurcations, see e.g. [9].

The parameter vector for the case at hand is $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathcal{D} = [0.2, 0.4] \times [0.85, 1.0] \subset \mathbb{R}^2$; the value of μ_1 entails a narrowing of the bifurcation, thus simulating the effect of a stenosis obstructing the vessel; μ_2 determines instead the magnitude of the flow rate at the inlet entering the CCA. The radius at the inlet boundary at the entrance of the CCA measures approximately 0.27cm, leading to a peak of the inlet velocity profile of approximately 59 cm s^{-1}, when $\mu_2 = 1$, during the systolic phase. Two examples of deformation with respect to the reference domain are reported in Figure 8 for different instances of the parameter $\boldsymbol{\mu}^1 = (0.375, 0.975)$ and $\boldsymbol{\mu}^2 = (0.225, 0.875)$. Finally, the blood kinematic viscosity is chosen as $\nu = 0.035\mathrm{cm}^2\mathrm{s}^{-1}$, which represents a physiological value. By using these characteristic length and velocity, the resulting Reynolds number is about $Re \approx 450$.

Taylor-Hood ($\mathbb{P}^2 - \mathbb{P}^1$) finite elements are employed for the spatial discretization, leading to $N_h^u = 248'019$ dofs for the velocity and $N_h^p = 11'911$ for the pressure, respectively, such that $N_h = N_h^u + N_h^p = 259'930$, and the BDF2 scheme with $\Delta t = 0.02$ for the time discretization. In order to simulate an entire heartbeat, we take $T = 0.64$ seconds as final time. The deformation problem (36) is discretized by means of the FE method and solved with the AMG-preconditioned CG up to a tolerance of 10^{-7}. A similar procedure has been followed to calculate the lifting function; here we omit the details for the sake of space. Numerical simulations have been carried out by employing 32 cores.

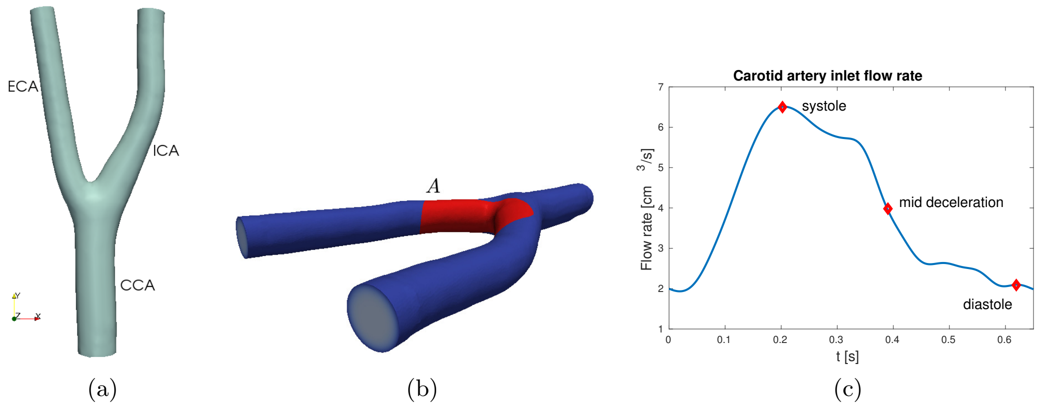
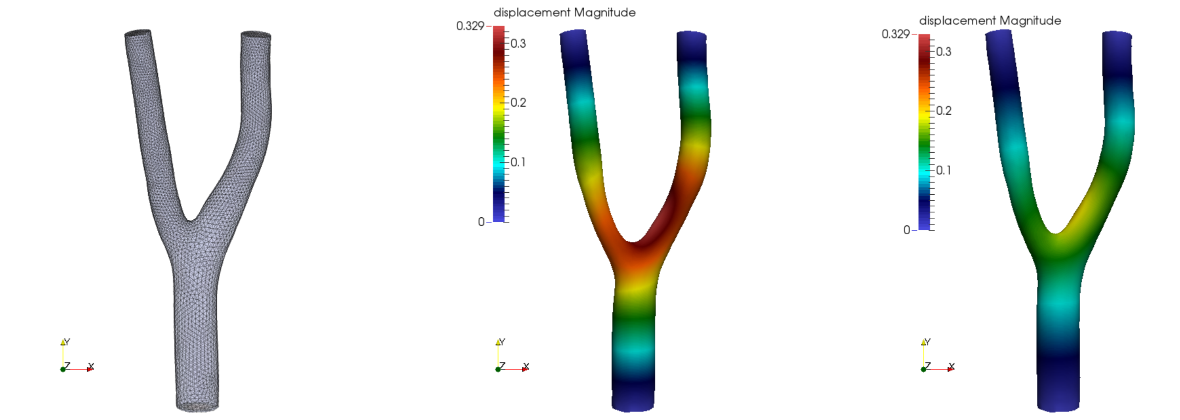


Figure 7: Left: Reference domain $\Omega(\boldsymbol{\mu})$; common carotid artery (CCA), internal carotid artery (ICA) and external carotid artery (ECA). Center: Region A (as defined in (37)) where the stress $\vec{h}(\boldsymbol{\mu})$ is applied. Right: reference inlet flow rate $Q^0_{\mathrm{CCA}}(t)$ $[\mathrm{cm}^3\mathrm{s}^{-1}]$ with highlighted systole, mid deceleration and diastole phases.

We employ in the offline phase $n_s = 20$ parameter instances $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$, chosen on a 5×4 tensor grid with equidistant points of $\mathcal{D}$. These parameter instances represent the training set used to compute the matrix snapshots for the affine approximations of the matrices $\mathbf{D}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$ and $\mathbf{M}^u(\boldsymbol{\mu})$, which leads to $Q_a = Q_d + Q_b + Q_m = 12$ affine terms for their approximation in total. Similarly, an affine approximation of the right-hand sides made by $Q_g^1 + Q_g^2 = 19$ terms is obtained by DEIM. The same offline parameters are then employed to construct the solution snapshots (each one consisting of $N_t = T/\Delta t = 0.64/0.02 = 32$ time steps).



(a) Starting mesh for $\Omega(\boldsymbol{\mu})$. (b) $\boldsymbol{\mu}^1 = (0.375, 0.975)$. (c) $\boldsymbol{\mu}^2 = (0.225, 0.875)$.

Figure 8: Starting mesh for $\Omega(\boldsymbol{\mu})$ (left) and deformation obtained with parameters $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$ (middle and right): the higher the value of μ_1, the larger the displacement entailed by $\vec{h}(\boldsymbol{\mu})$.

For the convective term $\mathbf{C}(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$, we compute the affine approximation following the double POD strategy as outlined in Sect. 3.3, by first constructing for each parameter $\boldsymbol{\mu}_i$, $i = 1, \dots, n_s$ an MDEIM basis in time with a tolerance $\varepsilon_C^{loc} = 10^{-7}$; for the case at hand, $Q_c^t = 32$ matrix bases are retained on average for each $i = 1, \dots, n_s$. Notice that compared to the test case of the previous section, a much larger number of bases are retained to properly approximate the time trajectory of $\mathbf{C}(\mathbf{V}_u\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$, due to the larger Reynolds number of the case under examination. MDEIM in parameter finally retains $Q_c = 463$ affine terms by employing a tolerance $\varepsilon_C = 5 \cdot 10^{-5}$. Regarding state reduction, the RB matrices $\mathbf{V}_u, \mathbf{V}_p, \mathbf{V}_s$ for velocity, pressure and supremizing functions, respectively, are built with POD; this latter retains $N_u = 776$, $N_p = 370$, and $N_s = 705$ basis functions, respectively.

Examples of solutions for different values of the parameters, computed with the HROM, are reported in Figure 9. On average, the NS system is solved by the HROM with a computational cost of 1.84 seconds per time step, yielding a speedup of 8.1 with respect to the FOM (GMRES with SIMPLE preconditioner). The numerical results are summarized in Table 6. A comparison between the velocity fields computed by the HROM and the FOM for a given parameter vector, as well as the resulting error between the two, are reported in Figure 10.

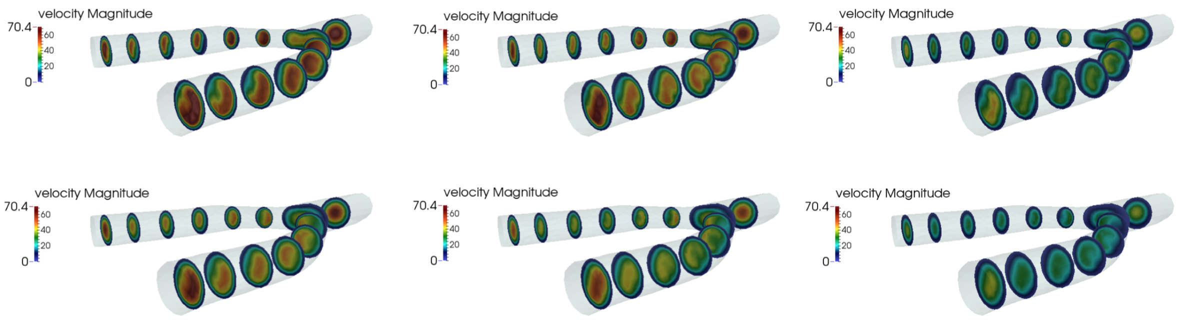


Figure 9: Slices of the velocity magnitude computed by the HROM for $\boldsymbol{\mu}^1 = (0.375, 0.975)$ (top) and $\boldsymbol{\mu}^2 = (0.225, 0.875)$ (bottom) at time $t = 0.2$, 0.3, 0.4 seconds (from left to right).

Table 6: Summary results for blood flow in bifurcation. Computational times are expressed in seconds and $t_{\text{RB}}^{\text{onl}}$ and $t_{\text{FE}}^{\text{onl}}$ refer to the average time needed for the solution of one time step with the ROM and the FOM, respectively. The computation has been carried out with the cluster Fidis at EPFL, an Intel Broadwell based cluster, composed of 408 compute nodes, each with 2 Intel Broadwell processors running at 2.6 GHz, with 14 cores each (28 cores per machine).

N_u	N_p	N_s	Q_c^t	Q_c	t_{off}	$t_{\text{RB}}^{\text{onl}}$	$t_{\text{FE}}^{\text{onl}}$	SPEEDUP
776	370	705	32	463	28579.94	1.84	15.01	8

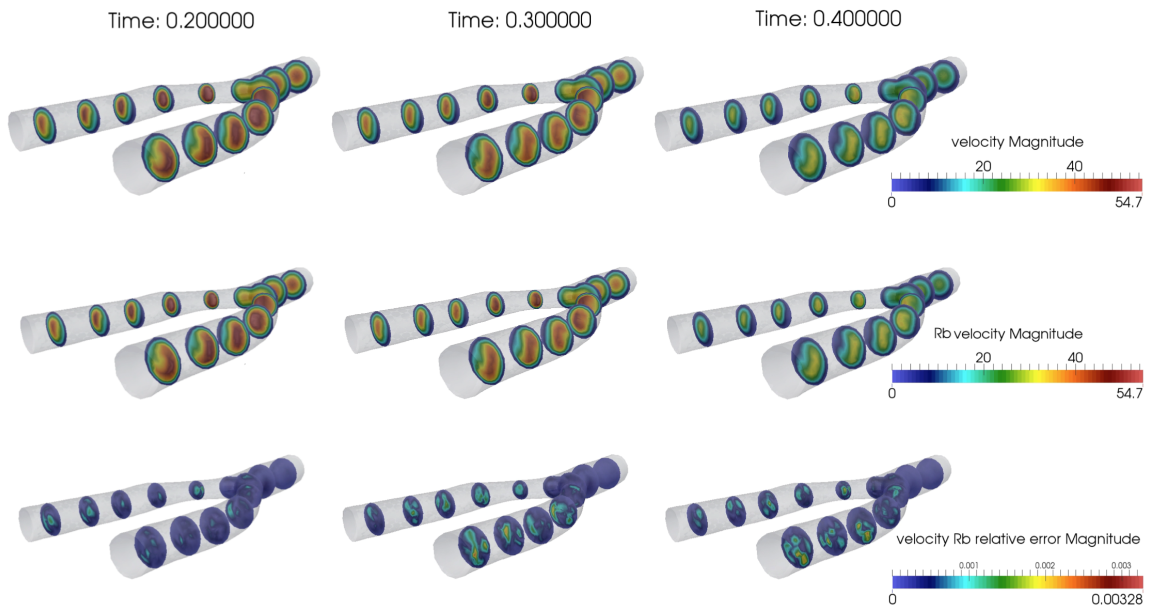


Figure 10: Slices of the velocity magnitude computed by the FOM (top) and HROM (middle) for a given parameter vector at time $t = 0.2,\ 0.3,\ 0.4$ seconds, and relative errors between FOM and HROM approximations (bottom).

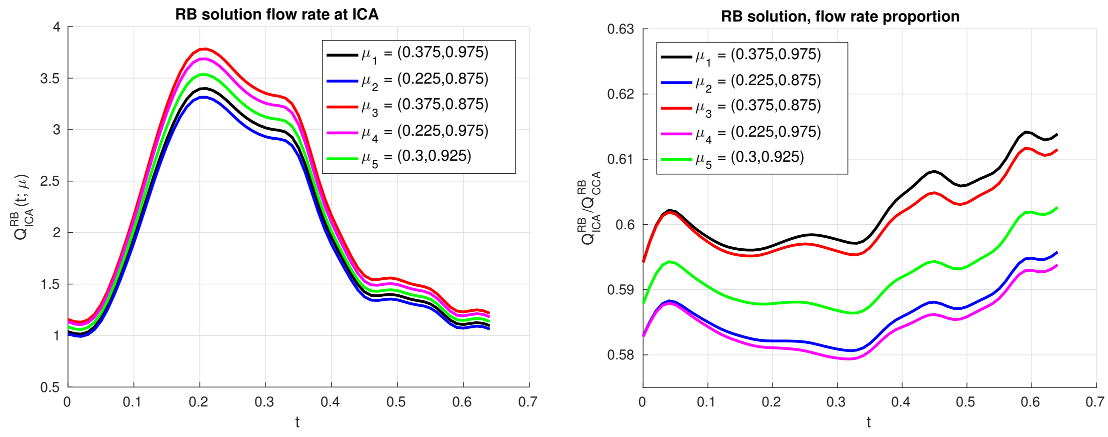


Figure 11: Flow rate $Q_{ICA}(t;\boldsymbol{\mu})$ (left) and ratio $Q_{ICA}(t;\boldsymbol{\mu})/Q_{CCA}(t;\boldsymbol{\mu})$ (right) for different values of $\boldsymbol{\mu}$.

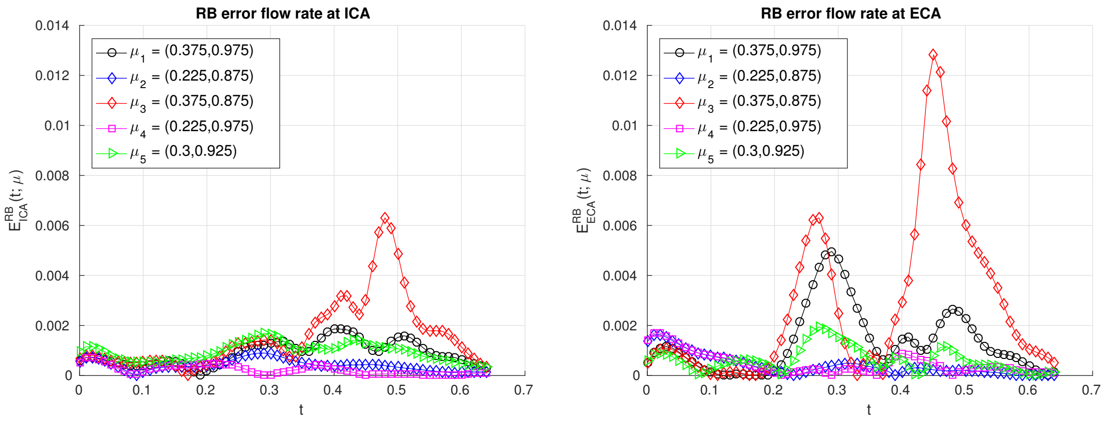


Figure 12: Relative error $E_i^{RB} = \frac{|Q_i^{RB}(t;\mu) - Q_i^{FE}(t;\mu)|}{|Q_i^{FE}(t;\mu)|}$, $i = ICA, ECA$, between FOM and ROM approximations of flow rates at the ICA (left) and ECA (right) outflows, for the values of $\boldsymbol{\mu}$ considered in Figure 11.

The velocity pattern is affected by the parameter values, also influencing the flow rate at the outlet boundaries: in Figure 11 we report the outflow rate $Q_{ICA}(t;\boldsymbol{\mu})$ at the ICA boundary as function of time, as well as the ratio $Q_{ICA}(t;\boldsymbol{\mu})/Q_{CCA}(t;\boldsymbol{\mu})$, that is, the percentage of flow rate exiting from the ICA branch. As a matter of fact, the physical parameter $\boldsymbol{\mu}_2$ mainly affects the absolute value of $Q_{ICA}(t;\boldsymbol{\mu})$, whereas the geometrical parameter $\boldsymbol{\mu}_1$ mostly affects the way blood flow is distributed between the two branches: the higher $\boldsymbol{\mu}_1$, the larger the portion of blood directed in the ICA with respect to the one entering the ECA. Relative errors between FOM and ROM approximations of flow rates at ICA and ECA outflows, are reported in Figure 12 for some parameter values; errors on the flow rate at ICA are slightly larger than those at ECA, both of them being on average lower than 1%.

As already remarked, a relevant quantity of interest when dealing with cardiovascular simulations is the wall shear stress (WSS) distribution on Γ_w, which is defined as

$$\vec{\tau}_w = (2\bar{\mu}\boldsymbol{\varepsilon}(\vec{u})\vec{n}) \cdot \vec{t} = 2\bar{\mu}(\boldsymbol{\varepsilon}(\vec{u})\vec{n} - (\boldsymbol{\varepsilon}(\vec{u})\vec{n} \cdot \vec{n})\vec{n}),$$

where $\vec{n}$ and $\vec{t}$ are the (outer) normal and tangential unit vectors on Γ_w, respectively, $\boldsymbol{\varepsilon}$ is the strain tensor defined in (4.3) and $\bar{\mu}$ is the dynamic viscosity of the fluid. In this context the WSS distribution clearly depends on the parameter $\boldsymbol{\mu}$, that is $\vec{\tau}_w = \vec{\tau}_w(\boldsymbol{\mu})$, due to the of both the solution $\vec{u}(\boldsymbol{\mu})$ and the geometry, that is $\vec{n} = \vec{n}(\boldsymbol{\mu})$ and $\vec{t} = \vec{t}(\boldsymbol{\mu})$, in its definition.

In Figure 5.8 the WSS magnitude distribution is reported for different values of the parameters and times; as expected, WSS magnitude is higher during the systolic peak and concentrated close to the bifurcation. To further investigate the phenomenon, we place three probes in different location on Γ_w (see Figure 13): P2 is located close to the outflow of the ICA, whereas P1 and P3 close to the bifurcation, at the entrance of ICA and ECA, respectively. The time dependence of the WSS magnitude at the points identifies by P1, P2 and P3 is reported for the values of $\boldsymbol{\mu}^1 = (0.375, 0.975)$ (top) and $\boldsymbol{\mu}^2 = (0.225, 0.875)$. As a matter of fact, both the geometrical and physical parameters give a large contribution to the time variability of τ_w, especially close to the bifurcation (points P1 and P3). In particular, the smaller the diameter of the ECA, the larger the WSS magnitude, as expected in practice.

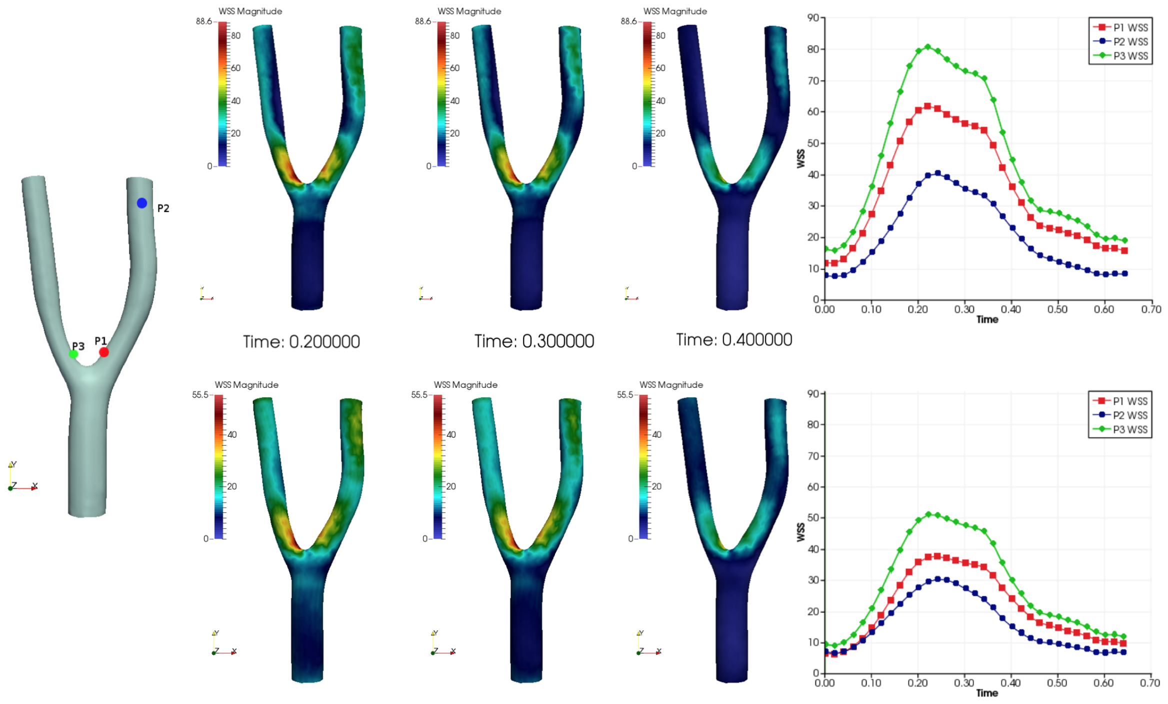


Figure 13: Wall shear stress (WSS, $[dyn \cdot cm^{-2}]$) magnitude distribution computed by the HROM for $\boldsymbol{\mu}^1 = (0.375, 0.975)$ (top) and $\boldsymbol{\mu}^2 = (0.225, 0.875)$ (bottom). From left to right: probes $P1, P2, P3$ location, visualization of WSS over the wall boundary at time $t = 0.2$, 0.3, 0.4 seconds, and evolution of WSS magnitude at the probes.

6 Conclusions

In this paper we have extended the state-of-the-art framework of RB methods for the treatment of unsteady NS equations in nonaffinely parametrized geometries. This is pursued by employing a mesh motion technique to tackle the domain deformation and a waterfall of ROMs to deal at first with the computation of the domain displacement and then with the fluid flow. In order to gain the maximum efficiency, an hyper-reduction strategy relying on the matrix version of DEIM has been employed to treat the nonaffine and nonlinear convective terms appearing in the NS equations. Moreover, a low-dimensional geometrical parametrization is devised in an almost automatic way, relying on the RB method for this goal as well. This results in an extremely efficient, purely algebraic (and less intrusive) framework capable to tackle fluid flows in complex parametrized shapes. By relying on a POD-Galerkin strategy, a suitable enrichment of the velocity space must be taken into account so that the stability of the resulting ROM is ensured; at the moment, further analysis concerning the possibility to rely on a *coarse* algebraic least squares RB method is ongoing; this latter option has been introduced and validated in [13] in the case of Stokes equations. This strategy, relying on a RB least squares method, would allow to recover automatically the stability of the resulting ROM, without entailing the cumbersome extracts usually entailed by Petrov-Galerkin RB method. Further extensions could involve the use of local reduced bases of smaller dimension (see, e.g., [41]) if compared to the global POD basis used to approximate the state solution in the proposed methodology.

A Appendix

A.1 Proper Orthogonal Decomposition

In this paper we employ POD both for the construction of RB spaces (see Sect. 3.1) and for the efficient approximation of $\boldsymbol{\mu}$-dependent arrays (see Sect. 3.3). For simplicity, here we focus on the former aspect; see, e.g., [39] and the following section for further insights on the latter.

Let us denote by $\mathbf{S} = [\mathbf{s}^{\boldsymbol{\mu}_1}| \dots |\mathbf{s}^{\boldsymbol{\mu}_{n_s}}] \in \mathbb{R}^{N_h \times n_s}$ a matrix collecting as columns n_s FE vectors $\{\mathbf{s}^{\boldsymbol{\mu}_i}\}_{i=1}^{N_s} \subset \mathbb{R}^{N_h}$ (called *snapshots*). For any prescribed dimension N, POD provides the N-dimensional subspace, spanned by the columns of $\mathbf{V} = [\boldsymbol{\xi}_1| \dots |\boldsymbol{\xi}_N] \in \mathbb{R}^{N_h \times N}$, which best approximates $\{\mathbf{s}^{\boldsymbol{\mu}_i}\}_{i=1}^{N_s}$ among all possible N-dimensional subspaces. To this goal, POD computes the singular value decomposition (SVD) $\mathbf{X}^{\frac{1}{2}}\mathbf{S} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{Z}^T$ of the matrix $\mathbf{S}$, with respect to a scalar product induced by a symmetric positive definite matrix $\mathbf{X}$, where $\mathbf{U} \in \mathbb{R}^{N_h \times N_h}$ and $\mathbf{Z} \in \mathbb{R}^{n_s \times n_s}$ denote orthogonal matrices and $\boldsymbol{\Sigma} \in \mathbb{R}^{N_h \times n_s}$ is a diagonal matrix containing the singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{n_s} \geq 0$. The matrix $\mathbf{V}$ is obtained by retaining the first N columns of $\mathbf{U}$ and represents, by construction, an $\mathbf{X}$-orthonormal basis of the best N-dimensional subspace approximating the snapshot set. In particular, we have that

$$\sum_{i=1}^{n_s} \|\mathbf{s}_i - \mathbf{V}\mathbf{V}^T\mathbf{X}\mathbf{s}_i\|_{\mathbf{X}}^2 = \min_{\mathbf{W} \in \mathcal{V}_N} \sum_{i=1}^{n_s} \|\mathbf{s}_i - \mathbf{W}\mathbf{W}^T\mathbf{X}\mathbf{s}_i\|_{\mathbf{X}}^2 = \sum_{i=N+1}^{n_s} \sigma_i^2.$$

where $\mathcal{V}_N = \{\mathbf{W} \in \mathbb{R}^{N_h \times N} : \mathbf{W}^T\mathbf{X}\mathbf{W} = \mathbf{I}_N\}$ and $\mathbf{I}_N$ is the N-dimensional identity matrix. Moreover, the discarded singular values provide an estimate of the relative error, since

$$\frac{1}{\sum_{i=1}^{n_s} \|\mathbf{s}_i\|_{\mathbf{X}}^2} \sum_{i=1}^{n_s} \|\mathbf{s}_i - \mathbf{V}\mathbf{V}^T\mathbf{X}\mathbf{s}_i\|_{\mathbf{X}}^2 = \frac{1}{\sum_{i=1}^{n_s} \sigma_i^2} \sum_{i=N+1}^{n_s} \sigma_i^2. \tag{38}$$

POD is performed by means of Algorithm 3; for a given tolerance ε_{POD}, (38) is employed to control the relative error on the approximation of the snapshots and to select N basis functions; alternatively, one could directly provide a dimension N instead of ε_{POD}.

Algorithm 3 Proper Orthogonal Decomposition (POD)

```
1: procedure POD($\mathbf{S}$, $\mathbf{X}$, $\varepsilon_{\text{POD}}$)
2:     form the correlation matrix $\mathbf{C}_{n_s} = \mathbf{S}^T\mathbf{X}\mathbf{S}$
3:     solve the eigenvalue problem $\mathbf{C}_{n_s}\boldsymbol{\psi}_i = \sigma_i^2\boldsymbol{\psi}_i$,   $i = 1, \dots, n_s$ and set $\boldsymbol{\xi}_i = \frac{1}{\sigma_i}\mathbf{S}\boldsymbol{\psi}_i$
4:     define $N$ as the minimum integer such that $\frac{\sum_{i=1}^{N}\sigma_i^2}{\sum_{i=1}^{n_s}\sigma_i^2} > 1 - \varepsilon_{\text{POD}}^2$ and $\mathbf{V} = [\boldsymbol{\xi}_1| \dots |\boldsymbol{\xi}_N]$
5: end procedure
```

A.2 Matrix DEIM

Because of its pivotal role in enhancing the efficiency of the proposed ROM workflow, here we sketch some details about the way (discrete) empirical interpolation can be used to approximate a matrix $\mathbb{K}(\tau) \colon \mathcal{T} \mapsto \mathbb{R}^{N_h \times N_h}$, where τ denotes a parameter vector and/or time. Given $\mathbb{K}(\tau) \colon \mathcal{T} \mapsto \mathbb{R}^{N_h \times N_h}$, MDEIM provides $M \ll N_h$ functions $\theta_q \colon \mathcal{T} \mapsto \mathbb{R}$ and τ-independent matrices $\mathbb{K}_q \in \mathbb{R}^{N_h \times N_h}$, $1 \leq q \leq M$, such that

Algorithm 4 Matrix Discrete empirical interpolation method (MDEIM)

```
1:  procedure MDEIM($\mathbf{S}$, $\delta_{mdeim}$)
2:      $[\boldsymbol{\phi}_1 \mid \ldots, \mid \boldsymbol{\phi}_M] = \text{POD}(\mathbf{S}, \delta_{mdeim}, \mathbf{I})$
3:      $i_m = \arg\max_{i=1,\ldots,N_q} |(\boldsymbol{\phi}_1)_i|$
4:      $\boldsymbol{\Phi} = \boldsymbol{\phi}_1, \ \mathcal{I} = \{i_1\},$
5:      for $m = 2 : M$ do
6:          $\mathbf{r} = \boldsymbol{\phi}_m - \boldsymbol{\Phi}\boldsymbol{\Phi}_{\mathcal{I}}^{-1}(\boldsymbol{\phi}_m)_{\mathcal{I}}$
7:          $i_m = \arg\max_{i=1,\ldots,N_q} |\mathbf{r}_i|$
8:          $\boldsymbol{\Phi} \leftarrow [\boldsymbol{\Phi} \ \ \boldsymbol{\phi}_m], \ \mathcal{I} \leftarrow \mathcal{I} \cup i_m$
9:      end for
10: end procedure
```

$$\mathbb{K}(\tau) \approx \mathbb{K}_m(\tau) = \sum_{q=1}^{M} \theta_q(\tau)\, \mathbb{K}_q. \tag{39}$$

To avoid misunderstadings, here we adopt a different notation from the one used in Section A.1, although some operations are indeed very similar. The offline stage of MDEIM consists of two main steps. First we express $\mathbb{K}(\tau)$ in vector format by stacking its columns, that is, we set $\mathbf{k}(\tau) = \text{vec}(\mathbb{K}(\tau)) \in \mathbb{R}^{N_h^2}$. Hence, (39) can be reformulated as: find $\{\boldsymbol{\Phi}, \boldsymbol{\theta}(\tau)\}$ such that

$$\mathbf{k}(\tau) \approx \mathbf{k}_m(\tau) = \boldsymbol{\Phi}\boldsymbol{\theta}(\tau), \tag{40}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{N_h^2 \times M}$ is a τ-independent basis and $\boldsymbol{\theta}(\tau) \in \mathbb{R}^M$ the corresponding coefficients vector. Then, we apply DEIM as in [11] to a set of snapshots $\mathbf{S} = [\text{vec}(\mathbb{K}(\tau_1)), \ldots, \text{vec}(\mathbb{K}(\tau_{n_s}))]$ in order to obtain the basis $\boldsymbol{\Phi}$ and a set of interpolation indices $\mathcal{I} \subset \{1, \cdots, N_h^2\}$. The former is computed by applying POD (for a given tolerance δ_{mdeim}) over the columns of $\mathbf{S}$, whereas the latter is iteratively selected by employing the *magic points* algorithm [32]. Both these steps are reported in Algorithm 4.

During the online phase, given a new $\tau \in \mathcal{T}$, we can compute $\mathbb{K}_m(\tau)$ as

$$\mathbb{K}_m(\tau) = \text{vec}^{-1}(\boldsymbol{\Phi}\boldsymbol{\theta}(\tau)) \quad \text{with} \quad \boldsymbol{\theta}(\tau) \text{ such that } \boldsymbol{\Phi}_{\mathcal{I}}\boldsymbol{\theta}(\tau) = \mathbb{K}_{\mathcal{I}}(\tau), \tag{41}$$

where $\boldsymbol{\Phi}_{\mathcal{I}}$ and $\mathbb{K}_{\mathcal{I}}(\tau)$ denote the matrices formed by the $\mathcal{I}$ rows of $\boldsymbol{\Phi}$ and $\mathbb{K}(\tau)$, respectively. Note that evaluating $\mathbb{K}_{\mathcal{I}}(\tau)$ online can be performed efficiently when $\mathbb{K}(\tau)$ results from a FE discretization of a PDE operator, by employing the same assembly routine used for the FOM on the *reduced mesh* associated to the selected interpolation indices $\mathcal{I}$; see, e.g., [39, 7].

Acknowledgements

The research of N. Dal Santo has been supported by the Swiss State Secretariat for Education, Research and Innovation (SERI), project No. C14.0068, in the framework of the COST action number TD1307. We acknowledge the Swiss National Supercomputing Centre (CSCS) for providing us the CPU resources under project ID s796.

References

[1] J. Baiges, R. Codina, and S. Idelsohn. Explicit reduced-order models for the stabilized finite element approximation of the incompressible Navier-Stokes equations. *Int. J. Numer. Meth. Fluids*, 72(12):1219–1243, 2013.

[2] T. J. Baker. Mesh movement and metamorphosis. *Engineering with Computers*, 18(3):188–198, 2002.

[3] F. Ballarin, E. Faggiano, S. Ippolito, A. Manzoni, A. Quarteroni, G. Rozza, and R. Scrofani. Fast simulations of patient-specific haemodynamics of coronary artery bypass grafts based on a POD-Galerkin method and a vascular shape parametrization. *J. Comput. Phys.*, 315:609–628, 2016.

[4] F. Ballarin, E. Faggiano, A. Manzoni, A. Quarteroni, G. Rozza, S. Ippolito, C. Antona, and R. Scrofani. Numerical modeling of hemodynamics scenarios of patient-specific coronary artery bypass grafts. *Biomech. Model Mechan.*, 16(4):1373–1399, 2017.

[5] F. Ballarin, A. Manzoni, A. Quarteroni, and G. Rozza. Supremizer stabilization of POD–Galerkin approximation of parametrized steady incompressible Navier–Stokes equations. *Int. J. Numer. Methods Engng.*, 102(5):1136–1161, 2015.

[6] M. Bergmann, C.-H. Bruneau, and A. Iollo. Enablers for robust POD models. *J. Comput. Phys.*, 228(2):516 – 538, 2009.

[7] D. Bonomi, A. Manzoni, and A. Quarteroni. A matrix DEIM technique for model reduction of nonlinear parametrized problems in cardiac mechanics. *Comput. Methods Appl. Mech. Engrg.*, 324:300–326, 2017.

[8] A. Caiazzo, T. Iliescu, V. John, and S. Schyschlowa. A numerical investigation of velocity–pressure reduced order models for incompressible flows. *J. Comput. Phys.*, 259:598–616, 2014.

[9] I. C. Campbell, J. Ries, S. S. Dhawan, A. A. Quyyumi, W. R. Taylor, and J. N. Oshinski. Effect of inlet velocity profiles on patient-specific computational fluid dynamics simulations of the carotid bifurcation. *J. Biomech. Engng.*, 134(5):051001, 2012.

[10] K. Carlberg, C. Farhat, J. Cortial, and D. Amsallem. The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *J. Comput. Phys.*, 242:623–647, 2013.

[11] S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM J. Sci. Comput.*, 32(5):2737–2764, 2010.

[12] C. M. Colciago, S. Deparis, and A. Quarteroni. Comparisons between reduced order models and full 3d models for fluid–structure interaction problems in haemodynamics. *J. Comput. Appl. Math.*, 265:120–138, 2014.

[13] N. Dal Santo, S. Deparis, A. Manzoni, and A. Quarteroni. An algebraic least squares reduced basis method for the solution of parametrized Stokes equations. Technical Report 21.2017, MATHICSE–EPFL, 2017.

[14] E. De Sturler and J. Liesen. Block-diagonal and constraint preconditioners for nonsymmetric indefinite linear systems. part i: Theory. *SIAM J. Sci. Comput.*, 26(5):1598–1619, 2005.

[15] S. Deparis. Reduced basis error bound computation of parameter-dependent Navier–Stokes equations by the natural norm approach. *SIAM J. Numer. Anal.*, 46(4):2039–2067, 2008.

[16] P. Díez, S. Zlotnik, and A. Huerta. Generalized parametric solutions in Stokes flow. *Comput. Methods Appl. Mech. Engrg.*, 326:223 – 240, 2017.

[17] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. Block preconditioners based on approximate commutators. *SIAM J. Sci. Comput.*, 27(5):1651–1668, 2006.

[18] H. C. Elman and V. Forstall. Numerical solution of the parameterized steady-state Navier-Stokes equations using empirical interpolation methods. *Comput. Methods Appl. Mech. Engrg.*, 317:380 – 399, 2017.

[19] H. C. Elman and D. Silvester. Fast nonsymmetric iterations and preconditioning for Navier–Stokes equations. *SIAM J. Sci. Comput.*, 17(1):33–46, 1996.

[20] H. C. Elman, D. J. Silvester, and A. J. Wathen. Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics, 2005.

[21] D. Forti and L. Dedè. Semi-implicit bdf time discretization of the Navier–Stokes equations with vms-les modeling in a high performance computing framework. *Computers & Fluids*, 117:168–182, 2015.

[22] P. Gervasio, F. Saleri, and A. Veneziani. Algebraic fractional-step schemes with spectral methods for the incompressible navier–stokes equations. *J. Comput. Phys.*, 214(1):347–365, 2006.

[23] U. Ghia, K. N. Ghia, and C. Shin. High-re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *J. Comput. Phys.*, 48(3):387–411, 1982.

[24] B. Guerciotti, C. Vergara, L. Azzimonti, L. Forzenigo, A. Buora, P. Biondetti, and M. Domanin. Computational study of the fluid-dynamics in carotids before and after endarterectomy. *J. Biomech.*, 49(1):26 – 38, 2016.

[25] M. D. Gunzburger, J. S. Peterson, and J. N. Shadid. Reducer-order modeling of time-dependent PDEs with multiple parameters in the boundary data. *Comput. Methods Appl. Mech. Engrg.*, 196:1030–1047, 2007.

[26] B. Helenbrook. Mesh deformation using the biharmonic operator. *Int. J. Numer. Methods Engrg.*, 56(7):1007–1021, 2003.

[27] K. Ito and S. Ravindran. A reduced-order method for simulation and control of fluid flows. *J. Comput. Phys.*, 143(2):403–425, July 1998.

[28] D. Kay, D. Loghin, and A. Wathen. A preconditioner for the steady-state Navier–Stokes equations. *SIAM J. Sci. Comput.*, 24(1):237–256, 2002.

[29] D. J. Knezevic, N.-C. Nguyen, and A. T. Patera. Reduced basis approximation and a posteriori error estimation for the parametrized unsteady Boussinesq equations. *Math. Mod. Meth. Appl. Sci.*, 21(07):1415–1442, 2011.

[30] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM J. Numer. Anal.*, 40(2):492–515, 2002.

[31] R. M. Lancellotti, C. Vergara, L. Valdettaro, S. Bose, and A. Quarteroni. Large eddy simulations for blood dynamics in realistic stenotic carotids. *Int. J. Numer. Methods Biomed. Engng.*, 33(11), 2017.

[32] Y. Maday, N. C. Nguyen, A. T. Patera, and S. H. Pau. A general, multipurpose interpolation procedure: the magic points. *Commun. Pur. Appl. Anal.*, 8(1):383–404, 2009.

[33] A. Manzoni. An efficient computational framework for reduced basis approximation and a posteriori error estimation of parametrized Navier-Stokes flows. *ESAIM Math. Modelling Numer. Anal.*, 48(4):1199–1226, 2014.

[34] A. Manzoni and F. Negri. Efficient reduction of PDEs defined on domains with variable shape. In P. Benner, M. Ohlberger, A. Patera, G. Rozza, and K. Urban, editors, *Model Reduction of Parametrized Systems*, volume 17, pages 183–199. Springer, Cham, 2017.

[35] A. Manzoni, A. Quarteroni, and G. Rozza. Model reduction techniques for fast blood flow simulation in parametrized geometries. *Int. J. Numer. Methods Biomed. Engng.*, 28(6-7):604–625, 2012.

[36] A. Manzoni, A. Quarteroni, and G. Rozza. Shape optimization for viscous flows by reduced basis methods and free-form deformation. *Int. J. Numer. Methods Fluids*, 70(5):646–670, 2012.

[37] D. A. May and L. Moresi. Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics. *Physics of the Earth and Planetary Interiors*, 171(1):33–47, 2008.

[38] F. Negri. *Efficient Reduction Techniques for the Simulation and Optimization of Parametrized Systems.* PhD thesis, EPFL, 2015.

[39] F. Negri, A. Manzoni, and D. Amsallem. Efficient model reduction of parametrized systems by matrix discrete empirical interpolation. *J. Comput. Phys.*, 303:431–454, 2015.

[40] F. Negri, A. Manzoni, and G. Rozza. Reduced basis approximation of parametrized optimal flow control problems for the Stokes equations. *Comput. Math. Appl.*, 69:319–336, 2015.

[41] S. Pagani, A. Manzoni, and A. Quarteroni. Numerical approximation of parametrized problems in cardiac electrophysiology by a local reduced basis method. *Comput. Meth. Appl. Mech. Engrg.*, 340:530–558, 2018.

[42] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction.* Springer, 2016.

[43] A. Quarteroni and G. Rozza. Numerical solution of parametrized Navier–Stokes equations by reduced basis methods. *Numer. Meth. Part. D. E.*, 23(4):923–948, 2007.

[44] G. Rozza, D. Huynh, and A. Manzoni. Reduced basis approximation and error bounds for Stokes flows in parametrized geometries: roles of the inf–sup stability constants. *Numer. Math.*, 125(1):115–152, 2013.

[45] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, 1993.

[46] A. Segal, M. ur Rehman, and K. Vuik. Preconditioners for incompressible Navier-Stokes solvers. *Numerical Mathematics: Theory, Methods and Applications*, 3(3):245–275, 2010.

[47] D. Sieger, M. Botsch, and S. Menzel. On shape deformation techniques for simulation-based design optimization. In S. Perotto and L. Formaggia, editors, *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, volume 5 of *SEMA SIMAI Springer Series*, pages 281–303. Springer International Publishing, Switzerland, 2015.

[48] D. Silvester, H. C. Elman, D. Kay, and A. Wathen. Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow. *J. Comput. Appl. Math.*, 128(1):261–279, 2001.

[49] C. Slager, J. Wentzel, F. Gijsen, A. Thury, A. Van der Wal, J. Schaar, and P. Serruys. The role of shear stress in the destabilization of vulnerable plaques and related therapeutic implications. *Nature Reviews Cardiology*, 2(9):456, 2005.

[50] M. L. Staten, S. J. Owen, S. M. Shontz, A. G. Salinger, and T. S. Coffey. A comparison of mesh morphing methods for 3d shape optimization. In *Proceedings of the 20th international meshing roundtable*, pages 293–311. Springer, 2011.

[51] K. Stein, T. Tezduyar, and R. Benney. Mesh moving techniques for fluid-structure interactions with large displacements. *J. Appl. Mech.*, 70(1):58–63, 2003.

[52] K. Stein, T. E. Tezduyar, and R. Benney. Automatic mesh update with the solid-extension mesh moving technique. *Comput. Methods Appl. Mech. Engrg.*, 193(21-22):2019–2032, 2004.

[53] T. Tezduyar, M. Behr, S. Mittal, and A. Johnson. Computation of unsteady incompressible flows with the stabilized finite element methods: Space-time formulations, iterative strategies and massively parallel implementations. In *New methods in transient analysis*, volume 246/AMD, pages 7–24. ASME, New York, 1992.

[54] S. Turek. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approache*, volume 6. Springer Science & Business Media, 1999.

[55] K. Veroy and A. Patera. Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: rigorous reduced-basis a posteriori error bounds. *International Journal for Numerical Method in Fluids*, 47(8-9):773–788, 2005.

[56] K. Vuik, A. Saghir, and G. Boerstoel. The Krylov accelerated SIMPLE (R) method for flow problems in industrial furnaces. *Int. J. Numer. Methods Fluids*, 33(7):1027–1040, 2000.

[57] A. Wathen and D. Silvester. Fast iterative solution of stabilised stokes systems. part i: Using simple diagonal preconditioners. *SIAM J. Numer. Anal.*, 30(3):630–649, 1993.

[58] J. Weller, E. Lombardi, M. Bergmann, and A. Iollo. Numerical methods for low-order modeling of fluid flows based on POD. *Int. J. Numer. Methods Fluids*, 63(2):249–268, 2010.

[59] G. Wittum. Multi-grid methods for Stokes and Navier-Stokes equations. *Numer. Math.*, 54(5):543–563, 1989.

[60] D. M. Wootton and D. N. Ku. Fluid mechanics of vascular systems, diseases, and thrombosis. *Annual Review of Biomedical Engineering*, 1(1):299–329, 1999.

[61] M. Yano. A space-time petrov–galerkin certified reduced basis method: Application to the Boussinesq equations. *SIAM J. Sci. Comput.*, 36(1):A232–A266, 2014.