Multi space reduced basis preconditioners for parametrized partial differential equations

THÈSE N^{O} 8553 (2018)

PRÉSENTÉE LE 30 AVRIL 2018
À LA FACULTÉ DES SCIENCES DE BASE
CHAIRE DE MODÉLISATION ET CALCUL SCIENTIFIQUE
PROGRAMME DOCTORAL EN MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Niccolò DAL SANTO

acceptée sur proposition du jury:

Prof. F. Nobile, président du jury
Prof. A. Quarteroni, Dr A. Manzoni, directeurs de thèse
Prof. K. Vuik, rapporteur
Prof. H. Elman, rapporteur
Prof. J. S. Hesthaven, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2018

Studying, and striving for truth and beauty
in general, is a sphere in which we are allowed
to be children throughout life.
— A. Einstein

Acknowledgements

I am very grateful to my advisor, Prof. Alfio Quarteroni, for the invaluable opportunity of pursuing my doctoral studies at the Chair of Modeling and Scientific Computing (CMCS) at EPFL. His tireless striving for excellence has played a major role for the achievements of this dissertation and represents a teaching I will bring with me in my whole professional carreer.

My most sincere gratitude goes to my co-advisor, Dr. Andrea Manzoni, who has been a constant support and reference during my whole period at EPFL. This thesis has largely benefited from his kindness, patience and enthusiasm, both from the scientific and human perspective; to you my very best wishes for a brilliant career as a professor and advisor. Additionally, I would like to thank MER Simone Deparis, for being an irreplaceable scientific reference and co-author, and for having shared in these years coffee breaks and sport activities, and Dr. Luca Dede’ for the fruitful scientific discussions.

I acknowledge the members of the jury, Prof. Howard Elman, Prof. Jan S. Hesthaven and Prof. Kees Vuik, for having carefully read this manuscript and for providing useful comments and feedbacks to improve it. Many thanks to Prof. Fabio Nobile, who committed himself as president of the jury.

The path towards being a PhD definitely seemed very long at the beginning of this journey and now I hardly realize how fast the last three and a half years have gone, leaving me with a multitude of achievements, adventures and memories. Above all, I have been tremendously lucky in finding on my way splendid people I could work and interact with. In particular, I wish to thank the best office mate I could wish for, Antobello, for sharing jokes, thoughts, discussions, beers and a *retrobottega*, and no matter what future will reserve us, *we’ll always have Paris*. Many thanks also to Dr. Barte for all the adventures, the brotherhood, the once-in-a-lifetime experience of sharing a fridge with you and most of all for being a constant guide as the old PhD student of our lab. I would also like to acknowledge all my collegues at CMCS, and in particular Pego Pego for running together, Claudia for Colciago’s jokes and the advices, Ste for speaking about great food and Ste da Sanbo for dreaming together of our home town, you have continuously enriched my time at EPFL in memorable and sparkling ways.

Many thanks to all the wonderful MATHICSE colleagues and friends I had the chance

to meet in Lausanne and with whom I spent amazing moments, in particular to my awesome flatmates Mattia, for the multitude of remarkable jokes in the evenings spent together, and Sènan, for being my personal French teacher and party organizer, to Isa and Giorgia for suffering together at CP and being true friends, Babak for the trekking routes and the conferences together, Regula, Vil, Ana and Thierry for beers, beachvolley matches and funny moments, Carlo for teaching me windsurfing, Teo and Fredrik for sharing great European weekends. A special mention then goes to Dr. Paolo Gatto and Dr. Robert Luce, masters of life for young PhD students, and to all my mates in Innovation Forum Lausanne, the number one EPFL student association, in particular to Cate, Bea, Konrad and Fra, more than colleagues in IFL you are friends.
Eventually, I have to thank Antonella, Ale and Kira, for warmly welcoming me every time in Cesena, and my lifetime friends in Verona, in particular Negro, Ale, Ettore, Raky, Vonny, Emma, Vale, Angy, Giuly, Ferra, Tini, no matter how the time goes by, I always feel at the first day of school when we spend time together.

My gratefulness goes to my whole family and in particular to my mum Patrizia and my aunt Isabella, who unconditionally support me in all my decisions.
In the end, to Francesca goes my most beloved gratitude, for your patience, your kindness and your love: being at your side is an overwhelming experience which fills me of joy every single day.

Lausanne, April 2018 Niccolò Dal Santo

Abstract

The multiquery solution of parametric partial differential equations (PDEs), that is, PDEs depending on a vector of parameters, is computationally challenging and appears in several engineering contexts, such as PDE-constrained optimization, uncertainty quantification or sensitivity analysis. When using the finite element (FE) method as approximation technique, an algebraic system must be solved for each instance of the parameter, leading to a critical bottleneck when we are in a multiquery context, a problem which is even more emphasized when dealing with nonlinear or time dependent PDEs. Several techniques have been proposed to deal with sequences of linear systems, such as truncated Krylov subspace recycling methods, deflated restarting techniques and approximate inverse preconditioners; however, these techniques do not satisfactorily exploit the parameter dependence. More recently, the reduced basis (RB) method, together with other reduced order modeling (ROM) techniques, emerged as an efficient tool to tackle parametrized PDEs.

In this thesis, we investigate a novel preconditioning strategy for parametrized systems which arise from the FE discretization of parametrized PDEs. Our preconditioner combines multiplicatively a RB coarse component, which is built upon the RB method, and a nonsingular fine grid preconditioner. The proposed technique hinges upon the construction of a new Multi Space Reduced Basis (MSRB) method, where a RB solver is built at each step of the chosen iterative method and trained to accurately solve the error equation. The resulting preconditioner directly exploits the parameter dependence, since it is tailored to the class of problems at hand, and significantly speeds up the solution of the parametrized linear system.

We analyze the proposed preconditioner from a theoretical standpoint, providing assumptions which lead to its well-posedness and efficiency. We apply our strategy to a broad range of problems described by parametrized PDEs: (i) elliptic problems such as advection-diffusion-reaction equations, (ii) evolution problems such as time-dependent advection-diffusion-reaction equations or linear elastodynamics equations (iii) saddle-point problems such as Stokes equations, and, finally, (iv) Navier-Stokes equations. Even though the structure of the preconditioner is similar for all these classes of problems, its fine and coarse components must be accurately chosen in order to provide the best possible results. Several comparisons are made with respect to the current state-of-the-art preconditioning and ROM techniques. Finally, we employ the proposed technique to speed up the solution of problems in the field of cardiovascular modeling.

Keywords: preconditioning techniques, finite element method, reduced basis method, parametrized partial differential equations.

Résumé

Trouver la solution multi-requête d'équations différentielles partielles (EDPs), c'est-à-dire d'EDPs dépendantes d'un vecteur de paramètres, est un problème complexe du point de vue computationnel, et qui apparaît dans de multiples contextes en ingénierie, comme par exemple en optimisation d'EDP contraintes, en quantification d'incertitude ou en analyse de sensibilité. Quand la méthode des éléments finis (FE) est utilisée comme technique d'approximation, un système algébrique doit être résolu pour chaque valeur du paramètre, menant à un goulet d'étranglement critique lors de multiples requêtes. Ce problème devient encore plus important lorsque l'on considère des EDPs non linéaires ou dépendantes du temps. Plusieurs techniques ont été proposées pour traiter des suites de systèmes linéaires, comme les méthodes de recyclage des sous-espaces de Krylov tronquées (plus connues sous le nom de *truncated Krylov subspace recycling methods*), ou les préconditionneurs inverses approximés ; cependant, ces techniques n'exploitent pas la dépendance paramétrique d'une manière satisfaisante. Plus récemment, la méthode des bases réduites (RB), aussi connu comme *reduced basis method*, ainsi que d'autres techniques de modélisation d'ordre réduit (ROM) sont apparues comme étant des outils efficaces pour aborder les EDPs paramétriques.

Dans cette thèse, nous examinons une nouvelle stratégie de préconditionnement pour les systèmes paramétriques provenant de discrétisations aux éléments finis d'EDPs paramétriques. Notre préconditionneur combine de manière multiplicative une composante RB grossière, construite grâce à la méthode RB, et un préconditionneur non singulier. La technique proposée repose sur la construction d'une nouvelle méthode des bases réduites multi-espaces (MSRB), où un solveur RB est construit à chaque itération de la méthode itérative choisie, puis entrainé pour résoudre avec précision l'équation d'erreur. Le préconditionneur qui en résulte exploite directement la dépendance paramétrique puisqu'il s'adapte à la classe de problèmes considérée, et il permet une accélération significative de la résolution du système linéaire paramétrique.

Nous analysons le préconditionneur proposé d'un point de vue théorique, en fournissant les hypothèses nécessaires pour être bien posé et efficace. Nous appliquons notre stratégie à une large gamme de problèmes décrits par des EDPs paramétriques : (i) des problèmes elliptiques comme les équations d'advection-diffusion-réaction, (ii) des problèmes évolutifs comme les équations d'advection-diffusion-réaction dépendantes du temps ou les équations linéaires élastodynamiques, (iii) des problèmes de points-selle comme les équations de Stokes, et finalement (iv) les équations de Navier-Stokes. Bien que la structure du préconditionneur soit similaire pour toutes ces classes de problèmes, ses composantes grossières et raffinées doivent être choisies avec soin pour fournir les meilleurs résultats possibles. Une comparaison est effectuée avec l'actuel meilleur préconditionneur et les techniques ROM. Nous utilisons finalement les nouveaux préconditionneurs MSRB pour accélérer la résolution de problèmes dans le domaine de modélisation cardiovasculaire.

Mots cléfs : technique de préconditionnement, méthode des éléments finis, méthode des bases réduites, équations différentielles partielles paramétriques.

Contents

Acknowledgements **v**

Abstract (English/Français) **vii**

Contents **xi**

List of Figures **xviii**

List of tables **xxi**

Introduction **1**

1 Numerical approximation of parametrized PDEs **7**
- 1.1 Parametrized PDEs . . . 7
  - 1.1.1 Steady weakly coercive problems . . . 7
  - 1.1.2 Unsteady problems . . . 11
- 1.2 Review on solution strategies for parametrized FE systems . . . 13
  - 1.2.1 A short survey on classical preconditioning techniques . . . 14
  - 1.2.2 Solution strategies for sequences of linear systems . . . 20
- 1.3 Review on RB methods for parametrized steady PDEs . . . 22
  - 1.3.1 Building the RB approximation . . . 22
  - 1.3.2 Assembling the RB problem: hyper-reduction techniques . . . 25
  - 1.3.3 Error bounds for hyper-reduced RB approximation . . . 31
  - 1.3.4 Galerkin RB methods for time-dependent problems . . . 32
- 1.4 Towards RB coarse operators for preconditioning parametrized PDEs . . . 34

2 MSRB preconditioners for parametrized PDEs **37**
- 2.1 Preconditioning the Richardson method with a RB coarse operator . . . 37
- 2.2 Multi space RB preconditioners for elliptic PDEs . . . 39
  - 2.2.1 Multi space RB preconditioners for the Richardson method . . . 39
  - 2.2.2 Nonsingularity of MSRB preconditioners . . . 41
  - 2.2.3 Convergence results for the MSRB-preconditioned Richardson method 43
  - 2.2.4 Dealing with nonaffine problems: (M)DEIM in the context of MSRB preconditioning methods . . . 45

2.2.5 Multi space RB preconditioners for flexible GMRES 47
2.3 Algorithmic procedures 50
2.3.1 Offline phase 50
2.3.2 Online phase 52
2.4 Numerical experiments for elliptic problems 53
2.4.1 Test case I: diffusion with nonaffine right hand side 54
2.4.2 Test case II: Thermal beam 66
2.5 MSRB preconditioners for parabolic PDEs 70
2.5.1 MSRB preconditioner construction 70
2.5.2 Algorithmic procedures for unsteady problems 72
2.6 Numerical results for parabolic problems 75
2.6.1 Test case III: three-dimensional heat transfer past a cylinder . . . 75

3 RB methods & MSRB preconditioners for Stokes equations 81
3.1 Parametrized Stokes equations 81
3.1.1 Finite element approximation of the Stokes equations 83
3.2 Review on RB methods for Stokes equations 85
3.2.1 RB methods for parametrized saddle-point systems 86
3.3 Algebraic LSRB method for parametrized saddle-point problems 92
3.3.1 Well-posedness of the aLSRB problem 94
3.3.2 Assembling the aLSRB problem 96
3.3.3 On the choice of $\mathbf{P}_X$ 98
3.4 Numerical experiments with aLSRB approximation 100
3.4.1 Test case setting: Stokes problem in a parametrized cylinder . . . 100
3.4.2 Numerical results 104
3.5 MSRB preconditioners for the Stokes equations 109
3.5.1 MSRB preconditioners with enriched G-RB coarse operators . . . 111
3.5.2 MSRB preconditioners with aLSRB coarse operators 112
3.5.3 Nonsingularity of the preconditioner 112
3.5.4 Algorithmic procedures 114
3.6 Numerical results 118
3.6.1 Test case I: parametrized cylinder 119
3.6.2 Test case II: parametrized carotid bifurcations 123

4 RB methods and MSRB preconditioners for NS equations 131
4.1 Parametrized Navier-Stokes equations 131
4.1.1 FE discretization and BDF time integration 133
4.1.2 Algebraic formulation 134
4.2 A ROM framework for parametrized unsteady NS equations 136
4.2.1 Basis construction: double POD strategy 138
4.2.2 ROM Stability 139
4.2.3 Enhancing efficiency by hyper-reduction 140
4.3 Sequential ROMs for deformation and fluid flows 143

4.4 Numerical results for NS-HROM 145
4.5 MSRB preconditioners for the NS equations 153
4.5.1 Preconditioner construction 154
4.5.2 Algorithmic procedures 157
4.6 Numerical results: flow past a cylinder 159
4.6.1 Periodic regime case 160
4.6.2 Parametrized case 162

5 Applications to the cardiovascular system 169
5.1 The role of mathematical simulation in cardiovascular applications 169
5.2 Solute dynamics in carotid bifurcation 170
5.2.1 The physical model and its FE discretization 171
5.2.2 Numerical results using the MSRB preconditioner 173
5.3 Blood flows in carotid bifurcations 175
5.3.1 Test case setting 175
5.3.2 Numerical results 177
5.4 Abdominal aorta aneurysm in patient specific geometry 183
5.4.1 Test case setting 183
5.4.2 Variational formulation and FE discretization 185
5.4.3 Numerical results with MSRB preconditioner 187

6 Conclusions 191

Bibliography 195

Curriculum vitæ 211

List of Figures

1.1 Examples of multigrid cycles, image taken from [Saad, 2003]. The grid cycles are characterized by lev coarsening steps and γ iterations of the classic 2-level multigrid cycle. 18

2.1 Test case I: the computational domain Ω is divided into four regions, each featuring a constant diffusivity value ν_j, $j = 1, \ldots, 4$. The coefficients $\nu_j, , j - 1, 2, 3$ are taken as problem parameters, cf. (2.48), whereas we fix the value $\nu_4 = 1$. 55

2.2 Test case I: example of solutions for different values of $\boldsymbol{\mu}$ with a null Neumann condition on $x = 1$. 56

2.3 Test case I: eigenvalues of the correlation matrix $\mathbf{C}_k$, $k = 0, \ldots, L - 1$, $N_h = 365'254$. As the iteration k increases, the decay of the eigenvalues is less steep, leading to RB spaces with more basis functions to reach the same tolerance. 58

2.4 Test case I: speed up and break-even point (BEP) as function of N_h for both fixed accuracy and fixed dimension approaches. Time are expressed in seconds. 61

2.5 Test case I: scalability and speed up as function of N_{core} for both fixed accuracy and fixed dimension approaches. Time are expressed in seconds. 62

2.6 Test case I: average relative RB residual r_{RB} and wall time to solve assemble and solve the RB-DEIM problem as function of δ_{DEIM} for different ranges of σ: the larger the range, the less efficient the RB-DEIM approximation. Time are expressed in seconds. 64

2.7 Test case I: average wall time (seconds) divided into three phases for computing the RB solution when a new instance of the parameter is considered in the online phase as function of δ_{DEIM} and for different ranges of σ. Time are expressed in seconds. 65

2.8 Solution $T(\boldsymbol{\mu})$ for different values of $\boldsymbol{\mu}$ computed with $\mathcal{P}_2$ finite elements basis functions, leading to $N_h = 2'080'389$. 67

2.9 Test case II: singular values σ_i, $i = 1, \ldots, 500$ for $\mathcal{P}_1$ and $\mathcal{P}_2$ FE basis functions computed by POD within MDEIM algorithm. 69

2.10 Test case II: $\eta_{RB}^{(k)}$ as function of k with $\mathcal{P}_1$ (top) and $\mathcal{P}_2$ FE basis functions and $\delta_{RB,k} = \delta_{RB}$, $k = 0, 1, 2, 3$ with $\delta_{RB} = 10^{-2}$ (left) and $\delta_{RB} = 10^{-3}$ (right). 69
2.11 Test case III: computational domain Ω with boundary flags, with $H = 0.41$m, $L_0 = 2.5$m. The cylinder has radius $r = 0.05$m, its center is distant 0.5m from Γ_d and 0.2m from the bottom face. 74
2.12 Test case III: computational domain Ω obtained with Gmsh [Geuzaine and Remacle, 2009]. A refinement has been applied in the (orange) area around the cylinder. 75
2.13 Test case III: example of solutions for different parameter values. 77
2.14 SVD of MDEIM computed for the construction of $\mathbf{A}(\boldsymbol{\mu})$ and $\mathbf{M}(\boldsymbol{\mu})$. . . . 78
2.15 Test case III: POD used to build the RB projection matrices $\mathbf{V}_{k,0}$ (bottom, left) and $\mathbf{V}_{k,5}$ (bottom, right); the singular values for slab $s = 0$ decrease slower than the ones for $s = 5$ when the same k for the two charts is considered. 79

3.1 Displacement for different values of $\boldsymbol{\mu}$. 101
3.2 RB and (M)DEIM functions vs δ_{RB}, $\delta_{\text{DEIM}} = 10^{-l}$, $l = 2, 3, 4, 5, 6$. 106
3.3 FE solution and G-RB and aLSRB errors for $\boldsymbol{\mu} = (2, -0.3)$. 106
3.4 FE solution and G-RB and aLSRB errors for $\boldsymbol{\mu} = (3, 0.3)$. 107
3.5 FE solution and G-RB and aLSRB errors for $\boldsymbol{\mu} = (2.7, 0.12)$. 107
3.6 Convergence of the residual in norm $\mathbf{X}_h^{-1}(\boldsymbol{\mu})$ vs the number of basis functions $N = N_u + N_p$ for the two case aLSRB (with $\mathbf{X}_h^0$ and $\mathbf{P}_{\mathbf{X}_h^0}$). Results computed with $\delta_{\text{DEIM}} = 10^{-8}$. 108
3.7 Residuals in norm $\mathbf{X}_h^{-1}(\boldsymbol{\mu})$ vs δ_{RB} for $\delta_{\text{DEIM}} = 10^{-l}$, $l = 2, 3, 4, 5, 6$. . . . 108
3.8 Computational times (seconds) vs δ_{RB} for $\delta_{\text{DEIM}} = 10^{-l}$, $l = 2, 3, 4, 5, 6$. . 108
3.9 Building time (seconds) for $\mathbf{A}_N(\boldsymbol{\mu})$ vs δ_{RB} for $\delta_{\text{DEIM}} = 10^{-l}$, $l = 2, 3, 4, 5, 6$. 109
3.10 Deformation of the domain for test case I. 120
3.11 Test case I, numerical solution for three values of $\boldsymbol{\mu}$ obtained with the MSRB preconditioning technique. 120
3.12 Test case I, iteration number and computational times vs N_h. 124
3.13 Test case II, top row: reference domain Ω^0 (left) and displacement $\mathbf{d}(\boldsymbol{\mu})$ for $\boldsymbol{\mu} = (5.0, 0.5)$; bottom row: example of region $A(\boldsymbol{\mu})$ for two values of $\boldsymbol{\mu}$.125
3.14 Test case II, numerical solution for two values of $\boldsymbol{\mu}$ obtained with the MSRB preconditioning technique. 126

4.1 Offline strategy to build NS-HROM. Red blocks specify when the collection of solution, matrix and right hand side snapshots is performed, green blocks specify when a HROM is constructed and then used to boost the offline phase. 145
4.2 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). 147
4.3 SVDs for building D-HROM and L-HROM problems. 149

4.4 SVDs for building final RB spaces (left) and final MDEIM of the linearized term (right). Notice that in the latter, according to ε_C^{loc} of the MDEIMs in time, the number of snapshots and the decay of the resulting SVD for the final MDEIM change. 149
4.5 Velocity and pressure errors entailed by the NS-HROM as function of time for two values of the parameter. 152
4.6 Computational domain employed for the flow past a cylinder, with $H = 0.41$m, $L_0 = 2.5$m. The cylinder has radius $r = 0.05$m, its center is distant 0.5m from Γ_d and 0.2m from the bottom face. 159
4.7 Clip of velocity field at the central section at time $t = 2.5$ seconds (top) and $t = 4.8$ seconds (bottom) with $\nu = 0.01$. 160
4.8 POD corresponding to the construction of the RB spaces for Mesh #1 (top row) and Mesh #2 (bottom row). We distinguish between the creation of the RB spaces for velocity (left), supremizer (centre) and pressure (right). 161
4.9 SVD for local-in-time MDEIM Mesh #1 and Mesh #2. 162
4.10 Time evolution of C_D and C_L for two different computational grids. . . . 162
4.11 Phase diagram of the drag and lift coefficients evolution for $t \in [0, 10]$. . 163
4.12 Examples of deformation according to the value of r_y. If $r_y > 0$ the section of the cylinder in the y direction is narrowed, whereas a negative value enlarges it. 164
4.13 SVD for Mesh # 2. We report the PODs employed to build the RB spaces for velocity (top-left), supremizer (top-right), pressure (bottom-left); solid lines correspond to the first time slab $s = 0$, dash lines to the second time slab $s = 1$. Singular values (SV) for the final MDEIM for the two time slabs $s = 0, 1$ are in the bottom-right corner. A similar decay of the SV is observed for the two time slabs in all the cases. 165
4.14 Evolution of drag and lift coefficients for different values of the parameters computed during the online phase. 165
4.15 Clip of the domain at the central section of the channel: zoom next to the cylinder of velocity streamlines and pressure field for different values of parameters and times. 167

5.1 Inlet velocity profile with mesh and velocity field for the dynamics of a solute. 173
5.2 Sherwood number distribution for values of the parameter vector. 173
5.3 Scalability and speedup as function of the number of cores N_{core} for $\mathcal{P}_1$ and $\mathcal{P}_2$ FE basis functions and fixed accuracy and fixed dimension approach.175
5.4 Reference domain Ω_f; common carotid artery (CCA), internal carotid artery (ICA) and external carotid artery (ECA). 176
5.5 Region A where the stress $\vec{h}(\boldsymbol{\mu})$ is applied (left) and reference flow rate $Q^0_{\text{CCA}}(t)$ (right). 178

5.6 Starting mesh for Ω_f (left) and deformation entailed by parameter instances $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$ (middle and right): the higher the value of μ_1, the larger the displacement entailed by $\vec{h}(\boldsymbol{\mu})$. 178
5.7 Slices of velocity magnitude for $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$ at different times. 180
5.8 WSS [dyn·cm^{-2}] magnitude distribution at different times for $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$. . . 181
5.9 Flow rate $Q_{\mathrm{ICA}}(t;\boldsymbol{\mu})$ at the ICA (left) and ratio $\frac{Q_{\mathrm{ICA}}(t;\boldsymbol{\mu})}{Q_{\mathrm{CCA}}(t;\boldsymbol{\mu})}$ (right) for different values of $\boldsymbol{\mu}$. 182
5.10 WSS [dyn·cm^{-2}] magnitude distribution at different times for $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$. . . 182
5.11 Location of probes (top left) and time evolution of the distribution of the magnitude of the WSS [dyn·cm^{-2}] for different parameter values. 182
5.12 Domain Ω, boundary flags and computational domain. 185
5.13 Displacement for $\boldsymbol{\mu} = (1.85 \cdot 10^6, 1.0625, 1.05 \cdot 10^5)$ at different times. . . . 189
5.14 Displacement for $\boldsymbol{\mu} = (1.05 \cdot 10^6, 1.2, 7 \cdot 10^4)$ at different times. 189
5.15 Time evolution of magnitude displacement for different parameter values in locations P1, P2, P3. 190

List of Tables

2.1 Test case I: Richardson method results with $\mathcal{P}_1$ FE: $\delta_{RB,k} = 0.001$, $k = 0, \dots 2$. We report the dimension of the RB spaces N_k and the average accuracy $\xi_{RB}^{(k)}$ obtained online for $k = 0, \dots 2$. 57

2.2 Test case I: Richardson method results with $\mathcal{P}_1$ FE: $\delta_{RB,k} = 0.1$, $k = 0, \dots 8$. We report the dimension of the RB spaces N_k and the average accuracy $\xi_{RB}^{(k)}$ obtained online for $k = 0, \dots 8$. 57

2.3 Test case I: grid analysis results for FGMRES method with fixed accuracy approach, $L = 3$, $\delta_{RB,k} = 0.001$, $\forall k$, ~ 3800 dofs per CPU. Times are in seconds. 60

2.4 Test case I: grid analysis results for FGMRES method with fixed dimension approach, $N_k = 100$ $\forall k$, ~ 3800 dofs per CPU. Time are expressed in seconds. 60

2.5 Test case I: scalability analysis results for FGMRES method with fixed accuracy approach, $\delta_{RB,k} = 10^{-3}$ $\forall k$. Time are expressed in seconds. . . . 61

2.6 Test case I: scalability analysis results for FGMRES method with fixed dimension approach, $N_k = 150$ $\forall k$. $L = 8$ RB coarse operators are produced by Algorithm 5 in all the cases. Time are expressed in seconds. 62

2.7 Test case I: RB-DEIM results with $\delta_{\text{DEIM}} = 10^{-7}$ as function of the range of $\sigma \in [\sigma_{\min}, 0.5]$. The value of $\sigma_{\min}$ significantly impact on the results obtained by the RB-DEIM, both in terms of accuracy and efficiency, due to the number N of RB functions and number Q_f DEIM basis functions. 66

2.8 MSRB results with fixed dimension approach, with $N_k = N$, $k = 0, 1, \dots, L-1$ as function of the range of $\sigma \in [\sigma_{\min}, 0.5]$. The efficiency is not significantly impacted by the range of σ, since the MSRB-preconditioning method does not rely on any affine approximation of the right hand side, avoiding the huge overhead caused by DEIM. Time are expressed in seconds. 66

2.9 Test case II: FGMRES results with MSRB preconditioner for $\mathcal{P}_1$ FE basis functions, fixed accuracy approach with $\delta_{RB,k} = 10^{-2}$ and $L = 4$ varying δ_{MDEIM}. 70

2.10 Test case II: FGMRES results with MSRB preconditioner for $\mathcal{P}_1$ FE basis functions, fixed accuracy approach with $\delta_{RB,k} = 10^{-3}$ and $L = 4$ varying δ_{MDEIM}. 70

2.11 Test case II: FGMRES results with MSRB preconditioner for $\mathcal{P}_2$ FE basis functions, fixed accuracy approach with $\delta_{RB,k} = 10^{-2}$ and $L = 4$ varying δ_{MDEIM}. 70
2.12 Test case II: FGMRES results with MSRB preconditioner for $\mathcal{P}_2$ FE basis functions, fixed accuracy approach with $\delta_{RB,k} = 10^{-3}$ and $L = 4$ varying δ_{MDEIM}. 70
2.13 Test case III: number of RB coarse operators L_s computed by Algorithm 8 to reach a tolerance $\varepsilon_r = 10^{-9}$. They are the same for all the values of N_{core}. 78
2.14 Test case III: results with MSRB preconditioner with fixed dimension approach ($N = 15$) and $\mathcal{S} = 10$ time slabs. Computational times are expressed in seconds. 79

3.1 Computational time (seconds) to build (M)DEIM basis with $\delta_{\mathrm{DEIM}} = 10^{-6}$.105
3.2 Computational time (seconds) to build RB approximation with $\delta_{RB} = 10^{-6}$.105
3.3 Computational time (seconds) required by the RB methods to compute a solution satisfying a target accuracy. 109
3.4 Test case I, MDEIM offline results, $\delta_{\mathrm{MDEIM}} = 10^{-6}$. 121
3.5 Test case I, fixed accuracy with **GRB**, $L = 4$, $\delta_{RB,k} \approx 5.6 \cdot 10^{-3}\,\forall k$. . . . 122
3.6 Test case I, fixed accuracy with **aLSRB-$\mathbf{X_h^0}$**, $L = 4$, $\delta_{RB,k} \approx 5.6 \cdot 10^{-3}\,\forall k$. 123
3.7 Test case I, fixed accuracy with **aLSRB-$\mathbf{P_{X_h^0}}$**, $L = 4$, $\delta_{RB,k} \approx 5.6 \cdot 10^{-3}\,\forall k$. 123
3.8 Test case I, fixed dimension with **GRB**, $N_k^u = N_k^p = N_k^s = 10\,\forall k$. 123
3.9 Test case I, fixed dimension with **aLSRB-$\mathbf{X_h^0}$**, $N_k^u = N_k^p = 10\,\forall k$. 123
3.10 Test case I, fixed dimension with **aLSRB-$\mathbf{P_{X_h^0}}$**, $N_k^u = N_k^p = 10\,\forall k$. 123
3.11 Test case II, DEIM and MDEIM number of affine basis functions computed during the offline phase as function of the tolerances δ_{DEIM} and δ_{MDEIM} (always chosen with the same value). The number of affine components affects the duration of the offline phase of a time t_{affine}, which in the **aLSRB-$\mathbf{P_{X_h^0}}$** solver case depends quadratically on the number of affine terms. 127
3.12 Test case II, results with **aLSRB-$\mathbf{P_{X_h^0}}$** solver, basis computed with tolerance $\delta_{RB} = 10^{-9}$ leading to $N_u = 327$ and $N_p = 111$ basis functions. The accuracy of the method largely depends on the system approximation carried out with DEIM and MDEIM. 128
3.13 Test case II, MSRB preconditioner results with FGMRES with a final tolerance $\varepsilon_r = 10^{-5}$. The RB spaces are built with the fixed dimension approach, with $N_k^u = N_k^p = 50$, $\forall k$, and **aLSRB-$\mathbf{P_{X_h^0}}$** coarse operators. The computation is carried out with 360 cores, such that ~ 8890 dofs per core. 129

4.1 D-HROM: POD for state reduction and DEIM have been run with $\varepsilon_{\mathrm{POD}} = \delta_{\mathrm{DEIM}} = 10^{-7}$. Computational times are expressed in seconds. 148

4.2 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. 148
4.3 Chosen settings for numerical experiments of NS-HROM. 149
4.4 Results of the offline phase for settings defined in Table 4.3. computational times are reported in seconds. 150
4.5 Results averaged on 50 instances of the parameter considered online. Times are reported in seconds. 151
4.6 Summary results for periodic regime. The problem is solved online on the interval $(2, 10]$. Computational times are expressed in seconds and refer to the average time needed for the solution of one time step. 162
4.7 Number of affine terms (MDEIM decompositions) for the two meshes and the two time slabs. The value of Q_a sums the number of affine components of $\mathbf{D}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$ and $\mathbf{M}^u(\boldsymbol{\mu})$, Q_c^t is the average number of affine terms for the local-in-time MDEIMs and Q_c the number of affine terms for the final MDEIM. 166
4.8 Summary results for parametrized case. The problem is solved online on the interval $(0, 1]$ on instances of the parameter different from the training set. Computational times are expressed in seconds and refer to the average time needed for the solution of one time step. 166

5.1 results for FGMRES obtained with MSRB preconditioner built a fixed accuracy approach, $\delta_{RB,k} = 0.001$, $k =, 1, 2$, $n_s = 300$. Time are expressed in seconds. 174
5.2 results for FGMRES obtained with MSRB preconditioner built a fixed dimension approach, $\delta_{RB,k} = 0.001$, $k =, 1, 2$, $n_s = 300$. Time are expressed in seconds. 174
5.3 Summary results for blood flow in bifurcation. Computational times are expressed in seconds and $t^{\text{onl}}_{\text{MSRB}}$ and t_{SIMPLE} refer to the average time needed for the solution of one time step. 179
5.4 Results with MSRB preconditioner with $(N_k = 25)$, $\mathcal{S} = 4$ time slabs and $\mathbf{P} = \mathbf{P}_{\text{AS}\,2}$. Computational times are expressed in seconds and the online time refer to the solution of one time step. 188
5.5 Results with MSRB preconditioner with $(N_k = 25)$, $\mathcal{S} = 4$ time slabs and $\mathbf{P} = \mathbf{P}_{\text{AS}\,1}$. Computational times are expressed in seconds and the online time refer to the solution of one time step. 188

Introduction

This thesis deals with the development, analysis and application of innovative and reliable numerical methods for the efficient solution of large-scale parametrized partial differential equations (PDEs), that is, PDEs depending on a parameters vector. Parameters may encode physical and/or geometrical properties of the system and enter into play in many different ways by influencing, for instance, the model coefficients, the boundary data or the definition of the geometrical domain. Parametrized PDEs are encountered in a broad range of phenomena in applied sciences and engineering simulations, where relevant applications account, just to mention few prominent cases, for sensitivity analysis, uncertainty quantification, parameter estimation or PDE-constrained optimization. These classes of problems are in general referred to as *multiquery* problems, for which we are interested in computing their solution for many different scenarios.

In general, the solution of a PDE is almost never computable in closed form, whence the need of using a *high-fidelity* discretization technique, such as the finite element (FE) method, to determine an approximate solution which is close to the exact one up to a controllable discretization error, and is computed by solving a (non)linear algebraic system. Depending on the application at hand and on the desired accuracy, the dimension of such algebraic system can range from few thousands to tens, or even hundreds, millions for the most demanding applications. Even if in recent years computing hardware capabilities have significantly improved, solving such a large algebraic system still represents, in general, a demanding task, calling for properly designed numerical algorithms. A further difficulty is added when a parameter-dependent PDE is taken into account, since each new parameter instance would require, in principle, the solution of such algebraic system from scratch.

To face such a huge and general problem, in this thesis we propose and analyze a new preconditioning technique aimed at efficiently dealing with large-scale parameter-dependent problems arising from the FE discretization of PDEs. The preconditioning strategy we envision exploits the intrinsic parameter dependence appearing in a multiquery context, is theoretically analyzed and applied to a wide range of problems, from linear elliptic to nonlinear unsteady PDEs.

State of the art

Parametrized PDE problems appear in a variety of contexts, and a broad family of methods has been developed to tackle their solution. We recall in particular the reduced basis (RB) method, [Quarteroni et al., 2016a, Hesthaven et al., 2016], which in the last decade emerged as one of the most widespread reduced order modeling (ROM) technique for solving parameter-dependent PDEs, and represents one of the main ingredients of this work. The central idea of the RB method lies in approximating the solution corresponding to a parameter as a linear combination of solutions of the same PDE for preselected parameter instances; this is pursued, in practice, through a RB low-rank solver which is built from the FE problem by (Petrov) Galerkin projection. Such a method has been successfully used for a wide range of applications; its performance (both in terms of accuracy and efficiency) is largely affected by the nature of the parameter-dependence of the problem at hand. This is the case, for instance, of problems involving a parameter space with very large dimension, for which an extremely large number of RB functions may be needed to compute an accurate approximation, time-dependent PDEs defined over a long-time horizon or problems featuring different dynamics across the parameter space. An additional issue has to be overcome when considering nonaffine and nonlinear PDEs, where the classic RB method would still rely, in principle, on the dimension of the high fidelity problem to assemble the RB low-rank solver. To make RB methods efficient also in these latter cases, additional hyper-reduction techniques, such as the Empirical Interpolation Method (EIM), [Barrault et al., 2004], or its discrete variant DEIM, [Chaturantabut and Sorensen, 2010], are required to reduce the computational effort in the assembling of the RB problem, however entailing heavy additional costs and introducing further error sources in the solution due to the system approximation.

On the other hand, parametrized linear systems built from FE discretization of PDEs can be cast in the wider class of sequences of linear systems, for which the solutions of a collection of linear systems arising from the same problem are sought. Other examples appear, for instance, when we consider time-dependent problems discretized by means of a time advancing scheme, restarted algorithms (e.g. restarted GMRES, [Saad and Schultz, 1986, Van der Vorst and Vuik, 1994]) or flexibly preconditioned iterative solvers, as the case of flexible GMRES, [Saad, 1993], if the preconditioning step is carried out by inner iterations. In the last decades, several numerical techniques have been proposed to speed up their solution and, among the others, we mention Krylov subspace recycling, deflated and augmented methods, [Simoncini and Szyld, 2007, Parks et al., 2006, Morgan, 2005] and approximate inverse preconditioners, [Benzi and Bertaccini, 2003, Bertaccini and Durastante, 2016]. These techniques succeed in accelerating the solution of the sequence of linear systems, however they do not explicitly take advantage of the parametric dependence of the PDE, possibly showing different behaviors across the parameter space. This motivates the need to exploit ROM techniques (and in this work specifically the RB method) to speed up the solution of the preconditioned high fidelity system by directly exploiting the parametric dependence of the PDE and potentially allowing to obtain uniform performances across the parameter space.

Thesis contributions

In this thesis we propose, analyze and apply a new preconditioning strategy which exploits a RB low-rank solver as coarse operator combined with a classical fine grid operator, in a two-level preconditioning fashion. To this aim, we build a sequence of iteration-dependent RB low-rank solvers, each one tailored to provide an accurate solution of the error equation arising at iteration k of the iterative method. We have called the resulting iteration-dependent operator *multi space reduced basis* (MSRB) preconditioner. The proposed approach has been initially developed for affinely parametrized linear elliptic problems and is shown to compute the solution of the parametrized FE linear system in extremely competitive iteration count and computational time, if compared with state-of-the-art preconditioning and Krylov subspace recycling methods.

Next, several extensions have been considered: at first nonaffine linear elliptic and parabolic PDEs are taken into account, for which MSRB preconditioners are shown to overcome, both theoretically and computationally, the bottleneck originating from the nonaffine parameter dependence, by relying (at most) on a coarse and easily computable approximate affine approximation. We then turn our attention to computational fluid dynamics: we propose a MSRB preconditioning strategy for linear parametrized saddle-point problems which is used for the efficient solution of steady parametrized Stokes equations. Finally, the unsteady parametrized Navier-Stokes (NS) equations in parameter-dependent domains are considered, for which the MSRB preconditioners are extended to deal with nonlinear problems. Furthermore, the technique can be easily applied to tackle other different nonlinear unsteady parametrized PDEs. The MSRB preconditioning method is applied to FE problem of engineering interest featuring a large dimension, up to millions of degrees of freedom, with emphasis on (but not strictly limited to) cardiovascular applications.

The MSRB preconditioning strategy represents the main contribution presented in this work, however other innovative techniques related to the standard RB method have been explored and devised towards the construction of the aforementioned methodology. More specifically, we have developed a new Petrov-Galerkin method for linear saddle-point problems, which represents a generalization of the least squares RB (LSRB) method initially proposed in [Abdulle and Budáč, 2015]. The main idea of our new approach lies in suitably modifying, at the algebraic level, the matrix used for the creation of the RB test space, by substituting it with a more cheaply computable (but spectrally equivalent) surrogate. The resulting formulation, which is referred to as algebraic least squares RB (aLSRB) method, is provides a well-posed RB formulation and is tested on problems of interest.

Secondly, we extend the state-of-the-art framework of RB methods for the treatment of the 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 to treat the nonaffine and nonlinear convective term appearing in the NS equations is also devised, and applied for the first time to complex three-dimensional flows.

Thesis outline

The first chapter of the thesis introduces the classes of parametrized PDE problems which are dealt with; in particular, their differential and weak formulation is reported, together with the FE discretization and the techniques which are considered as state-of-the-art for the solution of the resulting FE linear systems. Furthermore, as one of the main ingredients of the methodologies proposed, the RB method for linear (un)steady PDEs is briefly outlined.

The MSRB preconditioning framework is developed in Chapters 2, 3 and 4 by gradually increasing the complexity of the addressed problem. In particular, Chapter 2 sets the foundations of the method by taking into account linear second-order elliptic and parabolic problems (namely, advection-diffusion-reaction equations in both steady and unsteady parametrized cases). In Chapter 3, at first the aLSRB method for Stokes equations is presented and used as RB coarse operator in the MSRB preconditioner for tackling saddle-point problems. Similarly, an alternative option for building the MSRB preconditioner by exploiting an *enriched velocity* Galerkin formulation is developed. Chapter 4 is devoted instead to the unsteady parametrized Navier-Stokes equations: the reduction strategy for the NS equations in deformed domains is initially presented and then exploited in the MSRB preconditioning framework. In Chapter 5, some cardiovascular applications involving arterial tissue dynamics in abdominal aortic aneurysms and solute dynamics and blood flow in parametrized carotid bifurcations are presented. Finally, several conclusions and some areas of future work are discussed in Chapter 6.

This thesis contains results which have already been published (or accepted for publication) in journal articles or contained on papers which are currently submitted. More specifically, Chapter 2 is based upon results contained in [Dal Santo et al., 2018a], which is the first work presenting the MSRB preconditioning framework. Further details and numerical results (not included in this thesis) can also be found in [Dal Santo et al., 2017a]. Chapter 3 is based upon [Dal Santo et al., 2017b] for the construction and analysis of the aLSRB method for the parametrized Stokes equations and upon [Dal Santo et al., 2018b] for MSRB preconditioners for linear saddle-point problems; all these reports are available as submitted pre-prints. Finally, the results concerning solute dynamics in Chapter 5 are already reported in [Dal Santo et al., 2018a].

Tools of the trade

The numerical experiments presented in this thesis have been obtained by employing the parallel FE library *LifeV*[1], designed to tackle large-scale problems in a high performance computing (HPC) environment; here a (MS)RB module has been developed by the author for parametrized PDEs. *LifeV* is an open-source C++ library distributed under LGPL license which takes advantage of the MPI-based linear algebra structures of `Trilinos` [Heroux et al., 2005]; in particular, the `Ifpack` [Sala and Heroux, 2005] and `ML` [Gee et al., 2006] packages are extensively used and referred to in this dissertation.

All the numerical simulations have been run on the Swiss National Supercomputing Center[2] (CSCS), which the author gratefully acknowledges for providing the computing hours under the project IDs s635 and s796. Here, we took advantage of the Piz Daint cluster, which is a hybrid Cray XC40/XC50 system; in particular we employed the multicore XC40 computing nodes, each with two Intel® Xeon® (E5-2695 v4 @ 2.10GHz,2 x 18 cores, 64/128 GB RAM).

---

[1] www.lifev.org

[2] www.cscs.ch

1 Numerical approximation of parametrized PDEs

In this chapter we introduce the classes of parametrized problems we will deal with and present a survey on the state-of-the-art methods for their solution. More specifically, we first consider parametrized steady and unsteady linear, second-order PDEs; we set the hypotheses which ensure their well-posedness and derive their algebraic counterpart when numerical methods for spatial (and possibly time) discretization are employed. In the second part, we discuss the role of preconditioning for the iterative solution of linear systems arising from the numerical discretization of the aforementioned PDE problems. We will shortly review some state-of-the-art techniques for the cases of interest; in particular we refer to domain decomposition (DD) and multilevel methods for elliptic and parabolic PDEs and block preconditioners for saddle-point problems. Preconditioning techniques suitable for sequences of linear systems, such as Krylov subspace recycling methods and approximate inverse preconditioners, will also be accounted for. In the last part, we present the RB method for parametrized PDEs, addressing its construction and discussing its main advantages and limitations.

1.1 Parametrized PDEs

In the following, we recall the variational formulation of steady and unsteady linear second-order parametrized problems, their finite element approximation and the algebraic structure of the resulting parametrized FE systems.

1.1.1 Steady weakly coercive problems

Variational formulation

Let us consider a parameter space $\mathcal{D} \subset \mathbb{R}^p$, $p \geq 1$, and denote by $\boldsymbol{\mu} \in \mathcal{D}$ a parameter vector encoding physical and/or geometrical properties of the problem. Furthermore, let us introduce an open and bounded domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, and denote by $\partial\Omega$

its boundary. Let us consider the Hilbert space $X = X(\Omega)$ and its dual space X' and the scalar product $(\cdot,\cdot)_X$ inducing the norm $\|\cdot\|_X$. In this thesis we will consider in general $\boldsymbol{\mu}$-dependent domains, that is $\Omega = \Omega(\boldsymbol{\mu})$, with the consequent dependence on $\boldsymbol{\mu}$ of $X = X(\Omega(\boldsymbol{\mu}))$, however we omit it throughout the first chapter for the sake of notation. We introduce the parameter-dependent bilinear form $a(\cdot,\cdot;\boldsymbol{\mu}) : X \times X \to \mathbb{R}$ and the linear functional $f(\cdot;\boldsymbol{\mu}) : X \to \mathbb{R}$ and consider the following parametrized variational problem: for any $\boldsymbol{\mu} \in \mathcal{D}$, find $y(\boldsymbol{\mu}) = y(\boldsymbol{\mu}) \in X$ such that

$$a(y(\boldsymbol{\mu}), w; \boldsymbol{\mu}) = f(w; \boldsymbol{\mu}) \qquad \forall w \in X. \tag{1.1}$$

We further assume that for any $\boldsymbol{\mu} \in \mathcal{D}$, $a(\cdot,\cdot;\boldsymbol{\mu})$ is a continuous over $X \times X$ and $f(\cdot;\boldsymbol{\mu})$ continuous over X, that is, for any $\boldsymbol{\mu} \in \mathcal{D}$, there exist two positive $\boldsymbol{\mu}$-dependent factors $\gamma_1(\boldsymbol{\mu})$ and $\gamma_2(\boldsymbol{\mu})$ such that

$$|a(y, w; \boldsymbol{\mu})| \le \gamma_1(\boldsymbol{\mu}) \|y\|_X \|w\|_X \qquad |f(w; \boldsymbol{\mu})| \le \gamma_2(\boldsymbol{\mu}) \|w\|_X, \qquad \forall y, w \in X, \tag{1.2}$$

Moreover we assume the bilinear form $a(\cdot,\cdot;\boldsymbol{\mu})$ to be weakly coercive (or inf-sup stable), that is, for any $\boldsymbol{\mu} \in \mathcal{D}$ there exists a positive factor $\beta(\boldsymbol{\mu})$ such that

$$\inf_{y \in X} \sup_{w \in X} \frac{a(y, w; \boldsymbol{\mu})}{\|v\|_X \|w\|_X} \ge \beta(\boldsymbol{\mu}). \tag{1.3}$$

Under these hypotheses the Nečas theorem guarantees the well-posedness of problem (1.1), see [Necas, 1967, Boffi et al., 2013]. A special case is the one of strongly coercive problems, for which there exists $\alpha(\boldsymbol{\mu}) > 0$ such that

$$a(y, y; \boldsymbol{\mu}) \ge \alpha(\boldsymbol{\mu}) \|y\|_X^2 \qquad \forall y \in X. \tag{1.4}$$

When such assumption is met, the Lax-Milgram lemma guarantees the existence and uniqueness of the solution of the variational problem for any $\boldsymbol{\mu} \in \mathcal{D}$ (see e.g. [Salsa, 2016, Quarteroni and Valli, 2008]).

Finite element discretization

Solving problem (1.1) calls into play suitable numerical approximation techniques, here called *high fidelity* (or *full order*) approximations, providing a discretized solution which is close to the exact solution up to a (controllable) discretization error. Noteworthy examples are the finite element (FE) method [Ciarlet, 2002, Brenner and Scott, 2007, Quarteroni, 2014], spectral methods [Canuto et al., 2012] or the finite volume method [LeVeque, 2002, Wesseling, 2009]. All these methods are built upon the use of a finite dimensional space $X_h \subset X$, with $dim(X_h) = N_h$, and require to find an approximate solution $y_h(\boldsymbol{\mu})$

to (1.1) by solving the following problem: given $\boldsymbol{\mu} \in \mathcal{D}$, find $y_h(\boldsymbol{\mu}) \in X_h$ such that

$$a(y_h(\boldsymbol{\mu}), w_h; \boldsymbol{\mu}) = f(w_h; \boldsymbol{\mu}) \qquad \forall w_h \in X_h. \tag{1.5}$$

If for any $\boldsymbol{\mu} \in \mathcal{D}$, the bilinear form $a(\cdot, \cdot; \boldsymbol{\mu})$ and the linear functional $f(\cdot; \boldsymbol{\mu})$ are continuous on $X_h \times X_h$ and X_h, respectively, and $a(\cdot, \cdot; \boldsymbol{\mu})$ satisfies the *inf-sup* condition on $X_h \times X_h$, that is, there exist $\beta_h^{min} > 0$ such that

$$\beta_h(\boldsymbol{\mu}) = \inf_{y \in X_h} \sup_{w \in X_h} \frac{a(y, w; \boldsymbol{\mu})}{\|y\|_{X_h} \|w\|_{X_h}} \geq \beta_h^{min}, \tag{1.6}$$

then the well-posedness of problem (1.5) is guaranteed by the Babuška theorem (see [Babuška, 1971]).

Even if the methods proposed in this thesis are applicable to any discretization method which relies on Galerkin projection, we will specifically consider the FE case, for which problem (1.5) is equivalent to the solution of the linear system

$$\mathbf{A}(\boldsymbol{\mu})\mathbf{y}(\boldsymbol{\mu}) = \mathbf{f}(\boldsymbol{\mu}), \tag{1.7}$$

where $\mathbf{y}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ is the vector representations of the solution $y_h(\boldsymbol{\mu}) \in X_h$ over a Lagrangian basis $\{\phi_i^x\}_{i=1}^{N_h}$ of X_h, that is $X_h = span\{\phi_i^x,\, i = 1, \dots, N_h\}$. Indeed, given a set of nodes $\{\vec{x}_i\}_{i=1}^{N_h} \subset \overline{\Omega}$, we have that, for any $v_h \in X_h$, $v_h(\vec{x}_i) = \mathbf{v}_i$, $i = 1, \dots, N_h$. Similarly, we have that for the stiffness matrix $\mathbf{A}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ and the right hand side vector $\mathbf{f}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$

$$\left(\mathbf{A}(\boldsymbol{\mu})\right)_{ij} = a(\phi_j^x, \phi_i^x; \boldsymbol{\mu}) \qquad \left(\mathbf{f}(\boldsymbol{\mu})\right)_i = f(\phi_i^x; \boldsymbol{\mu}), \qquad \forall i, j = 1, \dots, N_h. \tag{1.8}$$

We also introduce the matrix $\mathbf{X}_h \in \mathbb{R}^{N_h \times N_h}$, such that

$$\left(\mathbf{X}_h\right)_{ij} = (\phi_j^x, \phi_i^x)_X, \qquad \forall i, j = 1, \dots, N_h; \tag{1.9}$$

$\mathbf{X}_h$ is the FE matrix encoding at the FE level the scalar product $(\cdot, \cdot)_X$. We then define the scalar product on vectors $(\mathbf{x}, \mathbf{y})_{\mathbf{X}_h} = (\mathbf{X}_h \mathbf{x}, \mathbf{y})_2$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{N_h}$ and the associated norm $\|\mathbf{x}\|_{\mathbf{X}_h}^2 = (\mathbf{x}, \mathbf{x})_{\mathbf{X}_h}$. Two classes of problem under the form (1.1) will be discussed in details in this thesis: elliptic advection-diffusion equation in Chapter 2 and linear saddle-point PDEs in Chapter 3.

Second order advection-diffusion equations describe a wide range of physical processes, in particular we consider the following parametrized advection-diffusion problem

$$\begin{cases} -\nabla \cdot (\mathcal{K}(\boldsymbol{\mu}) \nabla u(\boldsymbol{\mu})) + \vec{b}(\boldsymbol{\mu}) \cdot \nabla u(\boldsymbol{\mu}) = f(\boldsymbol{\mu}) & \text{in } \Omega \\ u(\boldsymbol{\mu}) = g(\boldsymbol{\mu}) & \text{on } \Gamma_D \\ \mathcal{K}(\boldsymbol{\mu}) \nabla u(\boldsymbol{\mu}) \cdot \vec{n} = h(\boldsymbol{\mu}) & \text{on } \Gamma_N. \end{cases} \tag{1.10}$$

Equation (1.10) models the concentration $u(\boldsymbol{\mu})$ of a solute in the domain Ω, transported by the advection field $\vec{b}(\boldsymbol{\mu})$ and subject to molecular diffusion processes or in the case $\vec{b}(\boldsymbol{\mu}) = \vec{0}$ the temperature of a body Ω heated by a source $f(\boldsymbol{\mu})$. In these problems, the parameter $\boldsymbol{\mu}$ can represent, e.g., the intensity and the spatial location of the source $f(\boldsymbol{\mu})$, the diffusion tensor $\mathcal{K}$ describing possible anisotropic materials, or the direction and the magnitude of the advection field, in the case $\vec{b}(\boldsymbol{\mu}) \neq \vec{0}$. Obviously, $\boldsymbol{\mu}$ determines the nature of the solution of (1.10). We assume the diffusion coefficient $\mathcal{K}(\boldsymbol{\mu}) \in \mathbb{R}^{d\times d}$ to be positive definite and the advection field $\vec{b}(\boldsymbol{\mu}) \in \mathbb{R}^d$ to be such that $\nabla \cdot \vec{b}(\boldsymbol{\mu}) = 0$ for any $\boldsymbol{\mu} \in \mathcal{D}$. Furthermore, we introduce the Hilbert space

$$X = H^1_{\Gamma_D}(\Omega) = \left\{ v \in H^1(\Omega) : v\big|_{\Gamma_D} = 0 \right\},$$

and a lifting function $R_g \in H^1(\Omega)$ such that $R_g(\boldsymbol{\mu})\big|_{\Gamma_D} = g(\boldsymbol{\mu})$. Then, the variational formulation of problem (1.10) is cast under the form (1.1) where for any $u, v \in X$ we define the bilinear form

$$a(u, v; \boldsymbol{\mu}) = \int_\Omega \mathcal{K}(\boldsymbol{\mu}) \nabla u \cdot \nabla v \, d\Omega + \int_\Omega (\vec{b}(\boldsymbol{\mu}) \cdot \nabla u) v \, d\Omega,$$

and the linear form

$$f(v; \boldsymbol{\mu}) = \int_\Omega f(\boldsymbol{\mu}) v \, d\Omega + \int_{\Gamma_N} h(\boldsymbol{\mu}) v \, d\Gamma_N - a(R_g(\boldsymbol{\mu}), v; \boldsymbol{\mu}).$$

We will assume that $f(\boldsymbol{\mu})$, $g(\boldsymbol{\mu})$ and $h(\boldsymbol{\mu})$ are chosen such that, together with the assumptions on $\mathcal{K}(\boldsymbol{\mu})$ and $\vec{b}(\boldsymbol{\mu})$, the continuity and the coercivity of $a(\cdot, \cdot; \boldsymbol{\mu})$ and the continuity of $f(\cdot; \boldsymbol{\mu})$ are verified, leading to a strongly coercive problem. The corresponding FE discretization automatically fulfills the coercivity of the bilinear form $a(\cdot, \cdot; \boldsymbol{\mu})$ over $X_h \times X_h$ and hence its well-posedness is verified, yielding, at algebraic level, a nonsingular linear system (1.7).

Saddle-point problems are a special case of weakly coercive problems which are encountered when considering the constrained minimization of a functional over a Hilbert space, for instance when dealing with mixed formulation of elliptic PDEs (e.g. incompressible linear elasticity), the Stokes equations or optimal control for PDEs. More specifically, given two Hilbert spaces V, Q, the bilinear forms $d(\cdot, \cdot; \boldsymbol{\mu}) : V \times V \to \mathbb{R}$, $b(\cdot, \cdot; \boldsymbol{\mu}) : V \times Q \to \mathbb{R}$ and the linear functionals $f_1(\cdot; \boldsymbol{\mu}) : V \to \mathbb{R}$, $f_2(\cdot; \boldsymbol{\mu}) : Q \to \mathbb{R}$, we consider the saddle-point problem: find $u(\boldsymbol{\mu}) \in V$ and $p(\boldsymbol{\mu}) \in Q$ such that

$$\begin{cases} d(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) + b(v, p(\boldsymbol{\mu}); \boldsymbol{\mu}) = f_1(v; \boldsymbol{\mu}) & \forall v \in V \\ b(u(\boldsymbol{\mu}), q; \boldsymbol{\mu}) = f_2(q; \boldsymbol{\mu}) & \forall q \in Q. \end{cases} \tag{1.11}$$

Problem (1.11) can be analyzed in the framework of weakly coercive problems by considering $X = V \times Q$, $y(\boldsymbol{\mu}) = (u(\boldsymbol{\mu}), p(\boldsymbol{\mu})) \in X$ and, for any (u, p), $(v, q) \in V \times Q$, by defining the forms

$$\begin{aligned} a((u,p),(v,q);\boldsymbol{\mu}) &= d(u,v;\boldsymbol{\mu}) + b(v,p;\boldsymbol{\mu}) + b(u,q;\boldsymbol{\mu}) \\ f((v,q);\boldsymbol{\mu}) &= f_1(v;\boldsymbol{\mu}) + f_2(q;\boldsymbol{\mu}), \end{aligned} \tag{1.12}$$

see [Brezzi, 1974, Boffi et al., 2013, Xu and Zikatanov, 2003] for further details. When a saddle-point problem as (1.11) is discretized through the FE method, the resulting stiffness matrix $\mathbf{A}(\boldsymbol{\mu})$ has the following block structure

$$\mathbf{A}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{A}_{1,1}(\boldsymbol{\mu}) & \mathbf{A}_{2,1}^T(\boldsymbol{\mu}) \\ \mathbf{A}_{2,1}(\boldsymbol{\mu}) & O \end{bmatrix}, \tag{1.13}$$

with $\mathbf{A}_{1,1}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^1 \times N_h^1}$ symmetric and positive definite and $\mathbf{A}_{2,1}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^2 \times N_h^1}$; the dimensions N_h^1 and N_h^2 are such that $N_h = N_h^1 + N_h^2$. The matrix appearing in (1.13) is a special case of matrices showing a more general block structure,

$$\mathbf{A}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{A}_{1,1}(\boldsymbol{\mu}) & \mathbf{A}_{1,2}(\boldsymbol{\mu}) \\ \mathbf{A}_{2,1}(\boldsymbol{\mu}) & -\mathbf{A}_{2,2}(\boldsymbol{\mu}) \end{bmatrix}, \tag{1.14}$$

where $\mathbf{A}_{1,1}(\boldsymbol{\mu})$ is positive definite and $\mathbf{A}_{2,1}(\boldsymbol{\mu})$ is as in (1.13), $\mathbf{A}_{1,2}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^1 \times N_h^2}$ and $\mathbf{A}_{2,2} \in \mathbb{R}^{N_h^2 \times N_h^2}$ is positive semidefinite. Saddle-point problems are encountered in many engineering fields, for this reason the preconditioning of saddle-point systems will be specifically addressed in the next section. By assuming that a couple of FE spaces satisfying the *inf-sup* condition (1.6) is employed, $\mathbf{A}(\boldsymbol{\mu})$ is proven to be an indefinite nonsingular matrix. For an in-depth discussion of saddle-point problems, we refer, e.g., to [Boffi et al., 2013, Brezzi, 1974, Brezzi and Bathe, 1990, Benzi et al., 2005].

1.1.2 Unsteady problems

Parabolic PDEs

Given a time interval $(0,T)$, $T > 0$, we consider the following evolutionary problem: for any $\boldsymbol{\mu} \in \mathcal{D}$, find $y(\boldsymbol{\mu}) = y(t;\boldsymbol{\mu}) \in X$ such that for any $t \in (0,T)$

$$\begin{cases} \Big(\dfrac{\partial y(\boldsymbol{\mu})}{\partial t}, w\Big)_{L^2(\Omega)} + a(y(\boldsymbol{\mu}), w; \boldsymbol{\mu}) = f(w;\boldsymbol{\mu}) & \forall w \in X \\ y(0;\boldsymbol{\mu}) = y_0(\boldsymbol{\mu}) & \text{in } \Omega, \end{cases} \tag{1.15}$$

where $y_0(\boldsymbol{\mu}) \in L^2(\Omega)$ is the assigned initial datum. We consider forms $a(\cdot,\cdot;\boldsymbol{\mu})$ and $f(\cdot;\boldsymbol{\mu})$ verifying the same hypotheses of continuity and coercivity as in (1.1), yielding the well-posedness of (1.15) (see e.g. [Quarteroni and Valli, 2008, Quarteroni, 2014]).

Similarly to the steady case, the FE method is employed to discretize problem (1.15). This is carried out by introducing a finite dimensional space $X_h \subset X$ and by considering the following finite dimensional problem: for any $\boldsymbol{\mu} \in \mathcal{D}$, find $y_h(\boldsymbol{\mu}) = y_h(t;\boldsymbol{\mu}) \in X_h$ such that for any $t \in (0,T)$

$$\begin{cases} \Big(\dfrac{\partial y_h(\boldsymbol{\mu})}{\partial t}, w_h\Big)_{L^2(\Omega)} + a(y_h(\boldsymbol{\mu}), w;\boldsymbol{\mu}) = f(t, w_h;\boldsymbol{\mu}) & \forall w_h \in X_h \\ y_h(0;\boldsymbol{\mu}) = y_{h0}(\boldsymbol{\mu}) & \text{in } \Omega, \end{cases} \tag{1.16}$$

where $y_{h0}(\boldsymbol{\mu})$ is obtained as L^2-projection of $y_0(\boldsymbol{\mu})$ over X_h. The algebraic representation of problem (1.16) results in the following dynamical system

$$\begin{cases} \mathbf{M}(\boldsymbol{\mu})\dfrac{d\mathbf{y}(\boldsymbol{\mu})}{dt} + \mathbf{A}(\boldsymbol{\mu})\mathbf{y}(\boldsymbol{\mu}) = \mathbf{f}(t;\boldsymbol{\mu}), & t \in (0,T) \\ \mathbf{y}(0;\boldsymbol{\mu}) = \mathbf{y}_0(\boldsymbol{\mu}), \end{cases} \tag{1.17}$$

where the stiffness matrix $\mathbf{A}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ and the FE vector $\mathbf{f}(t;\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ are defined as in (1.8) and $\mathbf{y}_0(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ is the FE vector representation of $y_{h0}(\boldsymbol{\mu})$. Similarly, the mass matrix $\mathbf{M}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ is such that

$$\big(\mathbf{M}(\boldsymbol{\mu})\big)_{ij} = (\phi_j^x, \phi_i^x)_{L^2(\Omega)} \qquad i,j = 1,\ldots,N_h. \tag{1.18}$$

Problem (1.17) is then discretized in time by introducing a partition of the time interval [0,T] into $N_t \in \mathbb{N}_+$ subintervals of size $\Delta t = T/N_t$, such that $t_n = n\Delta t$, $n = 0,\ldots,N_t$ and employing a time-discretization scheme to obtain a sequence (in time) of linear systems to be solved for each parameter $\boldsymbol{\mu}$ considered.

We will employ the backward differentiation formula (BDF) of order 1 or 2 for unsteady problems in Chapter 2 and 4. In both cases, $\mathbf{y}^n(\boldsymbol{\mu})$ denotes the approximation of $\mathbf{y}(\boldsymbol{\mu})$ at time t_n. BDF methods collect a family of implicit linear multistep schemes, for which the value of the time derivative at t_{n+1} is approximated by the value at t_{n+1} of the derivative of the Lagrange polynomial which interpolates the numerical solution at the previous times $t_{n+1}, t_n, \ldots, t_{n-\sigma_1+1}$: σ_1 denotes the BDF order, see [Ascher and Petzold, 1998, Brenan et al., 1995, Quarteroni et al., 2007]. We can generally represent the BDF scheme of order σ_1 as

$$\frac{d\mathbf{y}(\boldsymbol{\mu})}{dt} \approx \frac{\alpha_1 \mathbf{y}^{n+1}(\boldsymbol{\mu}) - \mathbf{y}^{n,\sigma_1}(\boldsymbol{\mu})}{\Delta t}; \tag{1.19}$$

in the numerical examples presented, we will limit ourselves to the case $\sigma_1 = \{1,2\}$, for which

$$\mathbf{y}^{n,\sigma_1}(\boldsymbol{\mu}) = \begin{cases} \mathbf{y}^n(\boldsymbol{\mu}), & n \geq 0 \text{ and } \sigma_1 = 1 \\ 2\mathbf{y}^n(\boldsymbol{\mu}) - \dfrac{1}{2}\mathbf{y}^{n-1}(\boldsymbol{\mu}), & n \geq 1 \text{ and } \sigma_1 = 2 \end{cases} \tag{1.20}$$

and $\alpha_1 = 1$, $3/2$ for $\sigma_1 = 1$ or $\sigma_1 = 2$, respectively. When considering a new parameter

$\boldsymbol{\mu}$, the following sequence of linear systems must be solved for any $n = 0, \dots, N_t - 1$

$$\mathbf{M}(\boldsymbol{\mu}) \frac{\alpha_1 \mathbf{y}^{n+1}(\boldsymbol{\mu}) - \mathbf{y}^{n,\sigma_1}(\boldsymbol{\mu})}{\Delta t} + \mathbf{A}(\boldsymbol{\mu}) \mathbf{y}^{n+1}(\boldsymbol{\mu}) = \mathbf{f}^{n+1}(\boldsymbol{\mu}), \tag{1.21}$$

or, equivalently,

$$\left(\frac{\alpha_1}{\Delta t} \mathbf{M}(\boldsymbol{\mu}) + \mathbf{A}(\boldsymbol{\mu}) \right) \mathbf{y}^{n+1}(\boldsymbol{\mu}) = \mathbf{f}^{n+1}(\boldsymbol{\mu}) + \frac{1}{\Delta t} \mathbf{M}(\boldsymbol{\mu}) \mathbf{y}^{n,\sigma_1}(\boldsymbol{\mu}), \tag{1.22}$$

with $\mathbf{y}^0 = \mathbf{y}_0$. We highlight that BDF1 corresponds to the well-known backward Euler scheme.

In this thesis, the unsteady parametrized Navier-Stokes equations, which describe the dynamics of a fluid in a domain Ω, are considered. Their (differential and weak) formulation and FE discretization are specifically addressed in Chapter 4; however, here we limit ourselves to comment the fact that when discretizing such a problem with the FE method in space and BDF in time, a nonlinear system similar to (1.21) is obtained, where $\mathbf{A}(\boldsymbol{\mu})$ is a nonlinear saddle-point matrix showing the same block structure of the matrix in (1.13).

1.2 Review on solution strategies for parametrized FE systems

As seen in the previous section, when dealing with the numerical approximation of PDEs, any discretization method leads to a (sequence of) linear system as those appearing in (1.7) or (1.21). This calls for the use of an efficient solver, especially when dealing with large scale problems. In this section we recall some state-of-the-art techniques. Although specifically referring to the steady problem (1.7), the great majority of the methods and ideas presented is easily extendable to the time dependent case by replacing $\mathbf{A}(\boldsymbol{\mu})$ with the matrix at the left hand side of (1.21). In the next paragraphs, we will also omit the dependence on $\boldsymbol{\mu}$ for ease of notation.

Parallel direct methods are based on suitable modifications of LU and Cholesky factorizations. They are very efficient on 2D discretizations, however, when dealing with large scale 3D FE discretizations, their scalability performances worsen, making less convenient their application. An alternative approach consists in using Krylov iterative solvers, see [Saad, 2003, Greenbaum, 1997, Van der Vorst, 2003] for a full description of these techniques. Given the initial guess $\mathbf{y}^{(0)}$ for solving system (1.7), with residual $\mathbf{r}^{(0)} = \mathbf{f} - \mathbf{A}\mathbf{y}^{(0)}$, the k-th iterate $\mathbf{y}^{(k)}$ satisfies

$$\mathbf{y}^{(k)} \in \mathbf{y}^{(0)} + \mathcal{K}_k(\mathbf{A}, \mathbf{r}^{(0)}), \tag{1.23}$$

where $\mathcal{K}_k(\mathbf{A}, \mathbf{r}^{(0)})$ denotes the well-known Krylov subspace of dimension k, such that

$$\mathcal{K}_k(\mathbf{A}, \mathbf{r}^{(0)}) = span\Big\{\mathbf{r}^{(0)},\, \mathbf{A}\mathbf{r}^{(0)},\, \dots,\, \mathbf{A}^{k-1}\mathbf{r}^{(0)}\Big\}. \tag{1.24}$$

The k-th solution is obtained by minimizing the norm of the residual over the Krylov subspace $\mathcal{K}_k(\mathbf{A}, \mathbf{r}^{(0)})$, and the minimization process characterizes the method. Relevant examples are the full orthogonalization method (FOM) [Saad, 1981], the minimal residual (MINRES) method for nonsingular symmetric (and possibly indefinite) matrices [Paige and Saunders, 1975], the conjugate gradient (CG) method for symmetric positive definite (SPD) matrices [Hestenes and Stiefel, 1952] and the generalized minimal residual (GMRES) method for general nonsingular matrices [Saad and Schultz, 1986]. The cost per iteration of these methods is comparable to a matrix-vector multiplication and their convergence rate highly depends on the spectral properties of the system, that is the relative position of the eigenvalues of the matrix on the complex plane. In the case of symmetric matrices, a relevant role is played by the condition number of the stiffness matrix, which in an elliptic PDE problem discretized with the FE method on a computational grid with characteristic step size h is $O(1/h^2)$. In order to maintain optimal rate of convergence with computational grids featuring small values of h, these methods need to be suitably preconditioned.

1.2.1 A short survey on classical preconditioning techniques

Preconditioning a linear system means to turn it into another (equivalent) system with more favorable spectral properties. Ideally, the preconditioned matrix has a (much) smaller condition number than the original one. Hereon, $\mathbf{P} \in \mathbb{R}^{N_h \times N_h}$ will denote a generic (nonsingular) preconditioner, and instead of solving (1.7), we are called to solve the corresponding preconditioned system

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{y} = \mathbf{P}^{-1}\mathbf{f}. \tag{1.25}$$

We refer to (1.25) as the *left* preconditioned system, since a right or symmetric preconditioning approach are also viable alternatives [Elman et al., 2005, Wathen, 2015]. In the former case we have that (1.7) is substituted by

$$\mathbf{A}\mathbf{P}^{-1}\mathbf{w} = \mathbf{f}, \qquad \mathbf{y} = \mathbf{P}^{-1}\mathbf{w}, \tag{1.26}$$

whereas in the latter two nonsingular preconditioners $\mathbf{P}_{\mathrm{L}} \in \mathbb{R}^{N_h \times N_h}$ and $\mathbf{P}_{\mathrm{R}} \in \mathbb{R}^{N_h \times N_h}$ are introduced and one is called to solve

$$\mathbf{P}_{\mathrm{L}}^{-1}\mathbf{A}\mathbf{P}_{\mathrm{R}}^{-1}\mathbf{w} = \mathbf{P}_{\mathrm{L}}^{-1}\mathbf{f}, \qquad \mathbf{y} = \mathbf{P}_{\mathrm{R}}^{-1}\mathbf{w}.$$

In this thesis we will mostly employ a right preconditioning approach. Left preconditioners will be used only occasionally.

Consider, for instance, the preconditioned CG (PCG) method. If $\mathbf{y}^{(0)} \in \mathbb{R}^{N_h}$ is the given initial guess and $\mathbf{A}$ and $\mathbf{P}$ are SPD matrices, the error at the k-th iteration is such that

$$\frac{\|\mathbf{y} - \mathbf{y}^{(k)}\|_{\mathbf{A}}}{\|\mathbf{y} - \mathbf{y}^{(0)}\|_{\mathbf{A}}} \leq 2 \left(\frac{\sqrt{\mathcal{K}_2(\mathbf{P}^{-1}\mathbf{A})} - 1}{\sqrt{\mathcal{K}_2(\mathbf{P}^{-1}\mathbf{A})} + 1} \right)^k, \tag{1.27}$$

where $\mathcal{K}_2$ is the condition number with respect to the Euclidean norm. Clearly from (1.27), the smaller the condition number $\mathcal{K}_2(\mathbf{P}^{-1}\mathbf{A})$, the faster the convergence rate of PCG, and in practice a good (SPD) preconditioner $\mathbf{P}$ for a SPD matrix should have two properties:

- it minimizes as much as possible $\mathcal{K}_2(\mathbf{P}^{-1}\mathbf{A})$;
- the application of $\mathbf{P}^{-1}$ is very cheap.

Suitably adapted, an equivalent of these statements also holds for nonsymmetric and/or indefinite problems: a preconditioner should turn the linear system into a new one with improved spectral properties and the the application of $\mathbf{P}^{-1}$ to a FE vector should feature a low computational cost.

Classical examples of preconditioning techniques hugely employed in large scale simulations are the multilevel (or multigrid) method and the domain decomposition (DD) method. As they will be employed throughout this thesis, we recall in the following paragraphs the basic ideas of these classes of preconditioners.

Domain decomposition preconditioners

DD methods rely on a partition of the domain Ω into M overlapping or non-overlapping subdomains; a local problem is then associated to each subdomain [Quarteroni and Valli, 1999, Toselli and Widlund, 2005, Smith et al., 1996]. The problem can be partitioned through specific algorithms which divide the computational domain, or a division could be a feature of the problem itself, e.g. when dealing with multiphysics problems. Then, at every iteration of the Krylov method, each local problem is solved exactly or approximately, using direct solvers, incomplete factorizations or relaxation methods depending on the chosen algorithm. The classical Schwarz method has been successfully employed for elliptic and saddle-point problems [Klawonn and Pavarino, 1998, Klawonn and Pavarino, 2000, Toselli and Widlund, 2005]. It exploits an overlap among the subdomains to ensure communication among the subdomains, and requires to solve exactly M local problems at every iteration. Given a domain Ω and a partition $\{\Omega_i\}_{i=1}^M$ such that $\overline{\Omega} = \cup_{i=1}^M \overline{\Omega}_i$, the additive 1-level form of the additive Schwarz (AS) preconditioner can be expressed as

$$\mathbf{P}_{\text{AS}}^{-1} = \sum_{i=1}^{M} \mathbf{R}_i \mathbf{A}_i^{-1}(\boldsymbol{\mu}) \mathbf{R}_i^T \tag{1.28}$$

where $\mathbf{R}_i$ is the restriction over the subdomain Ω_i and $\mathbf{A}_i^{-1}(\boldsymbol{\mu})$ is the local inverse matrix. For elliptic problems, the condition number of the preconditioned matrix is such that

$$\mathcal{K}_2(\mathbf{P}_{\text{AS}}^{-1}\mathbf{A}) \leq C_1 \frac{1}{H\delta_{\text{AS}}} \tag{1.29}$$

where δ_{AS} is the characteristic size of the overlap between two subdomains and H denotes the characteristic size of the partition, see [Toselli and Widlund, 2005]. We highlight that the condition number of the preconditioned matrix increases by decreasing the size of the subdomains, i.e. the AS preconditioner is not scalable, meaning that if each local problem is associated to a core in a parallel computing environment, the number of iterations needed to reach a fixed tolerance grows with the number of cores. A remedy to overcome this fact consists in adding a coarse grid level, whose characteristic size is H, that allows to enhance the communication among the cores. In this case, we can define the 2-level AS preconditioner as

$$\mathbf{P}_{\text{CAS}}^{-1} = \mathbf{P}_{\text{AS}}^{-1} + \mathbf{R}_0\mathbf{A}_0^{-1}\mathbf{R}_0^T, \tag{1.30}$$

for which

$$\mathcal{K}_2(\mathbf{P}_{\text{CAS}}^{-1}\mathbf{A}) \leq C_2 \frac{H}{\delta_{\text{AS}}}; \tag{1.31}$$

here $\mathbf{R}_0$ is the restriction matrix over the coarse grid and $\mathbf{A}_0^{-1}$ is the inverse of the coarse problem matrix. This latter option thus ensures the scalability of the resulting preconditioner. In a parameter-dependent context, the preconditioners $\mathbf{P}_{\text{AS}}^{-1}$ and $\mathbf{P}_{\text{CAS}}^{-1}$ (and specifically the local matrices $\mathbf{A}_i$, $i = 0, \dots, M$, and their inverse matrices), need to be recomputed for each new instance of $\boldsymbol{\mu}$; moreover, the coefficients C_1, C_2 appearing in (1.29) and (1.31) will also depend on $\boldsymbol{\mu}$, possibly yielding a nonuniform performance across the parameter space $\mathcal{D}$.

Another well-known branch of DD techniques is based on substructuring methods, which includes the 1-level and 2-level finite element tearing and interconnect (FETI) method and the balancing domain decomposition by constraints (BDDC) method, which have been widely used in structural and fluid mechanics problems. The former has been firstly introduced in the early 90's in [Farhat and Roux, 1991], and later improved in its FETI-DP variant, where DP stands for Dual-Primal, see [Farhat et al., 2001]. These methods rely on a non-connected partition of the domain with corresponding local problems, whose compatibility and communication at the interfaces are enforced by the introduction of suitable Lagrange multipliers. FETI preconditioners have been developed for the iterative solution of second-order solid mechanics and fourth-order beam problems, but they have also been applied to fluid dynamics problems, e.g. in [Li, 2002, Li, 2005]. On the other hand, the BDDC method has been introduced in [Dohrmann, 2003, Mandel and Dohrmann, 2003] for second- and fourth-order structural mechanics problems, and later applied to relevant classes of problems such as the (Navier-)Stokes equations and Maxwell equations for electromagnetic fields. It employs constraints associated with disjoint sets of nodes on subdomain boundaries and a coarse problem to ensure the independence of the condition number from the number of subdomains. In demanding application, the size of the coarse problem represents a bottleneck in terms of computational time, therefore this technique has been generalized to a multilevel version that recursively applies the BDDC algorithm, see e.g. [Mandel and Dohrmann, 2007].

Multilevel preconditioners

Multilevel (or multigrid) methods have been firstly introduced by [Brandt, 1977], and since then they have been on constant development; we refer to, e.g. [Hackbusch, 2013, Trottenberg et al., 2000, Briggs et al., 2000, Elman et al., 2005, Bramble et al., 1990] for an extensive outlook of these methods and limit ourselves in recalling their basic principles. The core idea of these methods lies in iteratively solving the same PDE problem over a sequence of meshes (or levels); each solution is then used to damp the high frequencies of the error introduced at its corresponding mesh level. This operation is called *smoothing*, and carried out by applying (one or multiple times per grid) a smoothing operator, which is often chosen as a simple Gauss-Seidel or Jacobi operator. At the coarse level, the final linear system features a small dimension, consequently it is often convenient to use a direct solver to obtain its solution; this latter is then used as coarse correction. Restriction and prolongation operators are used to communicate the information from one grid to another. Starting from this concept a huge variety of multilevel methods has been developed; in particular, we mention the well-known V- and W- cycle multigrid, whose examples are shown in Figure 1.1.

All the multilevel methods discussed so far rely on the fact that the linear system comes from the discretization of a PDE over a computational domain, or at least a geometrical knowledge of the problem is available; this is why they are often referred to as *geometric* multigrid methods. However, the same concept can be extended to those cases when one is just given a matrix (instead of the problem which leads to that matrix), giving birth to *algebraic* multigrid methods. These techniques try to mimic the construction of the smoothing and coarse correction operators of the geometric case only by relying on the knowledge of the matrix coefficient. See e.g. [Stüben, 2001, Xu and Zikatanov, 2017, Saad, 2003] for a description of algebraic multigrid methods.

Multigrid methods have been firstly introduced as solvers, however the multigrid operator provides a spectrally equivalent preconditioner for the matrix, such that the condition number of the preconditioned stiffness matrix is independent of the characteristic size h, under suitable assumptions on the smoothing, prolongation and restriction operators, see e.g. [Elman et al., 2005]. Nowadays, they can be considered among the state-of-the-art preconditioning techniques for problems featuring a symmetric positive definite matrix. They can be straightforwardly applied also to non-symmetric ones, as advection-diffusion

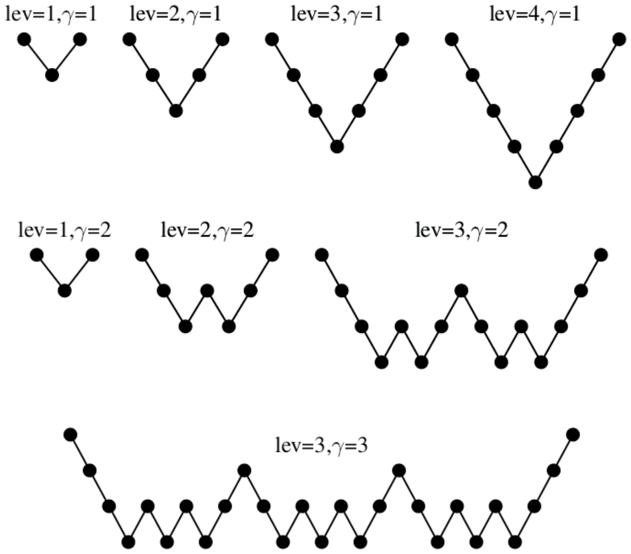


Figure 1.1 – Examples of multigrid cycles, image taken from [Saad, 2003]. The grid cycles are characterized by lev coarsening steps and γ iterations of the classic 2-level multigrid cycle.

equation, however for the latter case the smoothing and coarse operations must be wisely selected in order to avoid any reintroduction of high frequencies by the coarse operator, e.g. by smoothing in a preferred direction provided by the convective term [Ramage, 1999, Yavneh et al., 1998]. Moreover, as already mentioned, these methods rely on the construction of restriction and prolongation operators, whose assembling can be demanding, especially in the case an algebraic multigrid method is employed. This issue is even emphasized when dealing with parametrized problems, since the restriction and prolongation operators are based on the relative magnitude of the matrix coefficients, which can largely change depending on the parameter range of $\boldsymbol{\mu}$.

Preconditioners for saddle-point problems

Depending on the problem at hand, the specific structure of the system matrix $\mathbf{A}(\boldsymbol{\mu})$ can be exploited to develop an efficient preconditioner; this is the case of saddle-point (or mixed variational) problems. In the following we limit ourselves in reporting the basic ingredients to get this thesis self-contained, we point to [Benzi et al., 2005, Benzi and Wathen, 2008, Axelsson and Neytcheva, 2003, Zulehner, 2002] for a review on numerical methods for saddle-point systems.

A linear system involving a matrix of the form (1.14) occurs in many contexts; among the others, we recall the Stokes equations, optimal control for elliptic and incompressible fluid flows problems and incompressible elasticity just to mention some relevant cases.

As a result, the numerical solution of such linear system has been a prominent research subject in the last decades, and several techniques involving domain decomposition and (algebraic) multilevel methods have been proposed, see e.g. [Elman et al., 2005, Ghia et al., 1982, Turek, 1999, Wesseling and Oosterlee, 2001, Wittum, 1989] and references therein. On the other hand, block preconditioners are designed to explicitly exploit the block structure of $\mathbf{A}(\boldsymbol{\mu})$, providing optimal solvers for many types of saddle-point systems. Their main drawback is however that they are often very specialized, that is, by changing the problem (e.g. from steady to unsteady Stokes equations) the preconditioner should be changed. Starting from (1.13) the following factorization of the saddle-point matrix holds:

$$\mathbf{A} = \mathcal{L}\mathcal{D}\mathcal{U} \tag{1.32}$$

where

$$\mathcal{L} = \begin{bmatrix} \mathbf{I}_{N_h^1} & 0 \\ \mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1} & \mathbf{I}_{N_h^2} \end{bmatrix}, \qquad \mathcal{U} = \begin{bmatrix} \mathbf{I}_{N_h^1} & \mathbf{A}_{1,1}^{-1}\mathbf{A}_{2,1}^T \\ 0 & \mathbf{I}_{N_h^2} \end{bmatrix} \qquad \mathcal{D} = \begin{bmatrix} \mathbf{A}_{1,1} & 0 \\ 0 & \mathbf{S} \end{bmatrix}. \tag{1.33}$$

The matrix

$$\mathbf{S} = -\mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{A}_{2,1}^T \tag{1.34}$$

is the Schur complement matrix and plays an essential role in designing an efficient preconditioner for $\mathbf{A}$. Block preconditioners are indeed devised by considering factorization (1.32) where $\mathbf{A}_{1,1}$ and $\mathbf{S}$ are approximated by their (ideally spectrally equivalent) surrogates $\widetilde{\mathbf{A}}^{1,1}$ and $\widetilde{\mathbf{S}}$, respectively, since, as a matter of fact, the cost of using directly (1.32) would be equivalent to invert the matrix $\mathbf{A}$. Hence, the following factorization is employed in practice

$$\mathbf{A} \approx \widetilde{\mathcal{L}}\widetilde{\mathcal{D}}\widetilde{\mathcal{U}}, \tag{1.35}$$

where the surrogate operators $\widetilde{\mathbf{A}}_{1,1}$, $\widetilde{\mathbf{S}}$, such that

$$\widetilde{\mathbf{A}}_{1,1}^{-1} \approx \mathbf{A}_{1,1}^{-1}, \qquad \widetilde{\mathbf{S}}^{-1} \approx \mathbf{S}^{-1} \tag{1.36}$$

are used in (1.33) at the places of $\mathbf{A}_{1,1}$ and $\mathbf{S}$

$$\widetilde{\mathcal{L}} = \begin{bmatrix} \mathbf{I}_{N_h^1} & 0 \\ \mathbf{A}_{2,1}\widetilde{\mathbf{A}}_{1,1}^{-1} & \mathbf{I}_{N_h^2} \end{bmatrix}, \qquad \widetilde{\mathcal{U}} = \begin{bmatrix} \mathbf{I}_{N_h^1} & \widetilde{\mathbf{A}}_{1,1}^{-1}\mathbf{A}_{2,1}^T \\ 0 & \mathbf{I}_{N_h^2} \end{bmatrix} \qquad \widetilde{\mathcal{D}} = \begin{bmatrix} \widetilde{\mathbf{A}}_{1,1} & 0 \\ 0 & \widetilde{\mathbf{S}} \end{bmatrix}. \tag{1.37}$$

The matrices $\widetilde{\mathbf{A}}_{1,1}$ and $\widetilde{\mathbf{S}}$ are commonly chosen either as preconditioners of $\mathbf{A}_{1,1}$ and $\mathbf{S}$ or consist of inner iterations, where the linear system featuring at the left hand side the matrix $\mathbf{A}_{1,1}$ (resp. $\widetilde{\mathbf{S}}$) is solved up to a certain tolerance. Notice that in the latter case, one should rely on a flexible iterative algorithm, which allows to change preconditioner from one iteration to another.

The block-preconditioning strategy has been particularly exploited in the field of fluid dynamics and optimal control problems. Block diagonal preconditioners are obtained by setting

$$\mathbf{P} = \begin{bmatrix} \widetilde{\mathbf{A}}_{1,1} & 0 \\ 0 & -\widetilde{\mathbf{S}} \end{bmatrix},$$

see [Elman and Silvester, 1996, De Sturler and Liesen, 2005, Wathen and Silvester, 1993]. Instead, block-triangular preconditioners can be constructed by considering the product

$$\mathbf{P} = \widetilde{\mathcal{L}}\widetilde{\mathcal{D}} \qquad \text{or} \qquad \mathbf{P} = \widetilde{\mathcal{D}}\widetilde{\mathcal{U}},$$

see [Silvester and Wathen, 1994]. Relevant examples are the least-squares commutator (LSC) preconditioner [Elman et al., 2006, May and Moresi, 2008], the pressure-convection-diffusion preconditioner [Kay et al., 2002, Silvester et al., 2001] and the pressure mass matrix (PMM) preconditioner [Rehman et al., 2011]. Finally, SIMPLE type preconditioners are obtained with the factorization (1.35), where $\mathbf{A}_{1,1}$ is substituted with its diagonal when building the Schur complement and for the update of the velocity, [Vuik et al., 2000, Elman et al., 2008, Little and Saad, 2003, Forti, 2016]; the SIMPLE preconditioner will be discussed more in-depth in Chapter 4, where it is considered for solving the unsteady NS equations. As a matter of fact, SIMPLE preconditioners are particularly effective when dealing with problems diagonally dominant, as in the case of unsteady problems with relative small timesteps. In Chapter 3 and 4 we will deal with saddle-point problems arising in fluid dynamics, and we will take advantage of block-like preconditioners to design our efficient iterative solvers.

1.2.2 Solution strategies for sequences of linear systems

All the solution methods described in previous sections are developed for a single linear system, that is, they do not consider the case where a sequence of linear systems, possibly related one to each other, is taken into account. However, the repeated solution of parametric PDEs is a frequent issue when dealing, e.g., with sensitivity analysis, problems involving random input data, PDE-constrained optimization. In all these cases, solving the same problem for a huge number of parameters may become a critical bottleneck. As a matter of fact, the CPU time required by the solution of (1.7), which depends on $N_h = dim(X_h)$, can be highly demanding even on modern parallel architectures, since in some extreme cases, N_h can be of order $O(10^q)$, $6 \leq q \leq 9$. This issue is emphasized when dealing with time-dependent or non linear problems treated in an implicit fashion, due to the time advancing scheme employed within the simulation, the steps of a Newton method, or even both at the same time.

Taking advantage of storing repeated solutions to similar systems can enhance efficiency in such a context. For instance, several Krylov-subspace recycling approaches have been introduced to handle sequences of linear systems arising, e.g., from parametrized, time-dependent and/or nonlinear PDEs, see [Saad, 1997, Simoncini and Szyld, 2007]. The strategy consists in augmenting the usual Krylov subspace with data retrieved from previous cycles (in the case of restarted algorithms) or solves (in the case of problems with both varying matrices and right hand sides). For instance, the first contributions in this field made use of the whole Krylov subspaces of previous solutions of linear systems, see e.g. [Farhat et al., 1994, Risler and Rey, 2000, Roux, 1995, Saad, 1987], yielding however a severe computational and memory effort, especially when the problem features a large dimension and a slow convergence. Consequently, research has focused on truncation methods that select a limited number of (significant) linear combinations of Krylov vectors. For the solution of a single linear system of equations, in [De Sturler, 1996, De Sturler, 1999] the authors propose optimal truncation strategies of the GCR (generalized conjugate residual) method (GCRO), while in [Morgan, 2002, Chapman and Saad, 1997, Erhel et al., 1996] deflation techniques to find an approximation of the eigenvectors associated to the extremal eigenvalues are employed. See, e.g., [Eiermann et al., 2000, Nabben and Vuik, 2006] for further details related to these techniques. These methods have been extended to the case of a sequence of linear systems with varying right hand sides in [Saad et al., 2000], where a deflated version of the CG algorithm is presented, and in [Parks et al., 2006] where the GCRO method is combined with deflated restarting for sequences of linear systems where both matrices and right hand sides vary. Krylov subspace methods have been exploited in the context of ROM to deal with sequences of single linear systems in [Benner and Feng, 2011] and in the iterative rational Krylov algorithm (IRKA) for sequences of dual linear systems in [Ahuja et al., 2012]. More recently, proper orthogonal decomposition (POD) has been used in the context of solving a sequence of linear systems: in [Carlberg et al., 2016] POD-ROM has been employed to truncate the augmented Krylov subspace and retain only the high-energy modes. This technique, suited for linear systems with symmetric matrices, allows to compute efficiently inexact (yet, very accurate) solutions. In [Cortes et al., 2018], deflation vectors constructed through a POD-reduced set of snapshots are used to accelerate the deflated PCG method and applied to porous media problems.

Concerning the preconditioning of parametrized linear systems, remarkable efforts have been devoted to preconditioning strategies for shifted linear systems. At first, these techniques compute a preconditioner for the unshifted high-fidelity matrix, and then they suitably modify it for the shifted matrix. This has proven to be particularly helpful when employing time-advancing schemes with adaptively chosen time steps, see [Bellavia et al., 2011, Benzi and Bertaccini, 2003, Ferronato et al., 2012]. More recently, techniques to deal with sequences of (not necessarily shifted) linear systems, which compute approximate inverse (AINV) preconditioners by interpolation, have been developed in [Bertaccini and Durastante, 2016]. Furthermore, in [Zahm and Nouy, 2016] a preconditioner for the parametrized high-fidelity problem (1.7) which relies on an interpolation of the matrix inverse based on a pre-computed basis of matrix inverses corresponding to selected values of the parameter has been introduced. This latter method stores the basis of inverted matrices as exact factorizations, thus yielding a huge amount of storage memory, and is computationally efficient only for relatively small problems. Finally, in [Kressner and Tobler, 2011], a low-rank tensor approximation of $\mathbf{y}(\boldsymbol{\mu})$ has been exploited to present low-rank tensor variants of short-recurrence Krylov subspace methods.

1.3 Review on RB methods for parametrized steady PDEs

We start with a brief overview of the RB method for parametrized PDEs, for both elliptic and parabolic problems. Extensions to linear saddle-point problems will be presented in Chapter 3, whereas in Chapter 4 we will consider unsteady (nonlinear) problems. For an introduction to the RB method see, e.g., [Quarteroni et al., 2016a, Hesthaven et al., 2016]; here we limit ourselves to recall the construction of the RB approximation by means of *state reduction* and *system approximation*. The former is carried out by a POD-ROM strategy and guarantees that an accurate approximation to the FE solution is found; the latter, by employing the (discrete) Empirical Interpolation Method (EIM) [Barrault et al., 2004, Chaturantabut and Sorensen, 2010, Grepl et al., 2007, Maday et al., 2009], ensures that the corresponding RB system can be assembled in an inexpensive way.

1.3.1 Building the RB approximation

The RB method relies on the idea that the $\boldsymbol{\mu}$-dependent solution of the $N_h \times N_h$ high-fidelity problem (1.7) can be well approximated by a linear combination of $N \ll N_h$ FE solutions of (1.7) corresponding to (suitably chosen) parameter values. The computation is usually based on an *offline/online* splitting. In the former phase, a *reduced space* (the RB space)

$$V_N = span\{\xi_i,\, i = 1, \ldots, N\} \subset X_h, \tag{1.38}$$

with dimension N, is built; algebraically V_N is represented by the matrix $\mathbf{V} \in \mathbb{R}^{N_h \times N}$, $\mathbf{V} = [\boldsymbol{\xi}_1 | \ldots | \boldsymbol{\xi}_N]$, where $\{\boldsymbol{\xi}_i\}_{i=1}^N$ are the FE vector representation of $\{\xi_i\}_{i=1}^N$. In the latter, the high-fidelity problem (1.7) is replaced by a reduced problem of very small dimension for any new instance of the parameter $\boldsymbol{\mu}$:

$$\mathbf{A}_N(\boldsymbol{\mu})\mathbf{y}_N(\boldsymbol{\mu}) = \mathbf{f}_N(\boldsymbol{\mu}). \tag{1.39}$$

Here $\mathbf{y}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ is the RB solution, $\mathbf{A}_N(\boldsymbol{\mu}) \in \mathbb{R}^{N\times N}$ the RB matrix and $\mathbf{f}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ the RB right hand side, respectively defined as

$$\mathbf{A}_N(\boldsymbol{\mu}) = (\mathbf{W}(\boldsymbol{\mu}))^T\mathbf{A}(\boldsymbol{\mu})\mathbf{V}, \qquad \mathbf{f}_N(\boldsymbol{\mu}) = (\mathbf{W}(\boldsymbol{\mu}))^T\mathbf{f}_N(\boldsymbol{\mu}). \tag{1.40}$$

$\mathbf{W}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N}$ represents, from an algebraic viewpoint, a set of (possibly $\boldsymbol{\mu}-$dependent) functions $\{w_i(\boldsymbol{\mu})\}_{i=1}^N$ such that a test space $W_N(\boldsymbol{\mu})$ is obtained as $W_N(\boldsymbol{\mu}) = span\{w_i(\boldsymbol{\mu}),\, i = 1, \ldots, N\} \subset X_h$. As matter of fact, problem (1.39) is obtained by enforcing the projection of the FE residual evaluated at the RB solution $\mathbf{V}\mathbf{y}_N(\boldsymbol{\mu})$ onto $\mathbf{W}(\boldsymbol{\mu})$ to vanish, that is by requiring

$$(\mathbf{W}(\boldsymbol{\mu}))^T \Big(\mathbf{f}_N(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu})\mathbf{V}\mathbf{y}_N(\boldsymbol{\mu})\Big) = \mathbf{0}, \tag{1.41}$$

where the RB approximation is recovered as the linear combination of the columns of $\mathbf{V}$, that is $\mathbf{y}(\boldsymbol{\mu}) \approx \mathbf{V}\mathbf{y}_N(\boldsymbol{\mu})$. Moreover, the error between the FE solution $\mathbf{y}(\boldsymbol{\mu})$ and $\mathbf{V}\mathbf{y}_N(\boldsymbol{\mu})$ satisfies the following error estimate

$$\|\mathbf{y}(\boldsymbol{\mu}) - \mathbf{V}\mathbf{y}_N(\boldsymbol{\mu})\|_{\mathbf{X}_h} \leq \frac{1}{\beta_h(\boldsymbol{\mu})} \|\mathbf{f}(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu})\mathbf{V}\mathbf{y}_N(\boldsymbol{\mu})\|_{\mathbf{X}_h^{-1}},$$

where at the right hand side appear the $\mathbf{X}_h^{-1}$–norm of the FE residual and the stability factor $\beta_h(\boldsymbol{\mu})$ introduced in (1.6), computable as $\beta_h(\boldsymbol{\mu}) = \sigma_{\min}(\mathbf{X}_h^{-\frac{1}{2}}\mathbf{A}(\boldsymbol{\mu})\mathbf{X}_h^{-\frac{1}{2}})$. Should the high-fidelity solution $\mathbf{y}(\boldsymbol{\mu})$ belong to the reduced space, that is $\mathbf{y}(\boldsymbol{\mu}) = \mathbf{V}\mathbf{y}_N(\boldsymbol{\mu})$, it can be recovered as

$$\mathbf{A}_h^{-1}(\boldsymbol{\mu})\mathbf{f}(\boldsymbol{\mu}) = \mathbf{y}(\boldsymbol{\mu}) = \mathbf{V}\mathbf{y}_N(\boldsymbol{\mu}) = \mathbf{V}\mathbf{A}_N^{-1}(\boldsymbol{\mu})(\mathbf{W}(\boldsymbol{\mu}))^T\mathbf{f}(\boldsymbol{\mu}) = \mathbf{Q}_N(\boldsymbol{\mu})\mathbf{f}(\boldsymbol{\mu}).$$

We highlight that the matrix

$$\mathbf{Q}_N(\boldsymbol{\mu}) = \mathbf{V}\mathbf{A}_N^{-1}(\boldsymbol{\mu})(\mathbf{W}(\boldsymbol{\mu}))^T \tag{1.42}$$

represents a low-rank solver which mimics the effect of $\mathbf{A}^{-1}(\boldsymbol{\mu})$ on the RB subspace spanned by the columns of $\mathbf{V}$. This observation will be relevant in Chapter 2 when building our MSRB preconditioner. Furthermore, we remark that problem (1.39) can generally be solved inexpensively, usually with direct methods, since $N \ll N_h$.

To build the matrix $\mathbf{V}$, both greedy algorithms [Prud'homme et al., 2002, Prud'homme et al., 2002, Buffa et al., 2012, Hesthaven et al., 2014] or POD, [Berkooz et al., 1993, Volkwein, 2013], can be used. With the latter, we start by computing n_s high-fidelity solutions $\{\mathbf{y}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ (called snapshots) corresponding to selected parameter values $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$. POD then aims at compressing the snapshots data by finding the best N-dimensional subspace, with $N \leq n_s$, that approximates the space $X_{n_s} = span\{\mathbf{y}(\boldsymbol{\mu}_i),\, i = 1, \ldots, n_s\}$. This is pursued by performing a singular value decomposition of the snapshot matrix $\mathbf{S} = [\mathbf{y}(\boldsymbol{\mu}_1),\, \mathbf{y}(\boldsymbol{\mu}_2),\, \ldots,\, \mathbf{y}(\boldsymbol{\mu}_{n_s})]$, resulting in the factorization

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

where $\mathbf{U} \in \mathbb{R}^{N_h \times N_h}$, $\mathbf{Z} \in \mathbb{R}^{n_s \times n_s}$ and $\Sigma \in \mathbb{R}^{N_h \times n_s}$, containing the singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{n_s} \geq 0$ and such that $\Sigma_{ii} = \sigma_i$, $i = 1, \ldots n_s$, $\Sigma_{ij} = 0$, $i \neq j$. Then, the first N columns of the matrix $\mathbf{U}$, form an orthonormal basis of a N-dimensional subspace of $\mathbb{R}^{N_h}$ and $\mathbf{V} = \mathbf{U}(:, 1:N)$. Such a construction ensures that, among all possible N-dimensional subspaces of X_{n_s}, V_N is the best N-dimensional approximation subspace, as it minimizes the projection error of the snapshots in the Euclidean norm.

This result can be stated in a general (and algebraic) form by considering any matrix-induced norm and the SVD decomposition of $\mathbf{X}_h^{1/2}\mathbf{S}$ as follows.

Proposition 1.3.1. *Let* $\mathcal{V}_N = \{\mathbf{W} \in \mathbb{R}^{N_h \times N} : \mathbf{W}^T\mathbf{X}_h\mathbf{W} = \mathbf{I}_N\}$ *be the set of all* N*-dimensional* $\mathbf{X}_h$*-orthonormal bases. Then*

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

In other words, the POD method allows to compute the space of dimension N, that minimizes the $\mathbf{X}_h$-projection error of the snapshots in the $\mathbf{X}_h$-norm. Typically, in the RB method for second-order elliptic PDEs, $\mathbf{X}_h$ encodes the $H^1(\Omega)$ scalar product on the space X_h, that is, $(\mathbf{X}_h)_{ij} = (\phi_j^x, \phi_i^x)_{H^1(\Omega)}$, $i, j = 1, \dots, N_h$.

From a practical standpoint, POD is performed by solving an eigenvalue problem associated to the correlation matrix $\mathbf{C} = \mathbf{S}^T\mathbf{X}_h\mathbf{S}$, whose eigenvalues directly provide the singular values squared σ_i^2, $i = 1, \dots, n_s$. Through the eigenvectors $\mathbf{w}_i$, $i = 1, \dots, n_s$ of $\mathbf{C}$ one can build a $\mathbf{X}_h$-orthonormal basis $\{\boldsymbol{\xi}_i\}_{i=1}^{n_s}$ of the snapshots subspace X_{n_s} by setting

$$\boldsymbol{\xi}_i = \frac{1}{\sigma_i}\mathbf{S}\mathbf{w}_i, \qquad i = 1, \dots, n_s. \tag{1.43}$$

Finally, the matrix $\mathbf{V}$ is built by selecting the first N basis functions $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_N$. The complete POD algorithm is reported in Algorithm 1. According to Proposition 1.3.1, constructing the RB space with the first N eigenvectors yields a relative approximation

Algorithm 1 POD

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

accuracy on the snapshots equal to

$$\delta_{RB}^2 = \sum_{i=N+1}^{n_s} \sigma_i^2 \Big/ \sum_{i=1}^{n_s} \sigma_i^2.$$

Then, if we aim at building a RB space relying on POD we can follow two approaches:

- POD($\mathbf{S}, \mathbf{X}_h, \delta_{RB}$): given a target accuracy δ_{RB}, we choose the first $N = N(\delta_{RB})$ columns of $\mathbf{U}$ as basis for the RB space V_N, where N is such that $\sum_{i=1}^{N} \sigma_i^2 \Big/ \sum_{i=1}^{n_s} \sigma_i^2 \geq 1 - \delta_{RB}^2$;

- POD($\mathbf{S}, \mathbf{X}_h, N$): given a fixed dimension $N > 0$, we select the first N vectors, leading to an approximation accuracy on the snapshots of $\delta_{RB} = \delta_{RB}(N)$.

Depending on the reducibility of the problem at hand, the relation between N and δ_{RB} can significantly vary, ranging from a few (order of 10) to a hundreds or thousands RB functions. We refer, e.g., to the test case in Section 2.4.1, where the number N of RB functions to reach the same tolerance δ_{RB} changes significantly by modifying the parameter space $\mathcal{D}$. We refer to [Kunisch and Volkwein, 2002a, Rowley, 2006, Kerschen et al., 2005, Quarteroni et al., 2016a] for further details and references about POD.

There are essentially two practical choices for $\mathbf{W}(\boldsymbol{\mu})$: taking $\mathbf{W}(\boldsymbol{\mu}) = \mathbf{V}$ for any $\boldsymbol{\mu}$ leads to a Galerkin-RB (G-RB) formulation particularly suited for coercive elliptic problems, since the RB matrix automatically inherits the positive definiteness of $\mathbf{A}(\boldsymbol{\mu})$. A G-RB approximation does not directly lead to the well-posedness of RB saddle-point problems, which can however be obtained with $\mathbf{W}(\boldsymbol{\mu}) = \mathbf{X}_h^{-1}\mathbf{A}(\boldsymbol{\mu})\mathbf{V}$. This choice corresponds to a Least-Squares RB (LSRB) formulation; a more in-depth discussion is carried out in Chapter 3, where an algebraic variant of the latter LSRB method is proposed for the parametrized Stokes equations.

1.3.2 Assembling the RB problem: hyper-reduction techniques

A crucial assumption that allows to speed up the RB method is made by requiring that $\mathbf{A}_h(\boldsymbol{\mu})$ and $\mathbf{f}(\boldsymbol{\mu})$ depend affinely on the parameter $\boldsymbol{\mu}$, i.e. that they can be expressed as

$$\mathbf{A}_h(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu})\mathbf{A}_h^q, \qquad \mathbf{f}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \Theta_f^q(\boldsymbol{\mu})\mathbf{f}_h^q, \tag{1.44}$$

where $\Theta_a^q : \mathcal{D} \to \mathbb{R}$, $q = 1, \dots, Q_a$ and $\Theta_f^q : \mathcal{D} \to \mathbb{R}$, $q = 1, \dots, Q_f$ are $\boldsymbol{\mu}$-dependent functions, while the matrices $\mathbf{A}_h^q \in \mathbb{R}^{N_h \times N_h}$ and the vectors $\mathbf{f}_h^q \in \mathbb{R}^{N_h}$ are $\boldsymbol{\mu}$-independent.

If assumptions (1.44) are met, then the RB algebraic structures can be obtained as

$$\mathbf{A}_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu})\mathbf{V}^T\mathbf{A}_h^q\mathbf{V} = \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu})\mathbf{A}_N^q \tag{1.45}$$

$$\mathbf{f}_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \Theta_g^f(\boldsymbol{\mu})\mathbf{V}^T\mathbf{f}^q = \sum_{q=1}^{Q_f} \Theta_f^q(\boldsymbol{\mu})\mathbf{f}_N^q \tag{1.46}$$

in the G-RB case, and as

$$\begin{aligned}\mathbf{A}_N(\boldsymbol{\mu}) &= \sum_{q_1,q_2=1}^{Q_a} \Theta_a^{q_1}(\boldsymbol{\mu})\Theta_a^{q_2}(\boldsymbol{\mu})\mathbf{V}^T(\mathbf{A}_h^{q_1})^T\mathbf{X}_h^{-1}\mathbf{A}_h^{q_2}\mathbf{V} \qquad (1.47)\\ &= \sum_{q_1,q_2=1}^{Q_a} \Theta_a^{q_1}(\boldsymbol{\mu})\Theta_a^{q_2}(\boldsymbol{\mu})\mathbf{A}_N^{q_1,q_2}\\ \mathbf{f}_N(\boldsymbol{\mu}) &= \sum_{q_1=1}^{Q_a}\sum_{q_2=1}^{Q_f} \Theta_a^{q_1}(\boldsymbol{\mu})\Theta_f^{q}(\boldsymbol{\mu})\mathbf{V}^T(\mathbf{A}_h^{q_1})^T\mathbf{X}_h^{-1}\mathbf{f}^{q_2} \qquad (1.48)\\ &= \sum_{q_1=1}^{Q_a}\sum_{q_2=1}^{Q_f} \Theta_a^{q_1}(\boldsymbol{\mu})\Theta_f^{q_2}(\boldsymbol{\mu})\mathbf{f}_N^{q_1,q_2}\end{aligned}$$

in the LSRB case. The matrices $\mathbf{A}_N^q \in \mathbb{R}^{N\times N}$, $q = 1,\dots,Q_a$ and $\mathbf{A}_N^{q_1,q_2} \in \mathbb{R}^{N\times N}$, $q_1, q_2 = 1,\dots,Q_a$ and the vectors $\mathbf{f}_N^q \in \mathbb{R}^N$, $q = 1,\dots,Q_f$, and $\mathbf{f}_N^{q_1,q_2} \in \mathbb{R}^N$, $q_1 = 1,\dots,Q_f$, $q_2 = 1,\dots,Q_a$ can be precomputed and stored during the offline phase. During the online phase, only the sums in (1.45)-(1.46) must be calculated to assemble the RB problem. Notice that the construction of $\mathbf{A}_N(\boldsymbol{\mu})$ and $\mathbf{f}_N(\boldsymbol{\mu})$ in (1.45)-(1.46) depends linearly on the number of affine terms Q_a and Q_f, while it is quadratic in (1.47)-(1.48).

When building a RB approximation, it is essential to assume that the affine dependence on $\boldsymbol{\mu}$ of the FE arrays is satisfied, that is (1.44) are verified, in order to achieve a full independence of the assembling of the RB arrays from the size N_h of the high-fidelity problem. However, in almost every problem of applied interest, the dependence of the PDE on the parameter $\boldsymbol{\mu}$ is generally nonaffine, therefore an affine representation of $\mathbf{A}(\boldsymbol{\mu})$ and $\mathbf{f}(\boldsymbol{\mu})$ cannot be computed. Several methods have been designed to overcome this bottleneck and compute an approximated affine decomposition of the FE matrix and right hand side. If the assumptions (1.44) do not hold, such affine parametric dependence can indeed be recovered by using the empirical interpolation method (EIM) and its discrete variants discrete EIM (DEIM) and matrix DEIM (MDEIM), see [Barrault et al., 2004, Chaturantabut and Sorensen, 2010, Negri et al., 2015a], which lead to the so called *hyper-reduced* problem, where both state reduction and system reduction are employed. In the following we go through these approximation techniques, since they will be extensively employed in this thesis.

(Discrete) Empirical Interpolation Method

The first approach towards this goal has been empirical interpolation method (EIM) proposed in nonaffine problems [Barrault et al., 2004]; it has been further applied to general nonaffine and nonlinear problems in [Grepl et al., 2007], in [Manzoni et al., 2012b] for shape optimization and [Drohmann et al., 2012a] for operator interpolation. Despite its main application refers to RB methods, EIM represents a general procedure for Lagrangian interpolation of sets with small Kolmogorov n-width, hierarchically and adaptively constructing a set basis functions tailored to the problem at hand, see [Maday et al., 2009]. Given a nonlinear function $\mathbf{f}(\tau) : \tau \in \mathrm{T} \subset \mathbb{R}^p \to \mathbf{f}(\tau)$, EIM builds an *approximated* affine representation of $\mathbf{f}(\tau)$ in a low-dimensional subspace spanned by M basis functions which are encoded in a ($\tau-$independent) matrix $\mathbf{\Phi} = [\phi_1, \dots, \phi_M] \in \mathbb{R}^{N_h \times M}$. Once $\mathbf{\Phi}$ has been constructed, the approximation vector $\mathbf{f}_M(\tau)$ is such that

$$\mathbf{f}(\tau) \approx \mathbf{f}_M(\tau) = \mathbf{\Phi}\boldsymbol{\theta}(\tau), \tag{1.49}$$

where $\boldsymbol{\theta}(\tau) \in \mathbb{R}^M$ is a $\tau-$dependent vector containing suitable interpolation coefficients. Matrix $\mathbf{\Phi}$ is constructed by selecting a set of snapshots $\{\mathbf{f}(\tau_i)\}_{i=1}^{n_s}$ corresponding to the parameter values τ_i, $i = 1, \dots, n_s$. Then, a greedy algorithm is employed to build $\mathbf{\Phi}$: the function which is worst approximated by the current basis is suitably scaled and shifted to build the new basis function. Given a new parameter τ, an interpolation problem is solved to compute the coefficients $\boldsymbol{\theta}(\tau)$. More specifically, given the set $\mathcal{I} \subset \{1, \dots, N_h\}$, $|\mathcal{I}| = M$, of interpolating indexes iteratively chosen by the greedy procedure, $\boldsymbol{\theta}(\tau)$ corresponding to $\mathbf{f}(\tau)$ is computed by solving the following problem

$$\mathbf{\Phi}_{\mathcal{I}}\boldsymbol{\theta}(\tau) = \mathbf{f}_{\mathcal{I}}(\tau), \tag{1.50}$$

where $\mathbf{\Phi}_{\mathcal{I}} \in \mathbb{R}^{M \times M}$ and $\mathbf{f}_{\mathcal{I}}(\tau) \in \mathbb{R}^M$ correspond to the restrictions of $\mathbf{\Phi}$ and $\mathbf{f}(\tau)$ to the rows identified by the indeces $\mathcal{I}$. As a matter of fact, we have

$$\mathbf{f}_M(\tau) = \mathbf{\Phi}\mathbf{\Phi}_{\mathcal{I}}^{-1}\mathbf{f}_{\mathcal{I}}(\tau).$$

More recently a "discrete" variant of EIM, the Discrete Empirical Interpolation Method (DEIM), has been proposed in [Chaturantabut and Sorensen, 2010] for the approximation of nonlinear and nonaffine terms. The main difference between the two algorithms lies in the way the basis $\mathbf{\Phi}$ is constructed: DEIM computes at first a set of training snapshots $\{\mathbf{f}(\tau_i)\}_{i=1}^{n_s}$ and then builds $\mathbf{\Phi}$ with POD, so that an orthonormal basis is constructed at once. On the other hand, EIM employs a greedy procedure which adaptively builds the basis $\mathbf{\Phi}$, as explained above. The choice of the interpolating points and the interpolation problem are the same in the two algorithms, as explained above.

In this work we will rely on DEIM to build an approximated affine decomposition of nonaffine vectors, hence we report the corresponding procedure in Algorithm 2, where a basis is constructed by providing a set of training parameters $\{\tau_i\}_{i=1}^{n_s}$ and a tolerance

Algorithm 2 Discrete Empirical Interpolation Method

```
 1: procedure DEIM($\{\tau_i\}_{i=1}^{n_s}, \delta_{\text{DEIM}}$)
 2:     Compute snapshots $\{\mathbf{f}(\tau_i)\}_{i=1}^{n_s}$ and set $\mathbf{\Lambda} = [\mathbf{f}(\tau_1), \dots, \mathbf{f}(\tau_{n_s})]$
 3:     $[\phi_1, \dots, \phi_M] = \text{POD}(\mathbf{\Lambda}, \delta_{\text{DEIM}})$
 4:     $i = \arg\max_{\{1,\dots,N_h\}} = |\phi_1|$
 5:     $\mathbf{\Phi} = \phi$, $\mathcal{I} = \{i\}$
 6:     for $k = 2 : M$ do
 7:         $\mathbf{r} = \phi_k - \mathbf{\Phi}\mathbf{\Phi}_{\mathcal{I}}^{-1}(\phi_k)_{\mathcal{I}}$
 8:         $i = \arg\max_{\{1,\dots,N_h\}} = |\mathbf{r}|$
 9:         $\mathcal{I} \leftarrow \mathcal{I} \cup \{i\}$
10:         $\mathbf{\Phi} \leftarrow [\mathbf{\Phi}, \phi_k]$
11:     end for
12: end procedure
13: Output: $\mathbf{\Phi}$, $\mathcal{I}$
```

δ_{DEIM} (used in the POD at line 3). As alternative, we remark that one could directly provide the matrix $\mathbf{\Lambda}$ at the place of $\{\tau_i\}_{i=1}^{n_s}$, in this case step 2 of the procedure is skipped.

As concerns the error between the vector $\mathbf{f}(\tau)$ and its approximation $\mathbf{f}_M(\tau)$ when DEIM is employed, the following error estimate holds

$$\|\mathbf{f}(\tau) - \mathbf{f}_M(\tau)\|_2 \leq \|\mathbf{\Phi}_{\mathcal{I}}^{-1}\|_2 \|(\mathbf{I} - \mathbf{\Phi}\mathbf{\Phi}^T)\mathbf{f}(\tau)\|_2, \tag{1.51}$$

where the second term on the right hand side can be approximated with the first discarded singular value of POD algorithm, that is,

$$\|(\mathbf{I} - \mathbf{\Phi}\mathbf{\Phi}^T)\mathbf{f}(\tau)\|_2 \approx \sigma_{M+1}. \tag{1.52}$$

Notice that approximation (1.52) holds for any $\tau \in \mathrm{T}$ if a proper sampling of the parameter space has been performed, leading to a projection error comparable to the one obtained on the training snapshots, see [Chaturantabut and Sorensen, 2010] for the derivation of (1.52).

Matrix (Discrete) Empirical Interpolation Method

When dealing with nonaffinely parametrized PDEs, EIM and DEIM are used to approximate the nonaffine and nonlinear functions and terms which appear in the variational formulation, typically representing the parametrized physical data or geometrical mapping of the problem at hand. However, when dealing with complex problem, the parametrization can be rather nasty, and force to intrusively work on the variational problem to come up with an approximated affine decomposition. This is especially the case when an affine dependence for the FE matrix must be recovered, see e.g. [Negri, 2015], Section

3.3.2 for some examples. To overcome this bottleneck, a matrix version of DEIM, the so called Matrix-DEIM proposed in [Negri et al., 2015a], represents a convenient black-box option to affinely approximate a sparse FE matrix.

Given a $\tau-$dependent matrix $\mathbf{K}(\tau) \in \mathbb{R}^{N_h \times N_h}$, we seek its approximation $\mathbf{K}_M(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ such that

$$\mathbf{K}(\tau) \approx \mathbf{K}_M(\boldsymbol{\mu}) = \sum_{q=1}^{M} \tilde{\Theta}^q(\tau)\mathbf{K}_q, \tag{1.53}$$

where $\tilde{\Theta}^q(\tau) : \mathrm{T} \to \mathbb{R}$, $q = 1, \dots, M$, are $\tau-$dependent coefficients and $\mathbf{K}_q \in \mathbb{R}^{N_h \times N_h}$, $q = 1, \dots, M$, are $\tau-$independent. Approximation (1.53) can be obtained by MDEIM by considering the vectorized representation $\mathbf{k}(\tau) = \mathrm{vec}(\mathbf{K}(\tau)) \in \mathbb{R}^{N_h^2}$. Then, an affine approximation $\mathbf{k}_M(\tau) \in \mathbb{R}^{N_h^2}$ of $\mathbf{k}(\tau)$ can be found by applying the DEIM algorithm, that is,

$$\mathbf{k}(\tau) \approx \mathbf{k}_M(\tau) = \boldsymbol{\Phi}\boldsymbol{\theta}(\tau), \tag{1.54}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{N_h^2 \times M}$ and $\boldsymbol{\theta}(\tau) \in \mathbb{R}^M$ are the corresponding basis and interpolation coefficients vector, respectively. Finally, by reversing the vec operation one can obtain the approximated matrix $\mathbf{K}_M(\boldsymbol{\mu})$.

The MDEIM algorithm can be implemented by relying on the DEIM Algorithm 2, where vectorized matrix snapshots are used instead of the vector snapshots. The bases are then constructed by employing one of the following approaches:

1. providing a set of training parameters $\{\tau_i\}_{i=1}^{n_s}$ and employing the matrix assembler to build the corresponding matrix snapshots at step 2, that is

   $$\boldsymbol{\Phi} = \mathrm{MDEIM}(\{\tau_i\}_{i=1}^{n_s}, \delta_{\mathrm{MDEIM}}); \tag{1.55}$$

2. as for DEIM, directly sampling the procedure with a set of vectorized matrices $\boldsymbol{\Lambda}$, that is,

   $$\boldsymbol{\Phi} = \mathrm{MDEIM}(\boldsymbol{\Lambda}, \delta_{\mathrm{MDEIM}}); \tag{1.56}$$

   in this case, step 2 of Algorithm 2 is skipped.

We will mostly take advantage of this latter option within this work.

Remark 1.3.1. *Notice that in the FE context, the actual implementation suitably exploits the sparse format which is used to store the FE matrices, together with the assumption that the sparsity pattern of the matrix does not change for the range of parameters considered. Consequently, instead of working with vectors with dimension N_h^2, n_z-dimensional vectors* *are employed, where n_z, typically much lower than N_h^2, is the number of nonzero entries of the matrix. Furthermore, the computation of the approximated matrix $\mathbf{K}_M(\boldsymbol{\mu})$ requires the evaluation of $\mathbf{k}_{\mathcal{I}}(\tau) \in \mathbb{R}^M$, i.e. the restriction of $\mathbf{k}(\tau)$ to the elements identified by the indeces $\mathcal{I}$. These indeces refer to precise entries of the FE matrix $\mathbf{K}(\tau)$, which need to be evaluated. In the FE context, this operation is performed during the online phase by restricting the assembly of $\mathbf{K}(\tau)$ to those elements which provide a nonzero contribution to the entries identified by $\mathcal{I}$. This procedure can be efficiently performed when using the FE method, thanks to the local support of the FE basis functions, see [Negri et al., 2015a, Negri, 2015] for further details.*

The error between $\mathbf{K}$ and $\mathbf{K}_M(\boldsymbol{\mu})$ can be bounded similarly to (1.51) as

$$\begin{aligned}\|\mathbf{K} - \mathbf{K}_M(\boldsymbol{\mu})\|_2 &\leq \|\mathbf{K} - \mathbf{K}_M(\boldsymbol{\mu})\|_F \\ &= \|\mathbf{k}(\tau) - \mathbf{k}_M(\tau)\|_2 \leq \|\boldsymbol{\Phi}_{\mathcal{I}}^{-1}\|_2 \|(\mathbf{I} - \boldsymbol{\Phi}\boldsymbol{\Phi}^T)\mathbf{k}(\tau)\|_2.\end{aligned} \tag{1.57}$$

The last term on the right hand side can be again approximated by the last singular value computed by the POD, that is,

$$\|(\mathbf{I} - \boldsymbol{\Phi}\boldsymbol{\Phi}^T)\mathbf{k}(\tau)\|_2 \approx \sigma_{M+1}. \tag{1.58}$$

(M)DEIM in the context of RB methods

To circumvent the problems of nonaffinity (D)EIM and MDEIM are employed to recover an approximate affine decomposition of the FE arrays in the RB context. When such techniques are employed, the relations (1.44) are satisfied up to a certain tolerance provided to the corresponding algorithms, coming up to a decomposition of the following form

$$\mathbf{A}(\boldsymbol{\mu}) \approx \widetilde{\mathbf{A}}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \tilde{\Theta}_a^q(\boldsymbol{\mu})\mathbf{A}^q, \qquad \mathbf{f}(\boldsymbol{\mu}) \approx \widetilde{\mathbf{f}}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \tilde{\Theta}_f^q(\boldsymbol{\mu})\mathbf{f}^q, \tag{1.59}$$

where Q_a and Q_f are the number of selected basis. The coefficients $\tilde{\Theta}_a^q : \mathcal{D} \to \mathbb{R}\, q = 1, \ldots, Q_a$ (resp. $\tilde{\Theta}_f^q : \mathcal{D} \to \mathbb{R}$, $q = 1, \ldots, Q_f$) are computed by solving the interpolation problems (1.50), while the matrices $\mathbf{A}^q \in \mathbb{R}^{N_h \times N_h}\, q = 1, \ldots, Q_a$ and the vectors $\mathbf{f}^q \in \mathbb{R}^{N_h}\, q = 1, \ldots, Q_f$ are $\boldsymbol{\mu}-$ independent. We will suppose the matrix $\widetilde{\mathbf{A}}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ to be non singular for any $\boldsymbol{\mu} \in \mathcal{D}$, that is

$$\tilde{\beta}_h(\boldsymbol{\mu}) = \sigma_{\min}(\mathbf{X}_h^{-\frac{1}{2}} \widetilde{\mathbf{A}}(\boldsymbol{\mu}) \mathbf{X}_h^{-\frac{1}{2}}) > 0, \tag{1.60}$$

which holds if $M \to \infty$, thanks to the approximation of the singular values guaranteed by the Weyl-Mirsky theorem (see, e.g., [Stewart and Sun, 1990]). The approximated affinity property (1.59) can be used to approximate the RB arrays as follows

$$\mathbf{A}_N(\boldsymbol{\mu}) \approx \widetilde{\mathbf{A}}_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \tilde{\Theta}_a^q(\boldsymbol{\mu})\mathbf{V}^T\mathbf{A}^q\mathbf{V} = \sum_{q=1}^{Q_a} \tilde{\Theta}_a^q(\boldsymbol{\mu})\mathbf{A}_N^q, \tag{1.61}$$

$$\mathbf{f}_N(\boldsymbol{\mu}) \approx \widetilde{\mathbf{f}}_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \widetilde{\Theta}_f^q(\boldsymbol{\mu})\mathbf{V}^T\mathbf{f}^q = \sum_{q=1}^{Q_f} \widetilde{\Theta}_f^q(\boldsymbol{\mu})\mathbf{f}_N^q. \tag{1.62}$$

The RB problem (1.39) is then substituted by the following hyper-reduced problem

$$\widetilde{\mathbf{A}}_N(\boldsymbol{\mu})\widetilde{\mathbf{y}}_N(\boldsymbol{\mu}) = \widetilde{\mathbf{f}}_N(\boldsymbol{\mu}), \tag{1.63}$$

and the matrices $\mathbf{A}_N^q$, $q = 1, \dots, Q_a$ and vectors $\mathbf{f}_N^q$, $q = 1, \dots, Q_f$ can be computed and stored once for all during the offline phase, and only the sum in (1.61)-(1.62) must be carried out during the online phase. The affinely approximated RB problem (1.63) corresponds to the RB approximation obtained by projecting the following FE system

$$\widetilde{\mathbf{A}}(\boldsymbol{\mu})\widetilde{\mathbf{y}}(\boldsymbol{\mu}) = \widetilde{\mathbf{f}}(\boldsymbol{\mu}), \tag{1.64}$$

where $\widetilde{\mathbf{A}}(\boldsymbol{\mu})$ and $\widetilde{\mathbf{f}}(\boldsymbol{\mu})$ are as in (1.61)-(1.62) and $\widetilde{\mathbf{y}}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ is the solution of the (M)DEIM-approximated system.

Remark 1.3.2. *(M)DEIM can be similarly used to treat an RB approximation with non-linear contributions, by computing an approximated affine decomposition of the nonlinear terms to cheaply assemble the RB problem during the online phase.*

1.3.3 Error bounds for hyper-reduced RB approximation

We briefly report here the error bounds between the FE high fidelity solution $\mathbf{y}(\boldsymbol{\mu})$ and the hyper-reduced RB approximation $\widetilde{\mathbf{y}}_N(\boldsymbol{\mu})$, see [Negri et al., 2015a, Negri, 2015] for further details. The following Proposition holds when computing the RB solution of problem (1.63).

Proposition 1.3.2. *If* $Q_a \in \mathbb{N}^+$ *and* $\{\tilde{\Theta}_a^q(\boldsymbol{\mu})\}_{q=1}^{Q_a}$ *are such that the matrix* $\widetilde{\mathbf{A}}(\boldsymbol{\mu})$ *is nonsingular for any* $\boldsymbol{\mu} \in \mathcal{D}$, *then the error* $\mathbf{y}(\boldsymbol{\mu}) - \widetilde{\mathbf{y}}_N(\boldsymbol{\mu})$ *can be bounded by*

$$\begin{aligned} \|\mathbf{y}(\boldsymbol{\mu}) - \mathbf{V}\widetilde{\mathbf{y}}_N(\boldsymbol{\mu})\|_{\mathbf{X}_h} \leq{}& \frac{1}{\tilde{\beta}_h(\boldsymbol{\mu})}\|\widetilde{\mathbf{A}}(\boldsymbol{\mu})\mathbf{V}\widetilde{\mathbf{y}}_N(\boldsymbol{\mu}) - \widetilde{\mathbf{f}}_N(\boldsymbol{\mu})\|_{\mathbf{X}_h^{-1}} \\ &+ \frac{1}{\beta_h(\boldsymbol{\mu})}\Big(\|\mathbf{f}(\boldsymbol{\mu}) - \widetilde{\mathbf{f}}(\boldsymbol{\mu})\|_{\mathbf{X}_h^{-1}} + \|\mathbf{A}(\boldsymbol{\mu}) - \widetilde{\mathbf{A}}(\boldsymbol{\mu})\|_{\mathbf{X}_h, \mathbf{X}_h^{-1}}\|\widetilde{\mathbf{y}}(\boldsymbol{\mu})\|_{\mathbf{X}_h}\Big). \end{aligned} \tag{1.65}$$

Error (1.65) is not suited for a proper offline-online decomposition, since it assumes to compute the solution $\widetilde{\mathbf{y}}(\boldsymbol{\mu})$ of the FE problem (1.64). A more usable estimate is given by the following result.

Proposition 1.3.3. *Under the assumptions of Proposition 1.3.2, the following estimate holds:*

$$\|\mathbf{y}(\boldsymbol{\mu}) - \mathbf{V}\widetilde{\mathbf{y}}_N(\boldsymbol{\mu})\| \leq \Delta_N(\boldsymbol{\mu}) + \Delta_M(\boldsymbol{\mu}), \tag{1.66}$$

where

$$\Delta_N(\boldsymbol{\mu}) = \frac{1}{\beta_h(\boldsymbol{\mu})}\|\widetilde{\mathbf{A}}(\boldsymbol{\mu})\mathbf{V}\widetilde{\mathbf{y}}_N(\boldsymbol{\mu}) - \widetilde{\mathbf{f}}_N(\boldsymbol{\mu})\|_{\mathbf{X}_h^{-1}} \tag{1.67}$$

$$\Delta_M(\boldsymbol{\mu}) = \frac{1}{\beta_h(\boldsymbol{\mu})}\Big(\|\mathbf{f}(\boldsymbol{\mu}) - \widetilde{\mathbf{f}}(\boldsymbol{\mu})\|_{\mathbf{X}_h^{-1}} + \|\mathbf{A}(\boldsymbol{\mu}) - \widetilde{\mathbf{A}}(\boldsymbol{\mu})\|_{\mathbf{X}_h,\mathbf{X}_h^{-1}}\|\mathbf{V}\widetilde{\mathbf{y}}_N(\boldsymbol{\mu})\|_{\mathbf{X}_h}\Big). \tag{1.68}$$

The computation of terms (1.67)-(1.68) does not involve the solution of any additional FE problem, however it still depends on the FE dimension N_h, due to the error of the affine approximation of the FE matrix and right hand side in (1.68). An efficient computation of bound (1.66) can be performed by employing estimates (1.51)-(1.57), together with a proper sampling of the parameter space, which ensures (1.52) and (1.58) to hold, leading to the computable estimator

$$\Delta_M(\boldsymbol{\mu}) \approx \frac{1}{\beta_h(\boldsymbol{\mu})}\Big(c_1\|(\boldsymbol{\Phi}_{\mathcal{I}}^f)^{-1}\|_2\sigma_{Q_f+1}^f + c_2\|(\boldsymbol{\Phi}_{\mathcal{I}}^a)^{-1}\|_2\sigma_{Q_a+1}^a\|\mathbf{V}\widetilde{\mathbf{y}}_N(\boldsymbol{\mu})\|_{\mathbf{X}_h}\Big), \tag{1.69}$$

where $\sigma_{Q_f+1}^f$ and $\sigma_{Q_a+1}^a$ are the first discarded eigenvalues of (M)DEIM for $\mathbf{f}(\boldsymbol{\mu})$ and $\mathbf{A}(\boldsymbol{\mu})$, respectively, and $c_1 = \|\mathbf{X}_h^{-\frac{1}{2}}\|_2$ and $c_2 = \|\mathbf{X}_h^{-1}\|_2$.

Remark 1.3.3. *Bounds* (1.65) *and* (1.66) *suggest that both an accurate state reduction, that is a rich enough RB space V_N, and system approximation, that is a precise affine representation of the FE matrix and right hand side, are required to obtain an accurate RB approximation.*

Remark 1.3.4. *The number of affine terms to obtain a target accuracy highly depends on the problem at hand, and if this is too large, it may impact on the efficiency of the RB solver. See e.g. Section 2.4.1, where the impact of the DEIM affine expansion is taken into account by varying the parametrization of the considered problem.*

1.3.4 Galerkin RB methods for time-dependent problems

We can generalize to the case of time-dependent problems the construction of a RB method, including the use of hyper-reduction techniques for cheaply assembling the resulting reduced system. For ease of presentation, we consider the sequence of linear systems (1.22) obtained by the BDF discretization of the algebraic dynamical system (1.17), for which we assume the right hand side to be expressed as $\mathbf{f}(t;\boldsymbol{\mu}) = g_f(t)\widetilde{\mathbf{f}}(\boldsymbol{\mu})$.

Even if the specific case of BDF schemes is treated, the strategies reported in the following can be easily extended to other implicit time-discretization technique.

Similarly to the steady case, at any time-step $n = 0, \dots, N_t$ the RB approximation is expressed as a linear combination of N RB functions collected in a matrix $\mathbf{V} \in \mathbb{R}^{N_h \times N}$

$$\mathbf{y}^n \approx \mathbf{V}\mathbf{y}_N^n(\boldsymbol{\mu}). \tag{1.70}$$

If we plug in this approximation in (1.22) and perform a Galerkin projection by enforcing the FE residual to vanish when it is evaluated at the RB approximation, we obtain a sequence of RB linear systems, to be solved for $n = 1, \dots, N_t - 1$,

$$\left(\frac{\alpha_1}{\Delta t}\mathbf{M}_N(\boldsymbol{\mu}) + \mathbf{A}_N(\boldsymbol{\mu})\right)\mathbf{y}_N^{n+1}(\boldsymbol{\mu}) = g_f(t_{n+1})\mathbf{f}_N(\boldsymbol{\mu}) + \frac{\alpha_1}{\Delta t}\mathbf{M}_N(\boldsymbol{\mu})\mathbf{y}_N^{n,\sigma_1}(\boldsymbol{\mu}), \tag{1.71}$$

with $\mathbf{y}_N^0(\boldsymbol{\mu}) \in \mathbb{R}^N$ obtained by projecting $\mathbf{y}_0$ onto $\mathbf{V}$ and $\mathbf{A}_N(\boldsymbol{\mu})$ defined as in (1.40); $\mathbf{f}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ and $\mathbf{M}_N(\boldsymbol{\mu}) \in \mathbb{R}^{N\times N}$ are instead obtained as

$$\mathbf{f}_N(\boldsymbol{\mu}) = \mathbf{V}^T\widetilde{\mathbf{f}}(\boldsymbol{\mu}) \qquad \mathbf{M}_N(\boldsymbol{\mu}) = \mathbf{V}^T\mathbf{M}(\boldsymbol{\mu})\mathbf{V}. \tag{1.72}$$

Remark 1.3.5. *A Petrov-Galerkin formulation, which is obtained by enforcing the projection of the FE residual onto the space spanned by the columns of a matrix $\mathbf{W}(\boldsymbol{\mu})$ (in general different from $\mathbf{V}$) to vanish, can also be used, see e.g. [Carlberg et al., 2013]. However, here we limit ourselves to recall the Galerkin case, since for unsteady problems it is the only one considered in this thesis.*

The construction of the ROM, that is the offline phase, is carried out in two main steps.

1. *State reduction:* a double POD approach, similar to the one used in [Paul-Dubois-Taine and Amsallem, 2015, Negri et al., 2015a, Negri, 2015], can be used to build the RB projection matrix $\mathbf{V}$. Given the parameter values $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$, n_s sets of finite elements functions $\{\mathbf{y}_N^n(\boldsymbol{\mu}_1)\}_{n=1}^{N_t}$, ..., $\{\mathbf{y}_N^n(\boldsymbol{\mu}_{n_s})\}_{n=1}^{N_t}$ are computed, and a local POD basis-in-time is constructed

   $$\mathbf{S}^i = [\mathbf{y}^1(\boldsymbol{\mu}_i), \dots, \mathbf{y}^{N_t}(\boldsymbol{\mu}_i)], \qquad \mathbf{V}^i = \mathrm{POD}\Big(\mathbf{S}^i, \mathbf{X}_h, \varepsilon_t\Big), \qquad i = 1, \dots, n_s,$$

   where ε_t is the tolerance employed to build the local basis. Subsequently, all these basis functions are collected and a final POD is used to construct the final RB space

   $$\mathbf{S} = [\mathbf{V}^1, \dots, \mathbf{V}^{n_s}], \qquad \mathbf{V} = \mathrm{POD}\Big(\mathbf{S}, \mathbf{X}_h, \varepsilon_\mu\Big).$$

   The tolerance ε_μ is employed in this second stage, usually chosen such that $\varepsilon_\mu \geq \varepsilon_t$.

2. *System approximation:* also in the unsteady case, it is advisable to rely on an

affine decomposition, equivalent to (1.45)-(1.46), of the RB arrays to enhance the efficiency during the online phase. If $\mathbf{M}(\boldsymbol{\mu})$, $\mathbf{A}(\boldsymbol{\mu})$, and $\widetilde{\mathbf{f}}(\boldsymbol{\mu})$ do not feature such affine property, an approximate affine decomposition of those arrays can be recovered by using (separately) MDEIM on $\mathbf{M}(\boldsymbol{\mu})$ and $\mathbf{A}(\boldsymbol{\mu})$ and DEIM on $\widetilde{\mathbf{f}}(\boldsymbol{\mu})$. The arrays $\{\mathbf{M}^q\}_{q=1}^{Q_m}$, $\{\mathbf{A}^q\}_{q=1}^{Q_a}$ and $\{\widetilde{\mathbf{f}}\}_{q=1}^{Q_f}$ are computed before constructing the RB projection matrix $\mathbf{V}$ and then used to construct the corresponding RB affine decompositions $\{\mathbf{M}_N^q\}_{q=1}^{Q_m}$, $\{\mathbf{A}_N^q\}_{q=1}^{Q_a}$ and $\{\mathbf{f}^q\}_{q=1}^{Q_f}$ such that

$$\mathbf{M}_N(\boldsymbol{\mu}) \approx \widetilde{\mathbf{M}}_N(\boldsymbol{\mu}) \sum_{q=1}^{Q_m} \tilde{\Theta}_m^q(\boldsymbol{\mu}) \mathbf{M}_N^q \tag{1.73}$$

and $\mathbf{A}_N(\boldsymbol{\mu}) \approx \widetilde{\mathbf{A}}_N(\boldsymbol{\mu})$, $\mathbf{f}_N(\boldsymbol{\mu}) \approx \widetilde{\mathbf{f}}_N(\boldsymbol{\mu})$ as in (1.61)-(1.62).

As usual, also a greedy or a POD-greedy approach represent viable options, see e.g. [Grepl and Patera, 2005, Drohmann et al., 2012b].

During the online phase, (1.22) is replaced by

$$\left(\frac{\alpha_1}{\Delta t}\widetilde{\mathbf{M}}_N(\boldsymbol{\mu}) + \widetilde{\mathbf{A}}_N(\boldsymbol{\mu})\right)\mathbf{y}_N^{n+1}(\boldsymbol{\mu}) = g_f(t_{n+1})\widetilde{\mathbf{f}}_N(\boldsymbol{\mu}) + \frac{\alpha_1}{\Delta t}\widetilde{\mathbf{M}}_N(\boldsymbol{\mu})\mathbf{y}_N^{n,\sigma_1}(\boldsymbol{\mu}), \tag{1.74}$$

for which suitable a posteriori error bounds, similar to the one recalled in 1.3.3 for steady problems, are verified, see [Negri, 2015, Negri et al., 2015a].

This framework to construct a ROM for unsteady parametrized problems is considered and extended in Chapter 4 to deal with unsteady parametrized nonlinear equations, such as the Navier-Stokes equations in parameter-dependent domains.

1.4 Towards RB coarse operators for preconditioning parametrized PDEs

As explained in Section 1.3, the RB method is a powerful tool to tackle parametrized systems, building an accurate approximation of the FE solution in a possibly very competitive computational cost. However, the RB approximation can sometimes show severe bottlenecks, depending on the problem at hand, leading to the deterioration of its efficiency and accuracy for different reasons: *i)* the number N of basis functions increases, leading to a costly computation to invert the RB matrix $\mathbf{A}_N(\boldsymbol{\mu})$, *ii)* an approximated affine decomposition of the FE matrix and/or right hand side features too many terms, yielding an extensive assembling phase, *iii*) the presence of different physical regimes when dealing with time-dependent problems.

Recently, preconditioning techniques for the RB matrix $\mathbf{A}_N(\boldsymbol{\mu})$ have been proposed, e.g. in [Elman and Forstall, 2015], to overcome issue *i*) by iteratively solving the RB problem (1.39); this demonstrates to be a convenient option when the RB dimension N is too large to use direct methods. Our aim is instead to build preconditioners for the high-fidelity FE problem (1.7), and not for the RB system (1.39). To tackle issue *ii*), one can try to improve the computed (D)EIM interpolation basis to (and without changing its dimension) by optimizing the location of the interpolation points, an example of this procedure can be found, for instance, in [Sargsyan et al., 2016]. As issue *iii*) concerns, local (in time) basis functions can be built by splitting the time interval in macro slabs, easing the handling of different time regimes of the solution, see e.g. [Peherstorfer et al., 2014, Amsallem and Haasdonk, 2016, Pagani et al., 2017].

On the other hand, as we have seen from (1.42), the RB method provides us with a ready-to-use low rank solver which seeks an approximated solution in a subspace of the FE space, in a similar fashion of coarse grid operators in domain decomposition or multilevel methods. In a two level approach, where it is combined with a fine grid operator, the role of the RB approximation is to provide a coarse correction able to boost the convergence rate of the iterative method applied to the high-fidelity FE problem, by correcting the scales of the error which are not treated by the fine operator. In this perspective, the computational load entailed by the presence of nonaffine or nonlinear terms is no longer an issue, as in the case of the standard RB method. Moreover, a coarse correction must not be necessarily accurate, and the number of RB functions (or approximated affine terms) characterizing the RB low-rank solver can be relatively low so that the overall efficiency is not compromised.

In the following chapters we develop this idea and tailor it to parametrized problems of engineering interest. We start by considering (nonaffine) linear elliptic and parabolic PDEs in Chapter 2; we turn to linear saddle-point problems, with focus on Stokes equations, in Chapter 3. Finally, we focus on unsteady nonlinear PDEs, with specific interest in the Navier-Stokes equations in Chapter 4.

2 Multi space RB preconditioners for parametrized PDEs

In this chapter we introduce our multi space reduced basis preconditioning strategy, focusing on parametrized elliptic and parabolic problems. We consider the algebraic FE system arising from a parametrized advection-diffusion equation, and show how to exploit a RB coarse component in a two-level preconditioning setting. We first apply the preconditioner to the Richardson method because of the simple structure of this latter. We deal with both affine and nonaffine problems, using in the latter case an approximated RB coarse operator which internally exploits MDEIM algorithm for its construction. Then, we apply the preconditioner to the FGMRES method and report its performances on problems characterized by a large dimension. We refer to [Dal Santo et al., 2018a, Dal Santo et al., 2017a] for additional details and numerical tests.

2.1 Preconditioning the Richardson method with a RB coarse operator

In this first section, we construct a preconditioned Richardson method which employs the low-rank RB solver (1.42) as coarse component. Towards this goal, let us consider a variational problem characterized by a continuous and coercive bilinear form $a(\cdot, \cdot; \boldsymbol{\mu})$ and the corresponding parametrized high-fidelity linear system which arises from the use of the FE element method

$$\mathbf{A}(\boldsymbol{\mu})\mathbf{u}(\boldsymbol{\mu}) = \mathbf{f}(\boldsymbol{\mu}). \tag{2.1}$$

We remark that such a linear system appears, for instance, when considering problem (1.10). The FE stiffness matrix $\mathbf{A}(\boldsymbol{\mu})$ in (2.1) is positive definite thanks to the coercivity of the bilinear form $a(\cdot, \cdot; \boldsymbol{\mu})$, weakly coercive problems will be specifically addressed in Chapter 3.

Given an initial guess $\mathbf{u}^{(0)} = \mathbf{u}^{(0)}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$, let us consider two matrices $\mathbf{Q}_1 = \mathbf{Q}_1(\boldsymbol{\mu})$,

$\mathbf{Q}_2 = \mathbf{Q}_2(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$, a multiplicative Richardson iteration for system (2.1) can be expressed as

$$\begin{cases} \mathbf{u}^{(k-1/2)}(\boldsymbol{\mu}) & = \mathbf{u}^{(k-1)}(\boldsymbol{\mu}) + \mathbf{Q}_1(\boldsymbol{\mu})\mathbf{r}^{(k-1)}(\boldsymbol{\mu}), \\ \mathbf{u}^{(k)}(\boldsymbol{\mu}) & = \mathbf{u}^{(k-1/2)}(\boldsymbol{\mu}) + \mathbf{Q}_2(\boldsymbol{\mu})\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu}), \qquad k = 1, 2, \dots \end{cases} \tag{2.2}$$

where $\mathbf{u}^{(k)} = \mathbf{u}^{(k)}(\boldsymbol{\mu})$ is the $\boldsymbol{\mu}-$dependent iterate at the step k, and $\mathbf{r}^{(k)} = \mathbf{r}^{(k)}(\boldsymbol{\mu})$ is the corresponding high-fidelity residual of system (2.1), defined as

$$\mathbf{r}^{(k)}(\boldsymbol{\mu}) = \mathbf{f}(\boldsymbol{\mu}) - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{u}^{(k)}(\boldsymbol{\mu}), \qquad k = 1, 2, \dots.$$

Equations (2.2) can be equivalently formulated as a single iteration

$$\mathbf{u}^{(k)}(\boldsymbol{\mu}) = \mathbf{u}^{(k-1)}(\boldsymbol{\mu}) + \mathbf{Q}(\boldsymbol{\mu})\mathbf{r}^{(k-1)}(\boldsymbol{\mu}), \qquad k = 1, 2, \dots, \tag{2.3}$$

where $\mathbf{Q}(\boldsymbol{\mu})$ in (2.3) is defined as

$$\mathbf{Q}(\boldsymbol{\mu}) = \mathbf{Q}_1(\boldsymbol{\mu}) + \mathbf{Q}_2(\boldsymbol{\mu}) - \mathbf{Q}_2(\boldsymbol{\mu})\mathbf{A}_h(\boldsymbol{\mu})\mathbf{Q}_1(\boldsymbol{\mu}).$$

If $\mathbf{Q}(\boldsymbol{\mu})$ is non singular, (2.3) can be regarded as a stationary Richardson iteration, with acceleration parameter equal to 1, for the preconditioned system

$$\mathbf{Q}(\boldsymbol{\mu})\mathbf{A}_h(\boldsymbol{\mu})\mathbf{u}(\boldsymbol{\mu}) = \mathbf{Q}(\boldsymbol{\mu})\mathbf{f}(\boldsymbol{\mu}),$$

where the preconditioner is $\mathbf{Q}^{-1}(\boldsymbol{\mu})$.

A two stage approach as the one in (2.2) is exploited in two level domain decomposition strategies which rely on a coarse (or low-rank) component derived from a coarse FE discretization. We want to apply this idea in the following by considering, at the place of a coarse grid operator, a RB low-rank solver as coarse component, and towards this goal we set

$$\mathbf{Q}_1(\boldsymbol{\mu}) = \mathbf{P}^{-1}(\boldsymbol{\mu}), \qquad \mathbf{Q}_2(\boldsymbol{\mu}) = \mathbf{V}\mathbf{A}_N^{-1}(\boldsymbol{\mu})\mathbf{V}^T, \tag{2.4}$$

where $\mathbf{P}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ is a nonsingular matrix playing the role of fine preconditioner (to be chosen among existing preconditioners) and $\mathbf{V}\mathbf{A}_N^{-1}(\boldsymbol{\mu})\mathbf{V}^T$ is the RB coarse component. As explained in Section 1.3.1, the matrix $\mathbf{V}\mathbf{A}_N^{-1}(\boldsymbol{\mu})\mathbf{V}^T$ mimics the inverse of $\mathbf{A}(\boldsymbol{\mu})$ on the RB subspace.

However, we have experienced that numerically the convergence rate of (2.2) with $\mathbf{Q}_2(\boldsymbol{\mu}) = \mathbf{V}\mathbf{A}_N^{-1}(\boldsymbol{\mu})\mathbf{V}^T$ is the same as the one obtained by setting $\mathbf{Q}_2(\boldsymbol{\mu}) = 0$ (i.e. just using $\mathbf{P}(\boldsymbol{\mu})$ as preconditioner); as a matter of fact, the RB coarse component in ineffective. This behavior can be ascribed to the fact that the application of $\mathbf{Q}_2(\boldsymbol{\mu})$ to the residual

$\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu})$

$$\mathbf{Q}_2(\boldsymbol{\mu})\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu}) = \mathbf{V}\mathbf{A}_N^{-1}(\boldsymbol{\mu})\mathbf{V}^T\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu})$$

can be reinterpreted as computing an approximation of the solution $\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu})$ of the error equation

$$\mathbf{A}_h(\boldsymbol{\mu})\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu}) = \mathbf{r}^{(k-1/2)}(\boldsymbol{\mu}), \tag{2.5}$$

through the RB method, where $\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu}) = \mathbf{u}(\boldsymbol{\mu}) - \mathbf{u}^{(k-1/2)}(\boldsymbol{\mu})$. In other words, by computing $\mathbf{V}\mathbf{A}_N^{-1}(\boldsymbol{\mu})\mathbf{V}^T\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu})$, we are implicitly seeking an approximation of $\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu})$ in the RB space V_N, that is, expressed as a linear combination of basis functions obtained from snapshots of the high-fidelity problem (2.1). The main issue related with this approach is that the employed ROM (i.e. the RB space V_N) is tailored only for equation (2.1), while here we are trying to use it to solve approximately equation (2.5), which features the same stiffness matrix $\mathbf{A}_h(\boldsymbol{\mu})$ as (2.1) but a different right hand side. Therefore, the space V_N is not well suited to approximate the solution of problem (2.5), yielding a very poor numerical approximation of the error, as confirmed by numerical experiments.

2.2 Multi space RB preconditioners for elliptic PDEs

The ineffective procedure outlined above suggests that a different RB space, more suitable to approximate the error $\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu})$, should be employed at step k of the Richardson method. The corresponding preconditioner is first analyzed in the (simple) context of Richardson iterations; subsequently, we extend its construction to the FGMRES method in Section 2.2.5.

2.2.1 Multi space RB preconditioners for the Richardson method

At each step k of the Richardson method we introduce a new RB space that is trained on equation (2.5), and where a better approximation of $\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu})$ can be found. Since the error highly depends on the iterate k, it makes sense to introduce a different RB space V_{N_k} at every iteration k, generated by high-fidelity solutions of problem (2.5), that is

$$V_{N_k} = span\Big\{\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu}_j)\Big\}_{j=1}^{N_k}. \tag{2.6}$$

Here $\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu}_j) = \mathbf{u}(\boldsymbol{\mu}_j) - \mathbf{u}^{(k-1/2)}(\boldsymbol{\mu}_j)$, $j = 1, \dots, N_k$ denote the errors at the $(k-1/2)$-th iteration, computed for (properly chosen) instances of the parameters $\boldsymbol{\mu}_j$, $j = 1, \dots, N_k$, where N_k is the dimension of the RB space used at iteration k, possibly changing with k. Following the standard G-RB method introduced in Section 1.3.1, we can construct the matrices

$$\mathbf{V}_k = [\boldsymbol{\xi}_1^k|\dots|\boldsymbol{\xi}_N^k], \qquad \mathbf{A}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k^T\mathbf{A}_h(\boldsymbol{\mu})\mathbf{V}_k, \qquad k = 0, 1, \dots \tag{2.7}$$

where $\{\boldsymbol{\xi}_j^k\}_{j=1}^{N_k}$ denotes an orthonormalized basis for V_{N_k}, and write the MSRB-preconditioned Richardson iterations as

$$\begin{cases} \mathbf{u}^{(k-1/2)}(\boldsymbol{\mu}) & = \mathbf{u}^{(k-1)}(\boldsymbol{\mu}) + \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{r}^{(k-1)}(\boldsymbol{\mu}) \\ \mathbf{u}^{(k)}(\boldsymbol{\mu}) & = \mathbf{u}^{(k-1/2)}(\boldsymbol{\mu}) + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu}), \qquad k = 0, 1, \dots, \end{cases} \tag{2.8}$$

now setting $\mathbf{Q}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k\mathbf{A}_{N_k}^{-1}(\boldsymbol{\mu})\mathbf{V}_k^T$. The formulation (2.8) leads to

$$\mathbf{u}^{(k)}(\boldsymbol{\mu}) = \mathbf{u}^{(k-1)}(\boldsymbol{\mu}) + \mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{r}^{(k-1)}(\boldsymbol{\mu}), \qquad k = 1, 2, \dots, \tag{2.9}$$

where the matrix $\mathbf{Q}_{\mathrm{MSRB},k} = \mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ (replacing $\mathbf{Q}(\boldsymbol{\mu})$ in (2.3)) is now

$$\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu}) = \mathbf{P}^{-1}(\boldsymbol{\mu}) + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\Big(\mathbf{I}_{N_h} - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big), \tag{2.10}$$

and can be regarded as a multiplicative combination of $\mathbf{P}^{-1}(\boldsymbol{\mu})$ and $\mathbf{Q}_{N_k}(\boldsymbol{\mu})$.

Given the error $\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu})$, the corresponding RB approximation onto V_{N_k} is defined by $\mathbf{e}_{N_k}^{(k-1/2)}(\boldsymbol{\mu}) \in \mathbb{R}^{N_k}$ such that

$$\mathbf{A}_{N_k}(\boldsymbol{\mu})\mathbf{e}_{N_k}^{(k-1/2)}(\boldsymbol{\mu}) = \mathbf{V}_k^T\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu}). \tag{2.11}$$

As a matter of fact, problem (2.11) is the RB approximation of problem (2.5) where $\mathbf{V}_k$ is employed as projection matrix. The FE representation $\mathbf{V}_k\mathbf{e}_{N_k}^{(k-1/2)}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ of $\mathbf{e}_{N_k}^{(k-1/2)}(\boldsymbol{\mu})$ can be expressed as

$$\mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu}) = \mathbf{V}_k\mathbf{A}_{N_k}^{-1}(\boldsymbol{\mu})\mathbf{V}_k^T\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu}) = \mathbf{V}_k\mathbf{e}_{N_k}^{(k-1/2)}(\boldsymbol{\mu}), \tag{2.12}$$

and is computed in the second step of (2.8).

Remark 2.2.1. *Here and in the following we will assume that, for any $k = 0, 1, \dots$, the columns of the matrix $\mathbf{V}_k$ are two by two linearly independent. This assumption ensures that the RB matrix $\mathbf{A}_{N_k}(\boldsymbol{\mu})$ is positive definite, since it is obtained by Galerkin projection from the positive definite matrix $\mathbf{A}(\boldsymbol{\mu})$. As a consequence, its inverse $\mathbf{A}_{N_k}^{-1}(\boldsymbol{\mu})$ exists.*

A natural initial guess for iterations (2.8) is the (standard) RB approximation $\mathbf{u}^{(0)}(\boldsymbol{\mu}) = \mathbf{V}_0\mathbf{A}_{N_0}^{-1}(\boldsymbol{\mu})\mathbf{V}_0^T\mathbf{f}(\boldsymbol{\mu})$, which can be obtained by setting $V_{N_0} = V_N$, i.e. the first RB space is the one provided by the standard RB method. As a matter of fact, the subsequent spaces V_{N_k}, $k \geq 1$, aim at damping those components of the error that have not been cured by the previous RB iterations and cannot be addressed by the application of $\mathbf{P}(\boldsymbol{\mu})$; they are therefore directly constructed using the error equation (2.5), whose solution can be expressed for any $\boldsymbol{\mu} \in \mathcal{D}$ as follows:

$$\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu}) = \mathbf{u}(\boldsymbol{\mu}) - \mathbf{u}^{(k-1/2)}(\boldsymbol{\mu}) = \mathbf{e}^{(k-1)}(\boldsymbol{\mu}) - \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{r}^{(k-1)}(\boldsymbol{\mu}). \tag{2.13}$$

Therefore, if the solution $\mathbf{u}(\boldsymbol{\mu})$ is known, the corresponding error $\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu})$ can be computed avoiding the solution of the linear system (2.5).

Remark 2.2.2. *When building the RB approximation for* (2.5)*, a Galerkin projection has been preferred to a more general Petrov Galerkin one, since, for elliptic coercive problems as the advection-diffusion PDE* (1.10)*, the G-RB method straightforwardly provides a well-posed RB approximation. However, this is not trivial when dealing with weakly coercive problems, as the case where $\nabla \cdot \vec{b}$ is positive or a saddle-point system is considered; this issue is taken into account in the following Chapter 3, when dealing with linear saddle-point problems.*

2.2.2 Nonsingularity of MSRB preconditioners

We show in this section that the matrix $\mathbf{Q}_{\text{MSRB},k}(\boldsymbol{\mu})$. Given a subspace $W \subset \mathbb{R}^{N_h}$ such that $dim(W) = M$ and a basis $\{\mathbf{w}_j\}_{j=1}^M$ such that $W = span\{\mathbf{w}_j, j = 1, \dots, M\}$, we denote by $W^\perp$ the orthogonal complement of W and by $\mathbf{W} \in \mathbb{R}^{N_h \times M}$, $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_M]$, the matrix of basis vectors. Moreover, given any nonsingular matrix $\mathbf{B} \in \mathbb{R}^{N_h \times N_h}$, we define the following spaces

$$\begin{aligned}\mathbf{B}W &= \Big\{\mathbf{x} \in \mathbb{R}^{N_h} : \ \mathbf{B}^{-1}\mathbf{x} \in W \ \ \Big\} = \Big\{\mathbf{x} \in \mathbb{R}^{N_h} : \ \ \mathbf{x} = \mathbf{B}\mathbf{z}, \, \mathbf{z} \in W\Big\},\\ \mathbf{B}W^\perp &= \Big\{\mathbf{x} \in \mathbb{R}^{N_h} : \ \mathbf{B}^{-1}\mathbf{x} \in W^\perp\Big\} = \Big\{\mathbf{x} \in \mathbb{R}^{N_h} : \ \ \mathbf{x} = \mathbf{B}\mathbf{z}, \, \mathbf{z} \in W^\perp\Big\}.\end{aligned}$$

We remark that $\mathbb{R}^{N_h} = \mathbf{B}W \oplus \mathbf{B}W^\perp$, because of the nonsingularity of $\mathbf{B}$.

Lemma 2.2.1. *Let W be a M-dimensional subspace of $\mathbb{R}^{N_h}$, $\{\mathbf{w}_j\}_{j=1}^M$ a basis thereof and $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_M] \in \mathbb{R}^{N_h \times M}$. Moreover, let $\mathbf{B}$ be a nonsingular $N_h \times N_h$ matrix and assume that $\mathbf{W}^T\mathbf{B}\mathbf{W}$ is nonsingular. Then the following implication holds:*

$$\mathbf{x} \in \mathbf{B}W \quad \textit{and} \quad \mathbf{W}^T\mathbf{x} = \mathbf{0} \quad \Rightarrow \quad \mathbf{x} = \mathbf{0}. \tag{2.14}$$

Proof. We take $\mathbf{x} \in \mathbf{B}W$ such that $\mathbf{W}^T\mathbf{x} = \mathbf{0}$ and show that it must be $\mathbf{x} = \mathbf{0}$. By definition of $\mathbf{B}W$, $\mathbf{B}^{-1}\mathbf{x} = \mathbf{W}\mathbf{z}_M$ for some $\mathbf{z}_M \in \mathbb{R}^M$. Thanks to the nonsingularity of $\mathbf{B}$, we obtain

$$\mathbf{0} = \mathbf{W}^T\mathbf{x} = \mathbf{W}^T\mathbf{B}\mathbf{B}^{-1}\mathbf{x} = \mathbf{W}^T\mathbf{B}\mathbf{W}\mathbf{z}_M.$$

As $\mathbf{W}^T\mathbf{B}\mathbf{W} \in \mathbb{R}^{M \times M}$ is invertible, $\mathbf{z}_M = \mathbf{0}$. Finally, we have

$$\mathbf{0} = \mathbf{W}\mathbf{z}_M = \mathbf{B}^{-1}\mathbf{x},$$

which implies $\mathbf{x} = \mathbf{0}$ thanks to the nonsingularity of $\mathbf{B}$. $\square$

In the following we employ Lemma 3.5.1 with $W = V_{N_k}$, $\mathbf{B} = \mathbf{P}(\boldsymbol{\mu})$, $\mathbf{W} = \mathbf{V}_k$ in order to prove that $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ is nonsingular. To this aim, we define

$$V_{N_k}^{\mathbf{P}/\!/} = \Big\{\mathbf{x} \in \mathbb{R}^{N_h} : \quad \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x} \in V_{N_k}\Big\}, \quad V_{N_k}^{\mathbf{P}\perp} = \Big\{\mathbf{x} \in \mathbb{R}^{N_h} : \quad \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x} \in V_{N_k}^{\perp}\Big\}.$$

Theorem 2.2.1. *For any $\boldsymbol{\mu} \in \mathcal{D}$, assume that $\mathbf{P}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ is a nonsingular matrix such that the matrix $\mathbf{V}_k^T\mathbf{P}(\boldsymbol{\mu})\mathbf{V}_k$ is nonsingular. Then the matrix $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ is nonsingular.*

Proof. We want to prove that if $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{x} = \mathbf{0}$, then it must be $\mathbf{x} = \mathbf{0}$. Since any $\mathbf{x} \in \mathbb{R}^{N_h}$ can be expressed as $\mathbf{x} = \mathbf{x}_{/\!/} + \mathbf{x}_{\perp}$, where $\mathbf{x}_{/\!/} \in V_{N_k}^{\mathbf{P}/\!/}$, $\mathbf{x}_{\perp} \in V_{N_k}^{\mathbf{P}\perp}$, we first compute the result of the application of $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ on $\mathbf{x}_{/\!/}$:

$$\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{x}_{/\!/} = \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{/\!/} + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\Big(\mathbf{I}_{N_h} - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{x}_{/\!/}$$

Being $\mathbf{x}_{/\!/} \in V_{N_k}^{\mathbf{P}/\!/}$, we can write $\mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{/\!/} = \mathbf{V}_k\mathbf{z}_N(\boldsymbol{\mu})$ for some $\mathbf{z}_{N_k}(\boldsymbol{\mu}) \in \mathbb{R}^{N_k}$, yielding

$$\begin{aligned}\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{x}_{/\!/} = {} & \mathbf{V}_k\mathbf{z}_N(\boldsymbol{\mu}) + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{x}_{/\!/} \\ & - \mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{A}_h(\boldsymbol{\mu})\mathbf{V}_k\mathbf{z}_N(\boldsymbol{\mu}) = \mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{x}_{/\!/},\end{aligned} \tag{2.15}$$

since $\mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{A}_h(\boldsymbol{\mu})\mathbf{V}_k\mathbf{z}_N = \mathbf{V}_k\mathbf{A}_{N_k}^{-1}(\boldsymbol{\mu})\mathbf{V}_k^T\mathbf{A}_h(\boldsymbol{\mu})\mathbf{V}_k\mathbf{z}_N = \mathbf{V}_k\mathbf{z}_N$. As of the component $\mathbf{x}_{\perp}$, we have

$$\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{x}_{\perp} = \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{\perp} + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\Big(\mathbf{I}_{N_h} - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{x}_{\perp},$$

which leads to

$$\begin{aligned}\mathbf{0} = \mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{x} = {} & \mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{x}_{/\!/} + \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{\perp} \\ & + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\Big(\mathbf{I}_{N_h} - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{x}_{\perp}.\end{aligned} \tag{2.16}$$

By rewriting equation (2.16) as follows

$$\mathbf{Q}_{N_k}(\boldsymbol{\mu})\Big(\mathbf{x}_{/\!/} + \mathbf{x}_{\perp} + \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{\perp}\Big) = -\mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{\perp}, \tag{2.17}$$

we can notice that the left hand side is an element of the space V_{N_k}, whereas the right hand side is an element of its orthogonal complement $V_{N_k}^{\perp}$, so that the only way these two elements are equal is when they are both zero. Being $\mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{\perp} = \mathbf{0}$, the nonsingularity of $\mathbf{P}(\boldsymbol{\mu})$ yields $\mathbf{x}_{\perp} = \mathbf{0}$, allowing us to rewrite equation (2.17) as

$$\mathbf{0} = \mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{x}_{/\!/} = \mathbf{V}_k\mathbf{A}_{N_k}^{-1}(\boldsymbol{\mu})\mathbf{V}_k^T\mathbf{x}_{/\!/}. \tag{2.18}$$

The columns of $\mathbf{V}_k$ being linearly independent, equation (2.18) yields

$$\mathbf{0} = \mathbf{A}_{N_k}^{-1}(\boldsymbol{\mu})\mathbf{V}_k^T\mathbf{x}_{/\!/}, \tag{2.19}$$

which, thanks to the non singularity of the RB matrix $\mathbf{A}_{N_k}(\boldsymbol{\mu})$, cf. Remark (2.2.1), implies

$$\mathbf{V}_k^T\mathbf{x}_{/\!/} = \mathbf{0}. \tag{2.20}$$

Finally, by applying Lemma 3.5.1 with $W = V_{N_k}$, $\mathbf{W} = \mathbf{V}_k$ and $\mathbf{B} = \mathbf{P}(\boldsymbol{\mu})$, we obtain that $\mathbf{x}_{/\!/} = \mathbf{0}$, and thus the thesis. □

Now, since the matrix $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ is invertible, we can define the MSRB preconditioner as

$$\mathbf{P}_{\mathrm{MSRB},k}(\boldsymbol{\mu}) = \mathbf{Q}_{\mathrm{MSRB},k}^{-1}(\boldsymbol{\mu}). \tag{2.21}$$

Remark 2.2.3. *The assumption that the matrix $\mathbf{V}_k^T\mathbf{P}(\boldsymbol{\mu})\mathbf{V}_k$ is nonsingular is fairly mild. For example, it is satisfied for any matrix $\mathbf{P}(\boldsymbol{\mu})$ such that $\mathbf{x}^T\mathbf{P}(\boldsymbol{\mu})\mathbf{x} \neq 0$ for any $\mathbf{x} \neq \mathbf{0}$. This is indeed the case for classical preconditioners like Jacobi, Gauss-Seidel or Additive Schwarz preconditioners.*

2.2.3 Convergence results for the MSRB-preconditioned Richardson method

In this section we prove a priori estimates of the error and the residual decay for the Richardson method (2.8). For ease of notation, hereon we omit the $\boldsymbol{\mu}-$dependence and denote by $\mathbf{I}_{N_h}$ the identity $N_h \times N_h$ matrix.

Proposition 2.2.1. *For any vector norm $\|\cdot\|$, let the spaces V_{N_k} $k = 1, \dots, L$ satisfy the following relation*

$$\|\mathbf{e}^{(k-1/2)} - \mathbf{V}_k\mathbf{e}_{N_k}^{(k-1/2)}\| \leq \delta_k\|\mathbf{e}^{(k-1/2)}\|, \qquad k = 1, \dots, L \qquad \forall\boldsymbol{\mu} \in \mathcal{D}, \tag{2.22}$$

for given tolerances δ_k, $k = 1, \dots, L$. Moreover, let the assumption of Theorem 2.2.1 be satisfied. Then the following estimate holds on the error generated at each iteration k of the Richardson method

$$\|\mathbf{e}^{(k)}\| \leq C^k\tilde{\delta}_k\|\mathbf{e}^{(0)}\|, \qquad k = 1, \dots, L, \qquad \forall\boldsymbol{\mu} \in \mathcal{D}, \tag{2.23}$$

with $C = \left\|\mathbf{I}_{N_h} - \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{A}_h(\boldsymbol{\mu})\right\|$ and $\tilde{\delta}_k = \prod_{j=1}^k \delta_j$.

Proof. We consider equations (2.8). The error $\mathbf{e}^{(k)} = \mathbf{u} - \mathbf{u}^{(k)}$ at iteration k can be computed as

$$\mathbf{e}^{(k)} = \Big(\mathbf{I}_{N_h} - \mathbf{Q}_{N_k}\mathbf{A}_h\Big)\mathbf{e}^{(k-1/2)} = \mathbf{e}^{(k-1/2)} - \mathbf{V}_k\mathbf{e}_{N_k}^{(k-1/2)},$$

where the equation (2.12) has been used. Then

$$\begin{aligned}\Big\|\mathbf{e}^{(k)}\Big\| &= \Big\|\Big(\mathbf{I}_{N_h} - \mathbf{Q}_{N_k}\mathbf{A}_h\Big)\mathbf{e}^{(k-1/2)}\Big\| \leq \delta_k\Big\|\mathbf{e}^{(k-1/2)}\Big\| \\ &= \delta_k\Big\|\Big(\mathbf{I}_{N_h} - \mathbf{P}^{-1}\mathbf{A}_h\Big)\mathbf{e}^{(k-1)}\Big\| \leq \delta_k\Big\|\mathbf{I}_{N_h} - \mathbf{P}^{-1}\mathbf{A}_h\Big\|\Big\|\mathbf{e}^{(k-1)}\Big\|.\end{aligned}$$

By proceeding recursively we obtain (2.23). □

A similar result holds for the residuals of the Richardson method.

Proposition 2.2.2. *For any vector norm* $\|\cdot\|$, *let the spaces* V_{N_k} $k = 1, \dots, L$ *satisfy the following relation*

$$\|\mathbf{r}^{(k-1/2)} - \mathbf{A}_h\mathbf{V}_k\mathbf{e}_{N_k}^{(k-1/2)}\| \leq \delta_k\|\mathbf{r}^{(k-1/2)}\|, \qquad k = 1, \dots, L \qquad \forall\boldsymbol{\mu} \in \mathcal{D}. \tag{2.24}$$

and given tolerances δ_k, $k = 1, \dots, L$. *Moreover, let the assumption of Theorem 2.2.1 be satisfied. Then the following estimate holds:*

$$\|\mathbf{r}^{(k)}\| \leq C^k\tilde{\delta}_k\|\mathbf{r}^{(0)}\|, \qquad k = 1, \dots, L \qquad \forall\boldsymbol{\mu} \in \mathcal{D}, \tag{2.25}$$

with $C = \Big\|\mathbf{I}_{N_h} - \mathbf{P}^{-1}\mathbf{A}_h\Big\|$ *and* $\tilde{\delta}_k = \prod_{j=1}^k \delta_j$.

Proof. We consider equations (2.8). The residual at iteration k can be computed as

$$\begin{aligned}\mathbf{r}^{(k)} &= \Big(\mathbf{I}_{N_h} - \mathbf{A}_h\mathbf{Q}_{N_k}\Big)\mathbf{r}^{(k-1/2)} = \Big(\mathbf{I}_{N_h} - \mathbf{A}_h\mathbf{V}_k\mathbf{A}_{N_k}^{-1}\mathbf{V}_k^T\Big)\mathbf{r}^{(k-1/2)} \\ &= \mathbf{r}^{(k-1/2)} - \mathbf{A}_h\mathbf{V}_k\mathbf{e}_{N_k}^{(k-1/2)}.\end{aligned} \tag{2.26}$$

Thanks to (2.24) we obtain

$$\begin{aligned}\Big\|\mathbf{r}^{(k)}\Big\| &= \Big\|\Big(\mathbf{I}_{N_h} - \mathbf{A}_h\mathbf{Q}_{N_k}\Big)\mathbf{r}^{(k-1/2)}\Big\| \leq \delta_k\Big\|\mathbf{r}^{(k-1/2)}\Big\| \\ &= \delta_k\Big\|\Big(\mathbf{I}_{N_h} - \mathbf{A}_h\mathbf{P}^{-1}\Big)\mathbf{r}^{(k-1)}\Big\| \leq \delta_k\Big\|\mathbf{I}_{N_h} - \mathbf{A}_h\mathbf{P}^{-1}\Big\|\Big\|\mathbf{r}^{(k-1)}\Big\|.\end{aligned}$$

By proceeding recursively we obtain (2.25). □

2.2.4 Dealing with nonaffine problems: (M)DEIM in the context of MSRB preconditioning methods

When using the MSRB preconditioning strategies for nonaffine problems, at each iteration we are called to solve the RB problem (2.11) for equation (2.5), which will provide us with an accurate and cheap approximation of $\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu})$. While doing so, there is no need to avoid operations whose cost is proportional to N_h for assembling the RB problem (2.11), as it would happen when using the standard RB method. When building the RB coarse operator $\mathbf{Q}_{N_k}(\boldsymbol{\mu})$ for the k-th iteration of Richardson method, the matrix $\mathbf{Q}_{N_k}(\boldsymbol{\mu})$ is not explicitly assembled, and $\mathbf{V}_k$ and $\mathbf{A}_{N_k}^{-1}(\boldsymbol{\mu})$, which is computed and stored as LU factorization of $\mathbf{A}_{N_k}(\boldsymbol{\mu})$, are applied consecutively to the right hand side of (2.5). At first, we build the RB right hand side of (2.11) by projecting $\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu})$ onto the RB space V_{N_k}; as a matter of fact, any potential affine dependence of the right hand side $\mathbf{f}(\boldsymbol{\mu})$ of (2.1), it is not exploited (therefore even if $\mathbf{f}(\boldsymbol{\mu})$ featured a nonaffine dependence, DEIM would not be required). Similarly, an affine dependence of the FE matrix is not needed *a priori* during the assembly of the RB matrix $\mathbf{A}_{N_k}(\boldsymbol{\mu})$, however a significant speedup can be achieved if $\mathbf{A}(\boldsymbol{\mu})$ verifies the affinity assumption (1.44), since, similarly to the standard RB method, it can be used to build $\mathbf{A}_{N_k}(\boldsymbol{\mu})$:

$$\begin{aligned}\mathbf{A}_{N_k}(\boldsymbol{\mu}) &= \mathbf{V}_k^T\mathbf{A}(\boldsymbol{\mu})\mathbf{V}_k \\ &= \mathbf{V}_k^T\Big(\sum_{q=1}^{Q_a}\Theta_a^q(\boldsymbol{\mu})\mathbf{A}_h^q\Big)\mathbf{V}_k = \sum_{q=1}^{Q_a}\Theta_a^q(\boldsymbol{\mu})\mathbf{V}_k^T\mathbf{A}_h^q\mathbf{V}_k = \sum_{q=1}^{Q_a}\Theta_a^q(\boldsymbol{\mu})\mathbf{A}_{N_k}^q.\end{aligned} \tag{2.27}$$

Here, the matrices $\mathbf{A}_{N_k}^q \in \mathbb{R}^{N_k\times N_k}$, $q = 1,\dots,Q_a$, are $\boldsymbol{\mu}$-independent. This allows to largely cut off the overhead of projecting $\mathbf{A}(\boldsymbol{\mu})$ onto $\mathbf{V}_k$ as in (2.7), since only the last sum in (2.27) must be carried out to assembly $\mathbf{A}_{N_k}(\boldsymbol{\mu})$.

On the other hand, if $\mathbf{A}(\boldsymbol{\mu})$ features a nonaffine parametric dependence, we rely on MDEIM to build an approximated affine one. By considering the decomposition (1.59), we have

$$\mathbf{A}_{N_k}(\boldsymbol{\mu}) \approx \widetilde{\mathbf{A}}_{N_k}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\tilde{\Theta}_a^q(\boldsymbol{\mu})\mathbf{V}_k^T\mathbf{A}^q\mathbf{V}_k = \sum_{q=1}^{Q_a}\tilde{\Theta}_a^q(\boldsymbol{\mu})\mathbf{A}_{N_k}^q, \quad k = 0, 1, \dots. \tag{2.28}$$

The matrices $\mathbf{A}_{N_k}^q$, $q = 1,\dots,Q_a$, $k = 0,\dots,L-1$ are $\boldsymbol{\mu}$-independent can be precomputed and stored offline, while only the sum in the last term of (2.28) needs to be performed when assembling $\widetilde{\mathbf{A}}_{N_k}(\boldsymbol{\mu})$. Then, the (affinely approximated) RB coarse operators for nonaffine problems are defined as

$$\mathbf{Q}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k\widetilde{\mathbf{A}}_{N_k}^{-1}(\boldsymbol{\mu})\mathbf{V}_k^T, \qquad k = 0, 1\dots. \tag{2.29}$$

Convergence results

In this section we provide a priori estimates of the error and residual decay when the coarse operators (2.29) with an approximated affine decomposition are employed as coarse components in the preconditioned Richardson method. These results essentially represent the counterpart of Proposition 2.2.1 and 2.2.2, respectively.

Proposition 2.2.3. *For any vector norm* $\|\cdot\|$, *let the spaces* V_{N_k} $k = 1, \dots, L$ *satisfy the following relation*

$$\|\mathbf{e}^{(k-1/2)} - \mathbf{V}_k \mathbf{e}_{N_k}^{(k-1/2)}\| \leq (\delta_{N_k} + \delta_M)\|\mathbf{e}^{(k-1/2)}\| \qquad k = 1, \dots, L, \qquad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{2.30}$$

for given tolerances δ_M *and* δ_{N_k}, *for* $k = 1, \dots, L$, *accounting for the state reduction and the affine approximation, respectively. Moreover, let the assumption of Theorem 2.2.1 be satisfied. Then the following estimate holds:*

$$\|\mathbf{e}^{(k)}\| \leq C^k \delta \|\mathbf{e}^{(0)}\|, \qquad k = 1, \dots, L \qquad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{2.31}$$

with $C = \left\|\mathbf{I}_{N_h} - \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{A}_h(\boldsymbol{\mu})\right\|$ *and* $\delta = \prod_{j=1}^{k}(\delta_{N_j} + \delta_M)$.

Proof. The proof is the same as the one for Proposition 2.2.1, where the role of δ_k is replaced by $\delta_{N_k} + \delta_M$. □

Similarly, a result for the residuals of the Richardson method holds.

Proposition 2.2.4. *For any vector norm* $\|\cdot\|$, *let the spaces* V_{N_k} $k = 1, \dots, L$ *satisfy the following relation*

$$\|\mathbf{r}^{(k-1/2)} - \mathbf{A}_h \mathbf{V}_k \mathbf{e}_{N_k}^{(k-1/2)}\| \leq (\delta_{N_k} + \delta_M)\|\mathbf{r}^{(k-1/2)}\| \qquad k = 1, \dots, L, \qquad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{2.32}$$

for given tolerances δ_M *and* δ_{N_k}, $k = 1, \dots, L$, *accounting for the state reduction and the affine approximation, respectively. Moreover, let the assumption of Theorem 2.2.1 be satisfied. Then the following estimate holds:*

$$\|\mathbf{r}^{(k)}\| \leq C^k \delta \|\mathbf{r}^{(0)}\|, \qquad k = 1, \dots, L \qquad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{2.33}$$

with $C = \left\|\mathbf{I}_{N_h} - \mathbf{P}^{-1}\mathbf{A}_h\right\|$ *and* $\delta = \prod_{j=1}^{k}(\delta_{N_j} + \delta_M)$.

Proof. The proof is the same as the one for Proposition 2.2.2, where the role of δ_k is replaced by $\delta_{N_k} + \delta_M$. □

In Proposition 2.2.3 and 2.2.4, the roles of δ_{N_k} and δ_M correspond to the ones of $\Delta_N(\boldsymbol{\mu})$ and $\Delta_M(\boldsymbol{\mu})$ in (1.66) for the standard RB method, for which, as highlighted in Remark

1.3.3, a precise affine representation of $\mathbf{A}(\boldsymbol{\mu})$ is necessary to reach a small target accuracy. By contrast, estimates (2.31) and (2.33) state that the final error given by the MSRB-preconditioned Richardson is provided by the combination of L iterations, each solving the error equation up to an accuracy of order $\delta_{N_k} + \delta_M$, $k = 1, 2, \dots, L$, and as a matter of fact, the affine approximation can be less accurate than the one we would use for the standard RB method. In practice, an affine decomposition of $\mathbf{A}(\boldsymbol{\mu})$ is built by plugging in MDEIM a tolerance δ_{MDEIM}, which is chosen such that δ_{MDEIM} is negligible with respect to δ_{N_k}, $k = 1, 2, \dots$ in (2.30)-(2.32).

Remark 2.2.4. *Assumptions* (2.30) *and* (2.32) *formally hold if a greedy algorithm is used to construct the RB spaces* V_{N_k} *and the affine approximation of* $\mathbf{A}(\boldsymbol{\mu})$*. By using POD and MDEIM, relations* (2.32) *are assessed numerically if a proper sampling of the parameter space* $\mathcal{D}$ *is carried out.*

2.2.5 Multi space RB preconditioners for flexible GMRES

In the previous Section, our MSRB preconditioner has been illustrated in the context of Richardson iterations, for sake of clarity. In order to use a more efficient Krylov iterative method, we opt instead for the flexible GMRES method, FGMRES [Saad, 1993], since the MSRB preconditioner changes at each iteration. Indeed, the (classical) preconditioned GMRES algorithm does not ensure convergence in the case the preconditioner changes at every iteration, while its flexible variant allows to precondition the system with an iteration-dependent operator. For ease of presentation, we report in Algorithm 3 the FGMRES algorithm taken from [Saad, 2003]. In the practical implementation, we employ the `Trilinos` software [Heroux et al., 2003], where the norm of the relative residual to be smaller than a prescribed tolerance ε_r is the stopping criterion.

Algorithm 3 Flexible GMRES (as formulated in [Saad, 2003])

```
1:  procedure FGMRES($\mathbf{A}$, $\mathbf{b}$, $\mathbf{u}_0$, $\{\mathbf{M}_k\}_k, m$)
2:      Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{u}_0$, $\beta = \|\mathbf{r}_0\|_2$, and $\mathbf{v}_1 = \mathbf{r}_0/\beta$
3:      for $k = 1, \dots, m$ do
4:          Compute $\mathbf{z}_k = \mathbf{M}_k^{-1}\mathbf{v}_k$
5:          Compute $\mathbf{w} = \mathbf{A}\mathbf{z}_k$
6:          for $j = 1, \dots, k$ do
7:              $h_{j,k} = (\mathbf{w}, \mathbf{z}_j)$
8:              $\mathbf{w} = \mathbf{w} - h_{j,k}\mathbf{v}_j$
9:          end for
10:         Compute $h_{k+1,k} = \|\mathbf{w}\|$ and $\mathbf{v}_{k+1} = \mathbf{w}/h_{k+1,k}$
11:         Define $\mathbf{Z}_m = [\mathbf{z}_1, \dots, \mathbf{z}_m]$, $\tilde{\mathbf{H}}_m = \{h_{j,k}\}_{1 \le j \le k+1;\, 1 \le k \le m}$
12:     end for
13:     Compute $\mathbf{y}_m = \arg\min_{\mathbf{y} \in \mathbb{R}^m} \|\beta\mathbf{e}_1 - \tilde{\mathbf{H}}_m\mathbf{y}\|_2$ and $\mathbf{u}_m = \mathbf{u}_0 + \mathbf{Z}_m\mathbf{y}_m$
14:     If satisfied Stop, else set $\mathbf{u}_0 \leftarrow \mathbf{u}_m$ and GoTo 2.
15: end procedure
```

In Algorithm 3, the preconditioner employed at iteration k is denoted by $\mathbf{M}_k$. Since its inverse is applied to the k-th element of the Krylov basis $\mathbf{v}_k$, we infer that $\mathbf{M}_k$ is generally used to find an approximation of $\mathbf{c}_k$, which is defined as the solution of the following problem:

$$\mathbf{A}\mathbf{c}_k = \mathbf{v}_k. \tag{2.34}$$

Indeed, if by chance $\mathbf{M}_k^{-1}\mathbf{v}_k = \mathbf{A}^{-1}\mathbf{v}_k$, FGMRES would yield the exact solution of the system. In the MSRB case, we have $\mathbf{M}_k^{-1} = \mathbf{M}_k^{-1}(\boldsymbol{\mu}) = \mathbf{Q}_{\text{MSRB},k}(\boldsymbol{\mu})$, meaning that its action on $\mathbf{v}_k(\boldsymbol{\mu})$ can be computed as

$$\begin{aligned}\mathbf{M}_k^{-1}(\boldsymbol{\mu})\mathbf{v}_k(\boldsymbol{\mu}) = \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{v}_k(\boldsymbol{\mu})& \\ + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\Big(\mathbf{I}_{N_h} - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{v}_k(\boldsymbol{\mu}).&\end{aligned} \tag{2.35}$$

To find the right problem for training the k-th RB space, we highlight that in equation (2.35) the reduced component of $\mathbf{Q}_{\text{MSRB},k}(\boldsymbol{\mu})$ is applied to $\left(\mathbf{I}_{N_h} - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\right)\mathbf{v}_k(\boldsymbol{\mu})$. In order to suitably precondition the FGMRES method, the k-th RB space must therefore be trained to solve the following problem

$$\mathbf{A}_h(\boldsymbol{\mu})\mathbf{y}_k(\boldsymbol{\mu}) = \mathbf{v}_{k+\frac{1}{2}}(\boldsymbol{\mu}), \qquad k = 1, 2, \ldots, \tag{2.36}$$

where $\mathbf{v}_{k+\frac{1}{2}}(\boldsymbol{\mu}) = (\mathbf{I}_{N_h} - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu}))\mathbf{v}_k(\boldsymbol{\mu})$. Equation (2.36) yields a RB space of the form

$$V_{N_k} = span\Big\{\mathbf{y}_k(\boldsymbol{\mu}_i)\Big\}_{i=1}^{N_k}, \qquad k = 1, 2, \ldots, \tag{2.37}$$

where $\mathbf{y}_k(\boldsymbol{\mu}_i)$ is the solution of equation (2.36) with $\boldsymbol{\mu} = \boldsymbol{\mu}_i$.

When using $\mathbf{M}_k^{-1}(\boldsymbol{\mu}) = \mathbf{Q}_{\text{MSRB},k}(\boldsymbol{\mu})$ in the preconditioning step, an approximation $\mathbf{y}_{N_k}(\boldsymbol{\mu}) \in \mathbb{R}^{N_k}$ of the solution $\mathbf{y}_k(\boldsymbol{\mu})$ of problem (2.36) can be found by solving

$$\mathbf{A}_{N_k}(\boldsymbol{\mu})\mathbf{y}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k^T\mathbf{v}_{k+\frac{1}{2}}(\boldsymbol{\mu}); \tag{2.38}$$

correspondingly, its high-fidelity representation $\mathbf{V}_k\mathbf{y}_{N_k}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ is computed as

$$\mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{v}_{k+\frac{1}{2}}(\boldsymbol{\mu}) = \mathbf{V}_k\mathbf{A}_{N_k}^{-1}(\boldsymbol{\mu})\mathbf{V}_k^T\mathbf{v}_{k+\frac{1}{2}}(\boldsymbol{\mu}) = \mathbf{V}_k\mathbf{y}_{N_k}(\boldsymbol{\mu}). \tag{2.39}$$

Following a similar argument to the one used for the Richardson method in Section 2.2.1, and exploiting the expressions of the Krylov basis given in Algorithm 3, we can find a formula which allows to compute $\mathbf{y}_k(\boldsymbol{\mu})$ without explicitly solving problem (2.36), thus playing the same role played by (2.13) in the Richardson method. The most suitable initial guess is the solution of the RB system, we therefore set $\mathbf{u}_0(\boldsymbol{\mu}) = \mathbf{V}_0\mathbf{A}_{N_0}^{-1}(\boldsymbol{\mu})\mathbf{V}_0^T\mathbf{f}(\boldsymbol{\mu})$, which yields

$$\begin{aligned}
&\mathbf{r}_0(\boldsymbol{\mu}) = \mathbf{f}(\boldsymbol{\mu}) - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{u}_0(\boldsymbol{\mu}), \qquad \beta(\boldsymbol{\mu}) = \|\mathbf{r}_0(\boldsymbol{\mu})\|_2, \\
&\mathbf{v}_1(\boldsymbol{\mu}) = \mathbf{r}_0(\boldsymbol{\mu})/\beta(\boldsymbol{\mu}).
\end{aligned} \tag{2.40}$$

Following (2.36), the first preconditioner $\mathbf{M}_1^{-1}(\boldsymbol{\mu})$ must precondition the problem

$$\mathbf{A}_h(\boldsymbol{\mu})\mathbf{y}_1(\boldsymbol{\mu}) = \Big(\mathbf{I}_{N_h} - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{v}_1(\boldsymbol{\mu}) = \frac{1}{\beta(\boldsymbol{\mu})}\Big(\mathbf{I}_{N_h} - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{r}_0(\boldsymbol{\mu}),$$

whose true high-fidelity solution $\mathbf{y}_1(\boldsymbol{\mu})$ has the following form:

$$\begin{aligned}
\mathbf{y}_1(\boldsymbol{\mu}) &= \mathbf{A}_h^{-1}(\boldsymbol{\mu})\Big(\mathbf{I}_{N_h} - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{v}_1(\boldsymbol{\mu}) \\
&= \frac{1}{\beta(\boldsymbol{\mu})}\mathbf{A}_h^{-1}(\boldsymbol{\mu})\mathbf{r}_0(\boldsymbol{\mu}) - \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{v}_1(\boldsymbol{\mu}) \\
&= \frac{1}{\beta(\boldsymbol{\mu})}\mathbf{A}_h^{-1}(\boldsymbol{\mu})\Big(\mathbf{f}(\boldsymbol{\mu}) - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{u}_0(\boldsymbol{\mu})\Big) - \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{v}_1(\boldsymbol{\mu}) \\
&= \frac{1}{\beta(\boldsymbol{\mu})}\mathbf{A}_h^{-1}(\boldsymbol{\mu})\Big(\mathbf{A}_h(\boldsymbol{\mu})\mathbf{u}(\boldsymbol{\mu}) - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{u}_0(\boldsymbol{\mu})\Big) - \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{v}_1(\boldsymbol{\mu}) \\
&= \frac{1}{\beta(\boldsymbol{\mu})}\Big(\mathbf{u}(\boldsymbol{\mu}) - \mathbf{u}_0(\boldsymbol{\mu})\Big) - \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{v}_1(\boldsymbol{\mu}).
\end{aligned}$$

We now proceed recursively, supposing to have built our preconditioner up to step k, and show how to build the $(k+1)$-th step. Following (2.36), $\mathbf{y}_{k+1}(\boldsymbol{\mu})$ must have the form

$$\mathbf{y}_{k+1}(\boldsymbol{\mu}) = \mathbf{A}_h^{-1}(\boldsymbol{\mu})\mathbf{v}_{k+1}(\boldsymbol{\mu}) - \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{v}_{k+1}(\boldsymbol{\mu}), \tag{2.41}$$

where $\mathbf{v}_{k+1}(\boldsymbol{\mu})$ is the $(k+1)$-th Krylov basis, that is (thanks to Algorithm 3)

$$\mathbf{v}_{k+1}(\boldsymbol{\mu}) = \frac{1}{h_{k+1,k}}\Big(\mathbf{A}_h(\boldsymbol{\mu})\mathbf{M}_k^{-1}(\boldsymbol{\mu})\mathbf{v}_k(\boldsymbol{\mu}) - \sum_{j=1}^{k} h_{j,k}\mathbf{v}_j(\boldsymbol{\mu})\Big) \qquad k = 1, 2, \dots,$$

thus yielding for $k = 1, 2, \dots$

$$\mathbf{y}_{k+1}(\boldsymbol{\mu}) = \frac{1}{h_{k+1,k}}\Big(\mathbf{M}_k^{-1}(\boldsymbol{\mu})\mathbf{v}_k(\boldsymbol{\mu}) - \sum_{j=1}^{k} h_{j,k}\mathbf{A}_h^{-1}(\boldsymbol{\mu})\mathbf{v}_j(\boldsymbol{\mu})\Big) - \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{v}_{k+1}(\boldsymbol{\mu}).$$

Finally, by recalling that $\mathbf{z}_k(\boldsymbol{\mu}) = \mathbf{M}_k^{-1}(\boldsymbol{\mu})\mathbf{v}_k(\boldsymbol{\mu})$, and expressing $\mathbf{A}_h^{-1}(\boldsymbol{\mu})\mathbf{v}_k(\boldsymbol{\mu}) = \mathbf{y}_k(\boldsymbol{\mu}) +$

$\mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{v}_k(\boldsymbol{\mu})$ when evaluating equation (2.41) at step k, we obtain the following relations

$$\begin{cases} \beta(\boldsymbol{\mu}) = \left\|\mathbf{f}(\boldsymbol{\mu}) - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{u}_0(\boldsymbol{\mu})\right\|_2, \\ \mathbf{y}_1(\boldsymbol{\mu}) = \dfrac{1}{\beta(\boldsymbol{\mu})}(\mathbf{u}(\boldsymbol{\mu}) - \mathbf{u}_0(\boldsymbol{\mu})) - \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{v}_1(\boldsymbol{\mu}), \\ \mathbf{y}_{k+1}(\boldsymbol{\mu}) = \dfrac{1}{h_{k+1,k}}\Big[\mathbf{z}_k(\boldsymbol{\mu}) - \displaystyle\sum_{j=1}^{k} h_{j,k}(\mathbf{y}_j(\boldsymbol{\mu}) + \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{v}_j(\boldsymbol{\mu}))\Big] \\ \qquad\qquad -\mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{v}_{k+1}(\boldsymbol{\mu}) \qquad\qquad k \geq 1. \end{cases} \tag{2.42}$$

Compared to the Richardson case, the snapshots of the k-th step depend on those obtained at all previous steps, hence requiring a higher data storage. However, FGMRES generally allows to reach convergence in a much slower number of iterations than the ones needed by the Richardson method.

2.3 Algorithmic procedures

In this section we detail the procedures required to build and use the MSRB preconditioner, by splitting the computation in an *offline* (typically expensive) and an *online* phase, where the FE problem (2.1) is solved for a new instance of $\boldsymbol{\mu}$. Propositions 2.2.1 and 2.2.2 state that the error $\mathbf{e}^{(k)}(\boldsymbol{\mu})$ and the residual $\mathbf{r}^{(k)}(\boldsymbol{\mu})$ of the Richardson method decay as the product of the tolerances $\delta_{RB,j}$, $j = 0, 1, \dots$ used to build the reduced spaces. If we employ a stopping criterion based on the relative residual for the Richardson method

- this means that, given a tolerance ϵ_r, the method reaches convergence at iteration m such that

$$\frac{\|\mathbf{r}^m(\boldsymbol{\mu})\|_2}{\|\mathbf{f}(\boldsymbol{\mu})\|_2} \leq \epsilon_r, \tag{2.43}$$

- we must build the RB spaces $V_{N_0}, \dots, V_{N_k}$, such that

$$\delta = \prod_{j=0}^{k} \delta_{RB,j} \leq \varepsilon_r. \tag{2.44}$$

In other words, we require that the combination of all RB spaces yields an error which is bounded by the target tolerance ε_r of the Richardson method.

2.3.1 Offline phase

During the offline phase, we build the structures needed for handling the application of $\mathbf{Q}_{\text{MSRB},k}(\boldsymbol{\mu})$ to any new possible instance of the parameter online, namely the RB spaces $\mathbf{V}_k$, $k = 0, 1, \dots$ and the corresponding coarse operators. In the algorithms we propose, we employ POD to build the basis for the RB spaces, whose construction is performed iteratively and according to the following steps:

1) we choose n_s parameter values $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$ and compute the snapshots $\{\mathbf{u}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ as the high-fidelity solutions of (2.1) for $\boldsymbol{\mu} = \boldsymbol{\mu}_i$, $i = 1, \ldots, n_s$;
2) following the standard RB method, we build V_{N_0} with POD on this set of snapshots, where N_0 is chosen as 5% to 20% of n_s, depending on the problem at hand;
3) we iteratively build the snapshots $\{\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ (for the Richardson case with (2.13)) or $\{\mathbf{y}_k(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$, (for the FGMRES case by relying on (2.42)) and construct the RB space for coarse operator $k = 1, 2, \ldots$ by using POD. In particular, Algorithm 1 with a tolerance $\delta_{RB,k}$ is employed for building the RB space k. We highlight that the construction of the k-th space, employing equation (2.13) or (2.42), does not require to solve any additional linear system;
4) we build an affine representation of $\mathbf{A}_{N_k}(\boldsymbol{\mu})$, $k = 0, 1, \ldots$, that is, build the matrices $\{\mathbf{A}_{N_k}^q\}_{q=1}^{Q_a}$, $k = 0, 1, \ldots$ as defined in (2.27).

In order to design our algorithm, a POD approach has been preferred to a (weak) greedy approach because of the intrinsic nonaffinity of $\mathbf{P}^{-1}(\boldsymbol{\mu})$, that appears in relation (2.13). Indeed, a greedy algorithm would build the reduced space relying on a fast evaluation of the error (or a residual-based a posteriori error bound) for a large number of offline parameters in a training set Ξ_{train}, typically computed with N_h-independent routines. On the other hand, computing the error or the residual for the equation (2.13) requires N_h-dependent operations, which would yield extremely huge offline costs for each $\boldsymbol{\mu} \in \Xi_{train}$. Relying on a POD approach makes the proposed technique also feasible in view of more involved applications (e.g. nonlinear problems) where residual-based a posteriori error bounds are not available. On the other hand, we underline that the hypothesis (2.22) of Proposition 2.2.1 holds only for a training set $\Xi_{train} \subset \mathcal{D}$ when the space V_{N_k} are constructed, for instance, relying upon a greedy algorithm with a prescribed tolerance $\delta_k = \delta_{RB,k}$ on the error and $\| \cdot \| = \| \cdot \|_{\mathbf{X}_h}$, where $\mathbf{X}_h$ is a symmetric positive definite matrix used to orthonormalize the RB functions. If we employ POD with a prescribed tolerance $\delta_{RB,k}$ on a set of training snapshots for the sake of space construction, hypothesis (2.22) does not hold, even if it is assessed from a numerical standpoint, since by solving the reduced problem relying on these reduced space provides an approximate solution $\mathbf{e}_{N_k}^{(k-1/2)}$ whose corresponding relative error $\|\mathbf{e}^{(k-1/2)} - \mathbf{V}_k \mathbf{e}_{N_k}^{(k-1/2)}\|_{\mathbf{X}_h} / \|\mathbf{e}^{(k-1/2)}\|_{\mathbf{X}_h}$ is of the order of $\delta_{RB,k}$. Similarly, the same is observed when considering the decay of residual in (2.24). Further discussions are reported for the test case I in Section 2.4.1.

Regarding the choice of the tolerances $\delta_{RB,k}$, $k = 0, 1, \ldots$, (and, consequently, of the number N_k, $k = 0, 1, \ldots$, of basis functions) for each RB space, we can follow two approaches:

- *fixed accuracy*: all the tolerances $\{\delta_{RB,k}\}_k$ are chosen equal to the same value δ_{RB}, that is $\delta_{RB,k} = \delta_{RB}$ for any k. This choice leads to RB coarse operators which provide a constant accuracy, leading the error norm to decrease at a fixed rate at each iteration. However, the RB spaces the dimension increases with k, leading to a larger computational time to assemble and solve the resulting RB system.
- *fixed dimension*: we build each RB space prescribing the same space dimension N, i.e. $N_k = N$, $k = 0, 1, \ldots$. This choice is more convenient when we are dealing with problems featuring less smooth dependence on the parameter $\boldsymbol{\mu}$, for which the slow decay of the POD singular values would require a large number of RB functions. By limiting the number of RB functions in each space, an overhead caused by a too large RB dimension is avoided, preserving the efficiency of the preconditioner.

Since we need to construct a sufficiently large number of spaces such that inequality (2.44) is satisfied, in the former approach we shall implicitly fix the number of spaces larger than $\lceil \log(\varepsilon_r)/\log(\delta_{RB})\rceil$, which however may lead to a huge number of RB functions employed at each RB space. In the latter, instead, we are not limiting the number of spaces. The detailed algorithms corresponding to these two approaches are reported in Algorithm 4 and 5, respectively. In Section 2.4 we report results for both these approaches. Here and in the following, we report the algorithms considering the FGMRES method for ease of presentation and since the it is more largely employed within this thesis; in the case the Richardson algorithm is used instead, the corresponding set of snapshots $\{\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ are computed by relying on (2.42). We refer to [Dal Santo et al., 2017a] for a more in-depth discussion on the Richardson method.

Algorithm 4 MSRB - Fixed Accuracy

```
procedure MSRB-fixedAccuracy($\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}, \varepsilon_r, \delta_{RB}$)
    Set the number of RB spaces $L = \lceil \log(\varepsilon_r)/\log(\delta_{RB})\rceil$
    Compute an (approximated) affine decomposition of $\mathbf{A}(\boldsymbol{\mu})$
    Compute $\{\mathbf{u}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ and set $\mathbf{S} = [\mathbf{u}(\boldsymbol{\mu}_1), \ldots, \mathbf{u}(\boldsymbol{\mu}_{n_s})]$
    Build the basis $\mathbf{V}_0 = \mathrm{POD}(\mathbf{S}, \delta_{RB})$
    for $k = 1, \ldots, L-1$ do
        Compute new snapshots $\mathbf{y}_k(\boldsymbol{\mu}_i)$, $i = 1, \ldots, n_s$, with (2.42)
        Set $\mathbf{S} = [\mathbf{y}_k(\boldsymbol{\mu}_1), \ldots, \mathbf{y}_k(\boldsymbol{\mu}_{n_s})]$
        $\mathbf{V}_k = \mathrm{POD}(\mathbf{S}, \mathbf{X}_h, \delta_{RB})$
        Build RB affine matrices $\{\mathbf{A}_{N_k}^q\}_{q=1}^{Q_a}$
    end for
end procedure
```

2.3.2 Online phase

In the online phase, we aim at computing the solutions of (2.1) for new instances of the parameter $\boldsymbol{\mu}$, which have not been considered during the offline phase. We thus

Algorithm 5 MSRB - Fixed Dimension

```
procedure MSRB-fixedDimension($\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}, \varepsilon_r, N$)
    Compute an (approximated) affine decomposition of $\mathbf{A}(\boldsymbol{\mu})$
    Compute $\{\mathbf{u}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ and set $\mathbf{S} = [\mathbf{u}(\boldsymbol{\mu}_1), \dots, \mathbf{u}(\boldsymbol{\mu}_{n_s})]$, $k = 0$
    while $\prod_k \delta_{RB,k} > \varepsilon_r$ do
        $\mathbf{V}_k = \mathrm{POD}(\mathbf{S}, \mathbf{X}_h, N)$ and $k = k + 1$
        Build RB affine matrices $\{\mathbf{A}_{N_k}^q\}_{q=1}^{Q_a}$
        Compute new snapshots $\mathbf{y}_k(\boldsymbol{\mu}_i)$, $i = 1, \dots, n_s$, with (2.42)
        Set $\mathbf{S} = [\mathbf{y}_k(\boldsymbol{\mu}_1), \dots, \mathbf{y}_k(\boldsymbol{\mu}_{n_s})]$
    end while
end procedure
```

need to compute the coarse operators $\{\mathbf{Q}_{N_k}^{\mu}\}_k$ through (2.27) and apply the FGMRES (Richardson) algorithm using $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ in the preconditioning step. The operations required by the matrix-vector multiplication $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{v}_k(\boldsymbol{\mu})$ are detailed in algorithm 6; step 3 corresponds to the solution of the RB problem

$$\mathbf{A}_{N_k}(\boldsymbol{\mu})\mathbf{y}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k^T \mathbf{v}_{k+\frac{1}{2}}(\boldsymbol{\mu}), \tag{2.45}$$

determined in our implementation by using the LU factorization of the matrix $\mathbf{A}_{N_k}(\boldsymbol{\mu})$. If the number of iterations required to reach a certain tolerance in the FGMRES method exceeds the number of RB coarse operators constructed, one can either continue to use the last coarse operator in the remaining operations or drop steps 2-3 of Algorithm 6.

Algorithm 6 Computation of $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{v}_k(\boldsymbol{\mu})$

```
1: apply the inverse of the fine component $\mathbf{P}(\boldsymbol{\mu})$: $\mathbf{w}^{(k)} = \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{v}_k(\boldsymbol{\mu})$;
2: build the residual $\mathbf{v}_{k+\frac{1}{2}}(\boldsymbol{\mu}) = \mathbf{v}_k(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu})\mathbf{w}^{(k)}$;
3: apply the RB coarse component $\mathbf{w}^{(k+\frac{1}{2})} = \mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{v}_{k+\frac{1}{2}}(\boldsymbol{\mu})$;
4: build the preconditioned residual $\mathbf{z}_k(\boldsymbol{\mu}) = \mathbf{w}^{(k)} + \mathbf{w}^{(k+\frac{1}{2})}$.
```

2.4 Numerical experiments for elliptic problems

Several numerical experiments involving the numerical solution of linear systems arising from the FE discretization of advection-diffusion PDEs are presented to illustrate the capability of the proposed MSRB-preconditioner. We first focus on heat diffusion in a domain showing a piecewise constant (affinely parametrized) thermal conductivity, to simulate the effect of different material properties in the domain (test case I). Then, we turn our attention to a thermal beam with a nonaffine thermal conductivity which is localized in space (test case II).

As fine preconditioner component, we employ $\mathbf{P}(\boldsymbol{\mu}) = \mathbf{P}_{\mathrm{BJ}}(\boldsymbol{\mu})$, a Block Jacobi preconditioner, where each block represents the restriction of the computational domain Ω to the degrees of freedom of a subdomain selected by `Parmetis`[1] at the mesh level. If the number of iterations required by the iterative solver to reach the prescribed tolerance ϵ_r exceeds the number of spaces (which is fixed once the offline phase has been completed) the final iterations just employ the fine preconditioner, i.e. we set $\mathbf{P}_{\mathrm{MSRB},k}(\boldsymbol{\mu}) = \mathbf{P}(\boldsymbol{\mu})\ \forall k \geq L$. Concerning the solutions of the RB systems, i.e. the computation of $\mathbf{A}_{N_k}^{-1}(\boldsymbol{\mu})$ in (2.39), the very small number of RB functions yields RB problems of small size; consequently a sequential LU factorization is employed to solve either (2.11) or (2.38). For all simulations we report the number of spaces L and RB functions N_k, $k = 0, 1, \dots$ produced by Algorithm 4 or 5, the results obtained online with the MSRB preconditioner averaging on $N_{\mathrm{onl}} = 250$ parameters and the computational time t_{off} required by the offline phase. We compare the results with those obtained using an algebraic multigrid (AMG) preconditioner $\mathbf{P}_{\mathrm{ML}}(\boldsymbol{\mu})$, that exploits an exact coarse component and 2-sweeps Gauss-Seidel smoother obtained with default settings from the `Trilinos` package `ML` [Gee et al., 2006], which is used as preconditioner in the GMRES method (noted as *GML*).

2.4.1 Test case I: diffusion with nonaffine right hand side

We consider a parametrized diffusion problem in a blockwise cubic domain, including anisotropy effects in the diffusion tensor and a nonaffine right hand side to model a source localized in space.

Problem setting

Consider $\Omega = (0,1)^3 \subset \mathbb{R}^3$, such that $\partial\Omega = \Gamma_D \cup \Gamma_N$ with $\overset{\circ}{\Gamma}_D \cap \overset{\circ}{\Gamma}_N = \emptyset$, we subdivide it into $\mathcal{J}$ subregions Ω_j, $j = 1, \dots \mathcal{J}$ s.t. $\bar{\Omega} = \cup_{j=1}^{\mathcal{J}} \bar{\Omega}_j$ and $\overset{\circ}{\Omega}_i \cap \overset{\circ}{\Omega}_j$, $i \neq j$. More precisely, we set $\mathcal{J} = 4$ and subdivide Ω such that $\overset{\circ}{\Omega}_1 = (0,1) \times (0,0.5) \times (0,0.5)$, $\overset{\circ}{\Omega}_2 = (0,1) \times (0,0.5) \times (0.5,1)$, $\overset{\circ}{\Omega}_3 = (0,1) \times (0.5,1) \times (0,0.5)$, $\overset{\circ}{\Omega}_4 = (0,1) \times (0.5,1) \times (0.5,1)$. We consider problem

$$\begin{cases} -\nabla \cdot (\mathcal{K}(\boldsymbol{\mu}) \nabla u(\boldsymbol{\mu})) = f(\boldsymbol{\mu}) & \text{in } \Omega \\ u(\boldsymbol{\mu}) = 0 & \text{on } \Gamma_D \\ \mathcal{K}(\boldsymbol{\mu}) \nabla u(\boldsymbol{\mu}) \cdot \vec{n} = 0 & \text{on } \Gamma_N, \end{cases} \tag{2.46}$$

---

[1] http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview

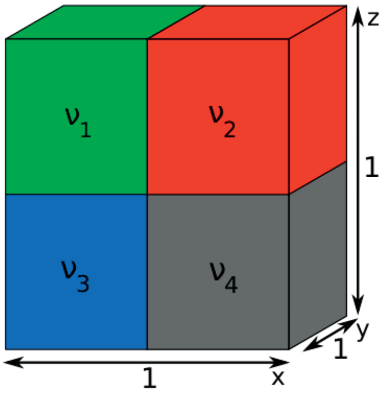


Figure 2.1 – Test case I: the computational domain Ω is divided into four regions, each featuring a constant diffusivity value ν_j, $j = 1, \dots, 4$. The coefficients $\nu_j, , j-1,2,3$ are taken as problem parameters, cf. (2.48), whereas we fix the value $\nu_4 = 1$.

which is a special case of (1.10) where $\vec{b}(\boldsymbol{\mu}) = \vec{0}$ and

$$\mathcal{K}(\boldsymbol{\mu}) = \mathcal{K}(\mathbf{x}; \boldsymbol{\mu}) = \nu(\mathbf{x}) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10^{-2} \end{bmatrix}.$$

Here $\nu(\mathbf{x}) > 0$ is the piecewise constant material property on each Ω_j, such that $\nu(\mathbf{x}) = \sum_{j=1}^{4} \nu_j \mathcal{X}_{\Omega_j}$, where $\mathcal{X}_{\Omega_j}$ is the indicator function that is equal to 1 on Ω_j and 0 otherwise, and

$$\Gamma_N = \Big\{ \mathbf{x} = (x_1, x_2, x_3) \in \bar{\Omega} \, : \, x_1 = 1 \Big\}, \qquad \Gamma_D = \partial\Omega \backslash \Gamma_N.$$

We provide in Figure 2.1 a sketch of the subdivision of the domain. We consider as source term the following parameter dependent function

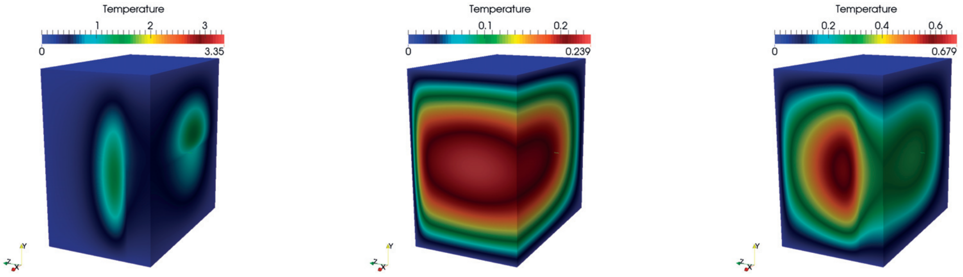
$$f(\boldsymbol{\mu}) = f(\mathbf{x}; \boldsymbol{\mu}) = \sigma + \frac{1}{\sigma} \exp\Big(\frac{\|\mathbf{x} - \mathbf{x}_0\|^2}{\sigma} \Big), \tag{2.47}$$

that is, a Gaussian function centered at $\mathbf{x}_0 \in \Omega$ and rescaled by a factor $\sigma > 0$.

We parametrize the problem with respect to the diffusion coefficients ν_j, $j = 1, \dots, \mathcal{J} - 1$, the coordinates $\mathbf{x}_0$ and the scaling factor σ appearing in the definition of (2.47), leading to the 7-dimensional parameter vector:

$$\boldsymbol{\mu} = (\nu_1, \dots, \nu_3, \mathbf{x}_0, \sigma) \in \mathcal{D} = [0.1, 1]^{\mathcal{J}-1} \times [0.4, 0.6]^3 \times [\sigma_{\min}, 0.5] \subset \mathbb{R}^7, \tag{2.48}$$

where $\sigma_{\min} > 0$; on the other hand, we fix the value of the coefficient ν_4 to 1. The localized (in space) parametrized nature of $f(\boldsymbol{\mu})$, together with the varying diffusion coefficients yield a problem which is challenging from the parameter viewpoint, as it is hardly solvable



(a) $(0.1, 0.1, 0.1, 0.4, 0.4, 0.4, 0.05)$ (b) $(1, 1, 1, 0.6, 0.6, 0.6, 0.5)$ (c) $(1, 0.5, 1, 0.5, 0.5, 0.5, 0.2)$

Figure 2.2 – Test case I: example of solutions for different values of $\boldsymbol{\mu}$ with a null Neumann condition on $x = 1$.

accurately by the standard RB method due to the nonaffinely parametrized source term $f(\boldsymbol{\mu})$. As we will see in the next sections, its localized nature requires a large amount of DEIM basis functions to accurately reconstruct it, significantly hampering the efficiency of the RB approximation. Moreover, multilevel preconditioners are known to be the state-of-the-art technique for second-order symmetric elliptic problems, however the anisotropy appearing in $\mathcal{K}(\boldsymbol{\mu})$ significantly lowers its capabilities with respect to the isotropy case.

For the sake of simplicity, we consider homogeneous Dirichlet and Neumann boundary conditions, although the whole framework can be easily adapted to the case of nonhomogeneous (parametrized) boundary conditions in a straightforward way. Moreover, in all simulations, we employ linear piecewise continuous FE tetrahedra on structured meshes as high-fidelity discretization. Examples of solutions obtained for different values of parameters, are reported in Figure 2.2. POD is always run with $n_s = 750$ snapshots with respect to the scalar product induced by the symmetric positive definite matrix $\mathbf{X}_h$, which represents the $H_0^1(\Omega)$ scalar product on V_h. A stopping criterion based on the Euclidean norm of the (finite element vector of the) residual, rescaled with respect to the Euclidean norm of the right hand side of the system has been used for all the tests, with a tolerance set to 10^{-7}. Furthermore, we build the RB spaces such that inequality (2.44) is satisfied with $\delta = 10^{-9}$, since POD is optimal in the sense of minimizing the sum of the squared projection errors onto the reduced space evaluated on the selected snapshots, when the reduced solution for a different parameter is computed, the corresponding error can be slightly larger. Results for both the Richardson and the FGMRES method are presented.

Numerical results for the Richardson method

We start by considering the Richardson method with a computational mesh leading to $N_h = 365'254$ (run on $N_{\text{core}} = 96$ cores and with $\sigma_{\min} = 0.25$) and assess the accuracy of

Table 2.1 – Test case I: Richardson method results with $\mathcal{P}_1$ FE: $\delta_{RB,k} = 0.001$, $k = 0, \dots 2$. We report the dimension of the RB spaces N_k and the average accuracy $\xi_{RB}^{(k)}$ obtained online for $k = 0, \dots 2$.

k	1	2	3
N_k	45	187	702
$\xi_{RB}^{(k)}$	2.5e-02	1.1e-02	3.9e-02

Table 2.2 – Test case I: Richardson method results with $\mathcal{P}_1$ FE: $\delta_{RB,k} = 0.1$, $k = 0, \dots 8$. We report the dimension of the RB spaces N_k and the average accuracy $\xi_{RB}^{(k)}$ obtained online for $k = 0, \dots 8$.

k	1	2	3	4	5	6	7	8	9
N_k	5	19	40	81	148	228	319	387	386
$\xi_{RB}^{(k)}$	4.3e-01	2.3e-01	3.3e-01	2.7e-01	2.1e-01	2.2e-01	2.5e-01	3.3e-01	4.0e-01

the RB solvers employed within the MSRB preconditioning strategy, to verify that the residual decays with a rate proportional to the tolerances employed to construct the RB spaces. Consequently, we compute the coefficients $\xi_{RB}^{(k)}(\boldsymbol{\mu}), k = 0, 1, \dots, L-1$ as

$$\xi_{RB}^{(k)}(\boldsymbol{\mu}) = \frac{\|\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu}) - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu})\|_{\mathbf{X}_h^{-1}}}{\|\mathbf{r}^{(k-1/2)}(\boldsymbol{\mu})\|_{\mathbf{X}_h^{-1}}} \qquad k = 0, \dots, L-1,$$

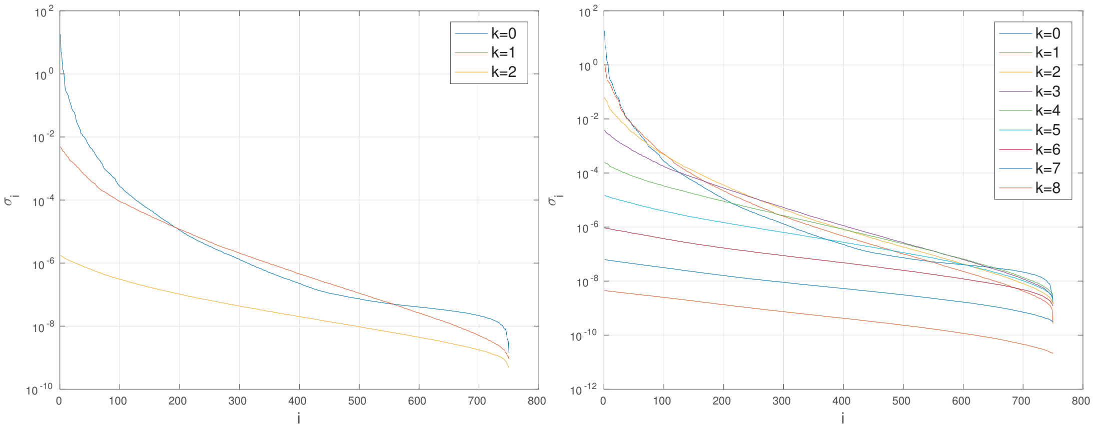
to measure the accuracy of the RB coarse component employed at step k. In fact, $\xi_{RB}^{(k)}(\boldsymbol{\mu})$ represents the RB residual associated to error equation (2.5) at iteration k; we denote by $\xi_{RB}^{(k)}$ the corresponding quantity obtained by average over the parameters considered online. We build the MSRB preconditioner with the fixed accuracy approach (Algorithm 4) and:

i) $\delta_{RB,k} = 0.001$, yielding the construction of $L = 3$ RB spaces,

ii) $\delta_{RB,k} = 0.1$, yielding $L = 9$ RB spaces.

In Table 2.1 and 2.2 the results corresponding to cases *i*) and *ii*) are reported: the dimension N_k of the RB spaces and the obtained average coefficients $\xi_{RB}^{(k)}$. All the RB spaces provide a fixed online accuracy $\xi_{RB}^{(k)}$, equal to about 10^{-2} in the first case, and 10^{-1} in the second case, thus yielding a constant decay of the residual.

The results show that generating different levels with the same tolerance yields RB spaces whose dimensions grow with the iteration count k, see Table 2.1-2.2. This fact is also confirmed by the decay of the eigenvalues of the correlation matrices $\mathbf{C}_k = \mathbf{S}_k^T \mathbf{X}_h \mathbf{S}_k$, $k =$



(a) $\delta_{RB,k} = 0.001, \, , k = 0, 1, 2$ (b) $\delta_{RB,k} = 0.1, \, , k = 0, \ldots, 8$

Figure 2.3 – Test case I: eigenvalues of the correlation matrix $\mathbf{C}_k$, $k = 0, \ldots, L-1$, $N_h = 365'254$. As the iteration k increases, the decay of the eigenvalues is less steep, leading to RB spaces with more basis functions to reach the same tolerance.

$0, L-1$, reported in Figure 2.3 for $L = 3, 9$. Here

$$\mathbf{S}_k = [\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu}_1)| \ldots |\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu}_{n_s})]$$

denotes the snapshots matrix employed for the construction of level k. As k grows, the decay of the eigenvalues is slower, so that larger RB spaces are needed to reach the same tolerance. This behavior can be ascribed to the fact that at step k the manifold $\mathcal{M}_k = \{\mathbf{e}^{(k-1/2)}(\boldsymbol{\mu}), \ \boldsymbol{\mu} \in \mathcal{D}\}$ is less regular compared to $\mathcal{M}_0, \ldots, \mathcal{M}_{k-1}$: the higher k, then the more noisy the pattern of the error, the smaller its magnitude and the more difficult its approximation.

Once a new instance of the parameter $\boldsymbol{\mu}$ is considered online, the linear system with the MSRB-preconditioned Richardson method employs on average 5 iterations to reach convergence in case *i)* and 13 in case *ii)*, thanks to the fast convergence obtained by using accurate RB coarse corrections.

Analysis with respect to the mesh size

From now on, we employ the FGMRES for all next simulations. We start by carrying out an analysis with respect to three different grids whose characteristic dimensions are $h = 0.05$, 0.025, 0.0125, leading to dimensions $N_h = 365'254$, $2'887'193$ and $22'767'295$, respectively, for the high-fidelity FE approximation. Similarly to what previously done, we choose $\sigma_{\min} = 0.25$ and construct the RB spaces by POD with $n_s = 750$ snapshots. These simulations have been carried out with 96, 768, 6144 cores, respectively, in order to maintain the same number of degrees of freedom (about 3800) per core. We compare the results with those obtained using an algebraic multigrid preconditioner $\mathbf{P}_{\mathrm{ML}}(\boldsymbol{\mu})$ from `ML` package [Gee et al., 2006] of `Trilinos`, which exploits an exact coarse component and 2-sweeps Gauss-Seidel smoother and with the GCRO-DR Krylov subspace recycling method proposed in [Parks et al., 2006], where $\mathbf{P}_{\mathrm{ML}}(\boldsymbol{\mu})$ is again used as preconditioner, for sequences of linear systems with varying matrices and right hand sides. The latter method combines the optimal truncation strategy of GCRO [De Sturler, 1999] with deflation employed in GMRES with deflated restarting, GMRES-DR [Morgan, 2002]. Both techniques are used with default settings stated by the Trilinos library.

The results are reported in Table 2.3, 2.4. The computational time employed online to solve the linear system (2.1) using $\mathbf{P}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ as preconditioner for the new instances of the parameter is not highly impacted by the FE dimension, since the number of RB coarse components and their dimensions are not significantly affected by the FE dimension. Indeed, the online computational time $t^{\mathrm{onl}}_{\mathrm{MSRB}}$ and the number of iterations are always lower than the ones obtained either by $\mathbf{P}_{\mathrm{ML}}(\boldsymbol{\mu})$ (t_{GML}) or GCRO-DR ($t_{\text{G-DR}}$), for both the fixed dimension and the fixed accuracy approaches. Moreover, we notice that the MSRB preconditioner built with the fixed accuracy approach features a faster online solution and a less expensive offline phase than the one built with the fixed dimension approach. In Table 2.3 and 2.4 the break-even point (BEP), quantifying the number of online evaluation needed to repay the offline phase, is also reported. This criterion is based on the wall time comparison, where

$$\mathrm{BEP} = \frac{t_{\mathrm{off}}}{\min\{t_{\mathrm{GML}}, t_{\text{G-DR}}\} - t^{\mathrm{onl}}_{\mathrm{MSRB}}}.$$

Here t_{off} denotes the wall time required by the offline computation, i.e. the construction of the RB coarse components, and is given by the sum of t_{n_s}, the time for the snapshots computation, and the time t_{POD} for building the basis with POD; as a matter of fact, $t_{\mathrm{off}} = t_{n_s} + t_{\mathrm{POD}}$. On the other hand, t_{GML}, $t_{\text{G-DR}}$ and $t^{\mathrm{onl}}_{\mathrm{MSRB}}$ denote the wall times needed to solve the FE linear system (2.1) for any new instance of $\boldsymbol{\mu}$ during the online phase using the preconditioner $\mathbf{P}_{\mathrm{ML}}(\boldsymbol{\mu})$, the GCRO-DR method and the preconditioner $\mathbf{Q}_{\mathrm{MSRB},k}$, respectively.

The larger the FE dimension, the lower the BEP (up to 1067 and 1240 parameters in the case of the finest grid, depending on the construction approach); indeed, by increasing N_h, the use of the MSRB preconditioner is more convenient compared to the use of $\mathbf{P}_{\mathrm{ML}}(\boldsymbol{\mu})$ or the GCRO-DR method, even though a more demanding offline phase must be performed. In Figure 2.4a and 2.4b the speedup obtained with respect to the most convenient choice between $\mathbf{P}_{\mathrm{ML}}(\boldsymbol{\mu})$ and GCRO-DR technique and the BEP are reported as function of the FE dimension. By comparing these quantities for both the fixed dimension and the fixed accuracy approaches, we conclude that the larger the FE dimension, the higher the speedup and the lower the break-even point for both approaches. In the case with $N_h = 22'767'295$, both $\mathbf{P}_{\mathrm{ML}}(\boldsymbol{\mu})$ and the GCRO-DR perform very poorly due to the very large FE dimension and the corresponding huge communication costs; in particular the latter succeeds in reducing the time of about 10% by recycling the Krylov subspace. On the other hand, the MSRB preconditioner employs embarrassingly parallel fine and coarse components, and the linear system (2.1) is solved by the MSRB preconditioned FGMRES up to 70 (resp. 50) faster than either $\mathbf{P}_{\mathrm{ML}}(\boldsymbol{\mu})$ or GCRO-DR for the fixed accuracy (resp. fixed dimension) approach. In terms of memory requirements, the fixed accuracy approach (entailing the storage of about 1050 FE vectors to build the RB spaces for the problem at hand) is less demanding than the fixed dimension approach (about 1400 FE vectors). These requirements make data storage related to our preconditioners heavier than the one required by the ML preconditioner, although this latter is used only for a single-instance of the parameter, if no updating or recycling techniques are employed. Nevertheless, compared to the standard RB method, the number of FE vectors stored by our preconditioners is comparable since it is an intrinsic fact caused by the RB method employed by the coarse operators.

Table 2.3 – Test case I: grid analysis results for FGMRES method with fixed accuracy approach, $L = 3$, $\delta_{RB,k} = 0.001$, $\forall k$, ~ 3800 dofs per CPU. Times are in seconds.

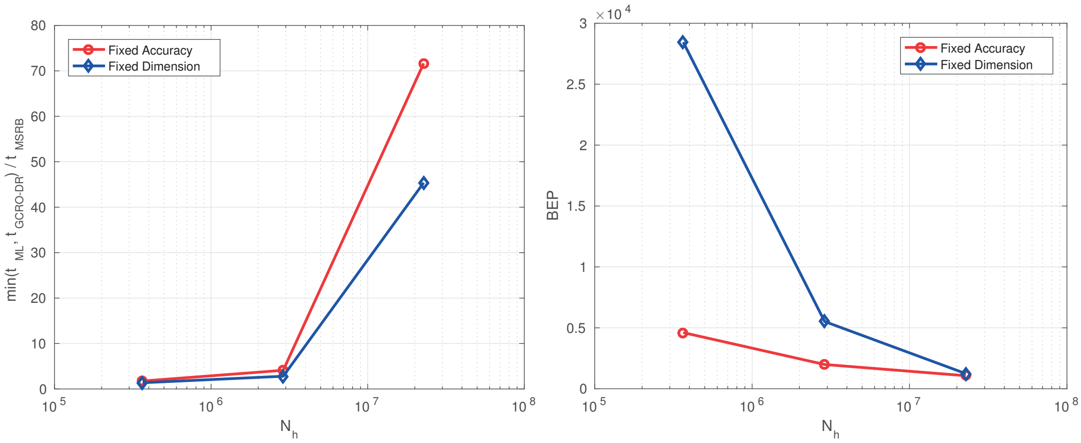
N_{core}	N_k	$t^{\text{onl}}_{\text{MSRB}}(It^{\text{onl}})$	$t_{\text{GML}}(It)$	$t_{\text{G-DR}}(It)$	t_{off}	t_{n_s}	t_{POD}	BEP
96	49 296 725	0.34 (5)	0.59 (28)	0.48 (28)	1161.21	1071.66	89.55	4606
768	48 279 721	0.46 (9)	1.91 (41)	2.29 (38)	2872.27	2746.07	126.20	1989
6144	49 269 713	0.75 (12)	55.73 (54)	49.98 (53)	56768.20	56486.86	281.34	1067

Table 2.4 – Test case I: grid analysis results for FGMRES method with fixed dimension approach, $N_k = 100\ \forall k$, ~ 3800 dofs per CPU. Time are expressed in seconds.

N_{core}	L	$t^{\text{onl}}_{\text{MSRB}}(It^{\text{onl}})$	$t_{\text{GML}}(It)$	$t_{\text{G-DR}}(It)$	t_{off}	t_{n_s}	t_{POD}	BEP
96	15	0.43 (13)	0.59 (28)	0.48 (28)	4546.21	4390.47	155.74	28448
768	14	0.68 (25)	1.91 (41)	2.29 (38)	6775.94	6597.13	178.81	5517
6144	13	1.19 (40)	55.73 (54)	49.98 (53)	65437.90	64951.30	486.60	1240

Scalability test

In this section we test the scalability capabilities of the MSRB preconditioner and compare them with the ones obtained with the AMG preconditioner when $\mathcal{P}_2$ FE basis functions are employed; the dimension of the FE linear system is $N_h = 2'848'000$. We keep the same setting of the previous tests and use both a fixed accuracy (Table 2.5) and fixed dimension (Table 2.6) approach. We build three RB coarse operators with $\delta_{RB,k} = 10^{-3}$ with the former approach, whereas we set $N_k = 150$ for any k in latter, which leads to the construction of eight RB coarse corrections in all the tests where it is employed. As a matter of fact, the computational times during the online phase dramatically reduces when using the MSRB preconditioner. This is due to the fact the proposed preconditioner is composed of two embarrassingly parallel components: the RB coarse operator entails a little cost since its dimension is very limited (its size is at most 700 in the last RB operator for the fixed accuracy approach), whereas the fine



(a) Speedup.

(b) Break-even point.

Figure 2.4 – Test case I: speed up and break-even point (BEP) as function of N_h for both fixed accuracy and fixed dimension approaches. Time are expressed in seconds.

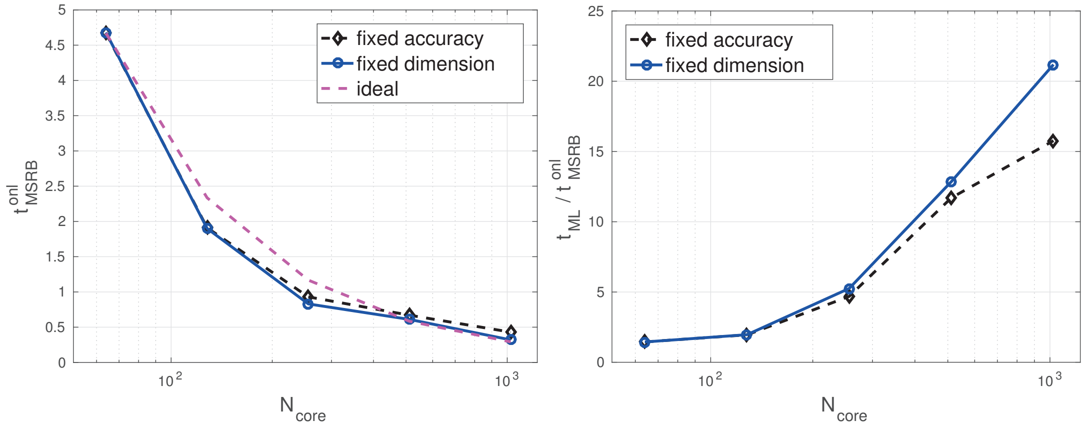
Table 2.5 – Test case I: scalability analysis results for FGMRES method with fixed accuracy approach, $\delta_{RB,k} = 10^{-3}\ \forall k$. Time are expressed in seconds.

N_{core}	N_k	$t^{\text{onl}}_{\text{MSRB}}(It^{\text{onl}})$	$t_{\text{GML}}(It)$	t_{off}
64	46 265 699	4.67(2)	6.77(59)	26626.7
128	46 260 694	1.91(2)	3.73(59)	12167.1
256	46 262 700	0.93(3)	4.36(60)	8887.1
512	46 259 701	0.67(3)	7.84(92)	11360.9
1024	46 255 698	0.43(3)	6.78(70)	10109.6

grid preconditioner is a Block Jacobi operator which does not require communication. This provides a good overall scalability, as shown in Figure 2.5a, where we report the time entailed by the MSRB preconditioner as function of the number of cores N_{core} to assemble the preconditioner and solve the FE linear system. On the other hand, when the number of cores increases, the computational time provided by the ML operator is stuck due to the communication required by the prolongation and restriction operators: the larger the number of cores, the more convenient the use of a MSRB preconditioner; in fact its use is up to 20 times faster than employing $\mathbf{P}_{\text{ML}}(\boldsymbol{\mu})$, as reported in Figure 2.5b. The computational time entailed by the fixed dimension approach is lower since the size of the RB matrices is smaller, thus yielding to a faster construction and LU factorization of $\mathbf{A}_{N_k}(\boldsymbol{\mu})$, $k = 0, \dots, L-1$. This is particularly evident when the number of cores grows, since the construction and application time of $\mathbf{P}_{\text{BJ}}(\boldsymbol{\mu})$ gets milder, hence the one of the RB coarse operators become relatively more important. Furthermore, the number of iteration It^{onl} is about the same and ranges from 2 to 4 in all the simulations when using the MSRB preconditioner; instead about 60 or more iterations are needed when the ML preconditioner is used.

Table 2.6 – Test case I: scalability analysis results for FGMRES method with fixed dimension approach, $N_k = 150\ \forall k$. $L = 8$ RB coarse operators are produced by Algorithm 5 in all the cases. Time are expressed in seconds.

N_{core}	$t^{\text{onl}}_{\text{MSRB}}(It^{\text{onl}})$	$t_{\text{GML}}(It)$	t_{off}
64	4.68(3)	6.77(59)	60821.5
128	1.9(3)	3.73(59)	25162.1
256	0.83(3)	4.36(60)	13457.3
512	0.61(4)	7.84(92)	15067
1024	0.32(3)	6.78(70)	11298.7



(a) Computational time as function of the number cores N_{core}.

(b) Speedup with respect the use of $\mathbf{P}_{\text{ML}}(\boldsymbol{\mu})$.

Figure 2.5 – Test case I: scalability and speed up as function of N_{core} for both fixed accuracy and fixed dimension approaches. Time are expressed in seconds.

Comparison with RB-DEIM

A natural question arising in this context is about the comparison, in terms of both accuracy and efficiency, between the proposed approach (MSRB preconditioning) and the classical RB method. In this latter case, the solution of system (2.1) is approximated by the one of the RB system (1.39). In this section we compare the results obtained with the standard RB method with the ones computed with the FGMRES method preconditioned with the proposed MSRB preconditioner, showing results for the FE grid with $N_h = 2'887'193$.

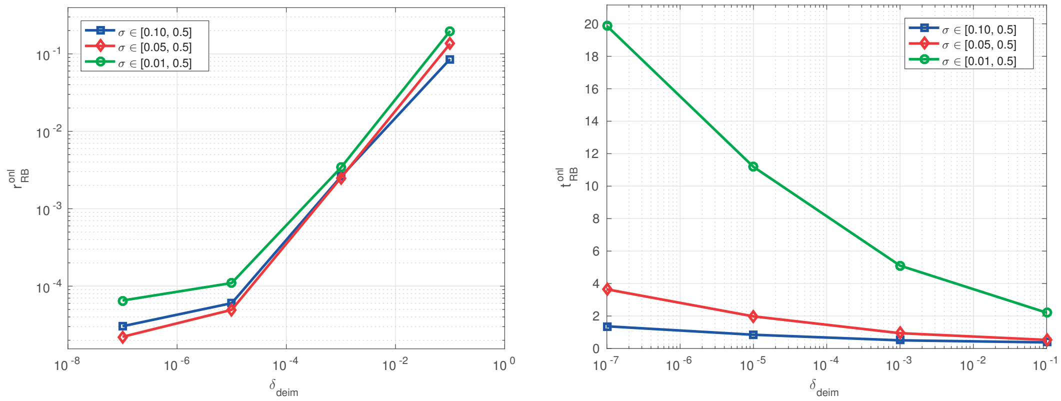
At first, we notice that the function (2.47) nonaffinely depends on the parameter $\boldsymbol{\mu}$, leading to a nonaffine right hand side $\mathbf{f}(\boldsymbol{\mu})$ in (2.1). The nonaffine dependence of the operators is one of the most limiting bottlenecks of the standard RB method, as it does not allow to assemble the RB arrays independently from the FE dimension and gain the maximum speed up with respect to the high-fidelity simulation. In our case, we employ the DEIM algorithm [Chaturantabut and Sorensen, 2010], see Section 1.3.2, to deal with the nonaffine right hand side. This latter is approximated as a linear combination of properly chosen DEIM basis functions up to a certain tolerance δ_{DEIM}, which is plugged in the DEIM algorithm. We use the shorthand notation RB-DEIM to indicate the RB method which exploits the DEIM algorithm to compute an affine approximation of the right hand side. It is well known that, on one hand, the tolerance δ_{DEIM} limits the accuracy of the RB-DEIM approximation and, on the other hand, it may yield a huge overhead in the online phase due to a (possibly) large number of DEIM basis functions. This is indeed the case of the data in (2.47) due to the localized (in space) nature of the source term, which depends on the value of σ (the smaller σ, the more localized the source term).

We employ RB-DEIM with different DEIM tolerances $\delta_{\text{DEIM}} = 10^{-1}$, 10^{-3}, 10^{-5}, 10^{-7} and values of $\sigma_{\text{min}} = 0.1, 0.05, 0.01$, such that the parameter $\sigma \in [\sigma_{min}, 0.5]$. The RB spaces are built through POD algorithm by setting a tolerance of $\varepsilon_{\text{POD}} = 10^{-9}$ for all the tests; we choose a number of snapshots equal to $n_s = 1000$ for $\sigma_{\text{min}} = 0.1$, $n_s = 2000$ for $\sigma_{\text{min}} = 0.05$ and $n_s = 3500$ for $\sigma_{\text{min}} = 0.01$, respectively. An increasing number of snapshots is chosen in order to properly sample the parameter, which requires a more accurate sampling as σ_{min} decreases. To assess the accuracy of the RB solution, the average FE relative residual computed in the RB solution, which is defined as

$$r_{\text{RB}}(\boldsymbol{\mu}) = \frac{\|\mathbf{f}(\boldsymbol{\mu}) - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{V}\mathbf{u}_N(\boldsymbol{\mu})\|_2}{\|\mathbf{f}(\boldsymbol{\mu})\|_2}, \tag{2.49}$$

is evaluated and averaged over $N_{\text{onl}} = 250$ online parameters. We denote by r_{RB} the corresponding averaged quantity.

In Figure 2.6a r_{RB} is reported for different DEIM tolerances δ_{DEIM}. The results show that the accuracy of RB-DEIM is strongly hampered by the tolerance δ_{DEIM}; not only, it is compulsory to use a small value of δ_{DEIM} to obtain a very accurate solution. Moreover, we observe that from a certain point on the residual stagnates to the value 10^{-5} even if a smaller δ_{DEIM} has been provided. In Figure 2.6b the wall time $t_{\text{RB}}^{\text{onl}}$ employed to assemble and solve the RB problem for a new instance of $\boldsymbol{\mu}$ is reported for different values of σ_{min} as function of δ_{DEIM}. The total time to solve the RB problem for a new instance of the $\boldsymbol{\mu}$ is largely affected by the value of σ_{min} and by δ_{DEIM}: the smaller the tolerance of the DEIM algorithm, the bigger the wall time required to compute the RB solution (even up to 19.87 seconds for $\sigma \in [0.01, 0.5]$ and $\delta_{\text{DEIM}} = 10^{-7}$). In particular, the large wall time required when using a small DEIM tolerance is caused by the assembling the RB right hand side, which depends on the huge number of DEIM basis functions and the communication needed to compute the coefficients $\tilde{\Theta}_f^q$, $q = 1, \dots, Q_f$ in (1.59), see Figures 2.7a-2.7b-2.7c, where the time required online for assembling the RB matrix and right hand side and solve the RB system are reported for different values of σ_{min} and δ_{DEIM}. In Table 2.7 we report the results for the different ranges of σ: the average relative residual r_{RB} and the wall time $t_{\text{RB}}^{\text{onl}}$ for the online computation of the RB solution, the number Q_f of DEIM terms, the offline phase time t_{off} and the snapshots number n_s.



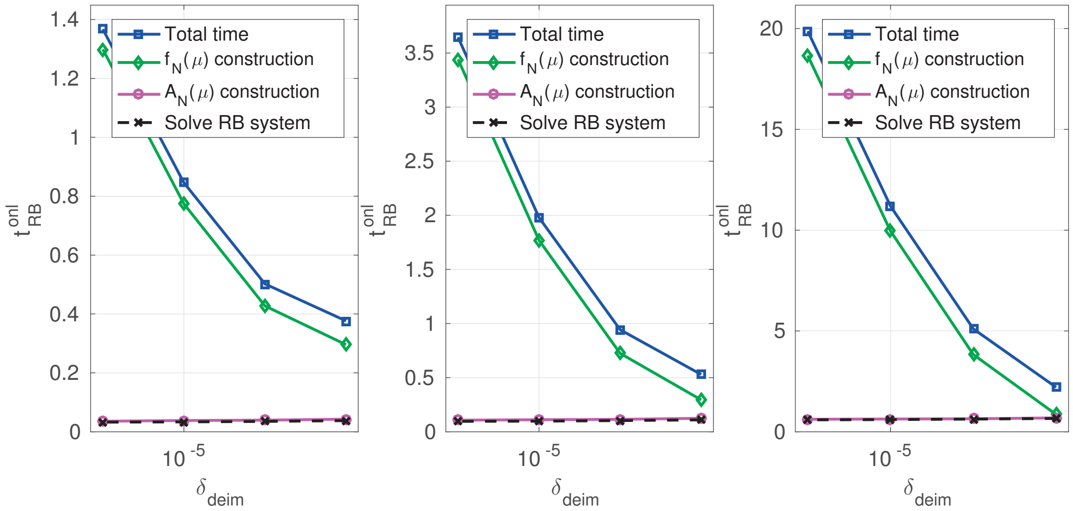
(a) Relative residual.

(b) Wall times for solving and assembling the RB problem.

Figure 2.6 – Test case I: average relative RB residual r_{RB} and wall time to solve assemble and solve the RB-DEIM problem as function of δ_{DEIM} for different ranges of σ: the larger the range, the less efficient the RB-DEIM approximation. Time are expressed in seconds.

By contrast, in the FGMRES method preconditioned with the MSRB preconditioner, an approximated affine decomposition of the right hand side is not needed, since we solve the full FE problem, therefore the use of DEIM is not required; as a matter of fact both the offline time to build a DEIM basis and the online time to assemble the RB right hand side with (1.62) are saved. In Table 2.8 the results obtained by setting a final relative tolerance for the FGMRES equal to $\varepsilon_r = 10^{-7}$ are shown for the fixed dimension approach. In particular, we report, together with the number L of coarse operators, the number of RB functions N defining each coarse operator, the wall time $t_{\mathrm{MSRB}}^{\mathrm{onl}}$ and the iterations It required to compute the solution with the MSRB-preconditioned FGMRES method, the computational time t_{off} of the offline phase and the number of snapshots n_s. For $\sigma_{\min} = 0.1, 0.05, 0.01$, we have set $N = N_k = 180, 300, 600$ for any k, respectively. In all cases, Algorithm 5 has built $L = 13$ RB spaces, and compared with the results obtained with the RB-DEIM method, we highlight that:

- the MSRB preconditioner is more insensitive to the range of parameters (and particularly to the values of σ), whereas the performance of the RB-DEIM method strongly depends on it;
- as a matter of fact, the wall time for each online solution ranges from 1.37 to 19.87 seconds for RB-DEIM ($t_{\mathrm{RB}}^{\mathrm{onl}}$) and from 1.05 to 1.59 seconds in the MSRB case ($t_{\mathrm{MSRB}}^{\mathrm{onl}}$);
- the relative residual of the RB-DEIM approximation stagnates at the value of 10^{-5}, even though a smaller tolerance δ_{DEIM}, equal to 10^{-7}, is employed; the MSRB preconditioning method allows instead to obtain a relative residual lower than the tolerance $\varepsilon_r = 10^{-7}$, which is fixed as stopping criterion in the FGMRES algorithm.



(a) $\sigma \in [0.1, 0.5]$. (b) $\sigma \in [0.05, 0.5]$. (c) $\sigma \in [0.01, 0.5]$.

Figure 2.7 – Test case I: average wall time (seconds) divided into three phases for computing the RB solution when a new instance of the parameter is considered in the online phase as function of δ_{DEIM} and for different ranges of σ. Time are expressed in seconds.

This remarkable gain obtained with the MSRB-preconditioned FGMRES with respect to the standard RB method in both accuracy and efficiency is achieved at the expense of a significantly higher offline time t_{off}, equal to 50973.70 seconds in the MSRB case and 30804.32 seconds in the RB-DEIM case (for $\sigma \in [0.01, 0.5]$). This overhead is caused by the larger number of PODs to be performed and the necessity to build the snapshots errors with (2.42), however, it is well repaid during the online phase, when the FGMRES method with the MSRB preconditioner reaches a much more accurate (100 times) result than the RB-DEIM approach, showing also a relevant speedup, up to almost 12 times faster of the standard RB method (for $\sigma \in [0.01, 0.5]$).

For problems involving a nonaffine (left and/or) right hand side, the RB method must rely on hyper-reduction like DEIM to compute an approximated affine decomposition of $\mathbf{f}(\boldsymbol{\mu})$. In the considered case, the use of DEIM strongly limits the accuracy of the RB approximation and entails a huge overhead, see Table 2.7. On the other hand, the MSRB-preconditioned FGMRES method does not require any approximated affine representation of $\mathbf{f}(\boldsymbol{\mu})$, therefore this limitation in accuracy and efficiency does not occur. As a matter of fact, the proposed preconditioning strategy is well-suited when dealing with challenging nonaffine PDEs, since it allows to exploit the parameter dependence overcoming the need to have an accurate affine decomposition of the FE arrays.

Table 2.7 – Test case I: RB-DEIM results with $\delta_{\text{DEIM}} = 10^{-7}$ as function of the range of $\sigma \in [\sigma_{\min}, 0.5]$. The value of $\sigma_{\min}$ significantly impact on the results obtained by the RB-DEIM, both in terms of accuracy and efficiency, due to the number N of RB functions and number Q_f DEIM basis functions.

	r_{RB}	N	Q_f	$t_{\text{RB}}^{\text{onl}}$	t_{off}	n_s
$\sigma \in [0.1, 0.5]$	3.03e-05	670	196	1.37	7913.75	1000
$\sigma \in [0.05, 0.5]$	2.22e-05	1055	341	3.65	17562.31	2000
$\sigma \in [0.01, 0.5]$	6.52e-05	2143	1060	19.87	30804.32	3500

Table 2.8 – MSRB results with fixed dimension approach, with $N_k = N$, $k = 0, 1, \ldots, L-1$ as function of the range of $\sigma \in [\sigma_{\min}, 0.5]$. The efficiency is not significantly impacted by the range of σ, since the MSRB-preconditioning method does not rely on any affine approximation of the right hand side, avoiding the huge overhead caused by DEIM. Time are expressed in seconds.

	ε_r	L	N	$t_{\text{MSRB}}^{\text{onl}}(It)$	t_{off}	n_s
$\sigma \in [0.1, 0.5]$	1.e-7	13	180	1.05 (22)	13820.00	1000
$\sigma \in [0.05, 0.5]$	1.e-7	13	300	1.00 (15)	26406.40	2000
$\sigma \in [0.01, 0.5]$	1.e-7	13	600	1.59 (17)	50973.70	3500

2.4.2 Test case II: Thermal beam

The second test case concerns a parametrized thermal beam. In the considered numerical example, the thermal conductivity $\alpha(\boldsymbol{\mu})$ of the beam features a nonaffine parameter dependence, which enters in the definition of the stiffness matrix $\mathbf{A}(\boldsymbol{\mu})$; as a result, an exact parameter affine dependence of the matrix cannot be computed. This setting represents a notable difference with respect to the problem considered in the previous section, where the FE matrix featured an affine parameter dependence. To overcome this issue an approximated affine decomposition is recovered with MDEIM and used within the MSRB-preconditioning framework, as outlined in Section 2.2.4.

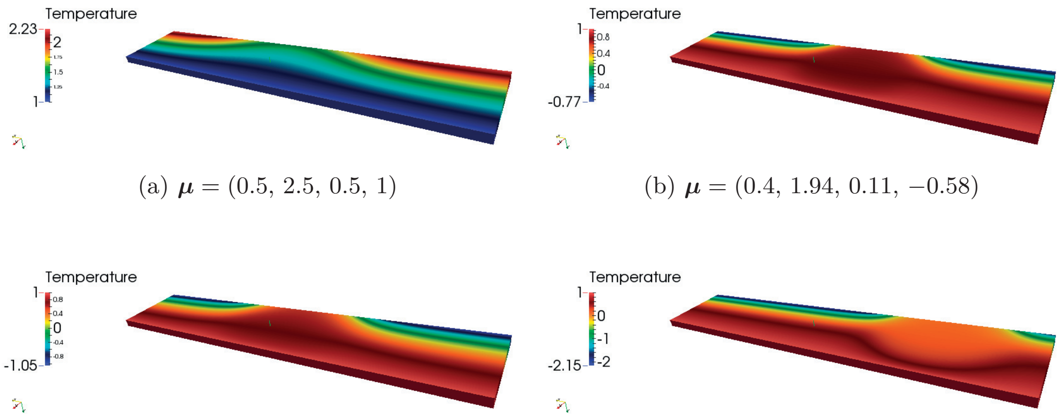
Problem setting

Let us consider the domain $\Omega = (0,1) \times (0,5) \times (0,0.1)$ and a diffusion problem under the form

$$\begin{cases} -\nabla \cdot (\alpha(\boldsymbol{\mu}) \nabla T) = 0 & \text{in } \Omega \\ \alpha(\boldsymbol{\mu}) \nabla T \cdot \vec{n} = h(\boldsymbol{\mu}) & \text{on } \Gamma_N^1 \\ \alpha(\boldsymbol{\mu}) \nabla T \cdot \vec{n} = 0 & \text{on } \Gamma_N^2 \\ T = 1 & \text{on } \Gamma_D, \end{cases} \tag{2.50}$$

where T is the temperature of the beam. In (2.50) we define the boundaries

$$\Gamma_D = \{\mathbf{x} \in \bar{\Omega} : \ x = 1\}, \qquad \Gamma_N^1 = \{\mathbf{x} \in \bar{\Omega} : \ x = 0\}, \qquad \Gamma_N^2 = \partial\Omega \backslash \Gamma_D \backslash \Gamma_N^1,$$



(a) $\boldsymbol{\mu} = (0.5,\ 2.5,\ 0.5,\ 1)$

(b) $\boldsymbol{\mu} = (0.4,\ 1.94,\ 0.11,\ -0.58)$

(c) $\boldsymbol{\mu} = (0.39,\ 2.33,\ 0.19,\ -0.91)$

(d) $\boldsymbol{\mu} = (0.2,\ 0.5,\ 0.1,\ -1)$

Figure 2.8 – Solution $T(\boldsymbol{\mu})$ for different values of $\boldsymbol{\mu}$ computed with $\mathcal{P}_2$ finite elements basis functions, leading to $N_h = 2'080'389$.

such that $\partial\Omega = \bar{\Gamma}_D \cup \bar{\Gamma}_N^1 \cup \bar{\Gamma}_N^2$, the parameter vector

$$\boldsymbol{\mu} = (x_0^1, x_0^2, \sigma, \eta_0) \in [0.2, 0.5] \times [0.5, 2.5] \times [0.1, 0.5] \times [-1, 1] \subset \mathbb{R}^4$$

and the coefficients

$$\alpha(\boldsymbol{\mu}) = \alpha(\mathbf{x}; \boldsymbol{\mu}) = \sigma + \frac{1}{\sigma} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0(\boldsymbol{\mu})\|^2}{\sigma} \right),$$
$$\mathbf{x}_0(\boldsymbol{\mu}) = (x_0^1, x_0^2, 0.05), \qquad h(\boldsymbol{\mu}) = \eta_0.$$

As in the previous test case, the FE approximation of (2.50) leads to the linear system (2.1), where now the stiffness matrix does not verify the affine property (1.44), due to the nonaffine nature of the coefficient $\alpha(\boldsymbol{\mu})$. In the following, we use either linear ($\mathcal{P}_1$) or quadratic ($\mathcal{P}_2$) basis functions, yielding a dimension of the high-fidelity FE system equal to $N_h = 282'835$ and $N_h = 2'080'389$, respectively. These two problems are run on 64 and 256 cores, respectively. In Figure 2.8, the solutions corresponding to different instances of the parameter are shown.

Numerical results with varying MDEIM affine approximation

The diffusion coefficient $\alpha(\boldsymbol{\mu})$ non affinely depends on $\boldsymbol{\mu}$, therefore in the following we employ MDEIM (see Section 1.3.2) to build the approximated affine decomposition (1.59), up to a certain tolerance δ_{MDEIM}, which is then used for building the RB coarse operators as in (2.29). We then solve the FE linear system with the MSRB-preconditioned FGMRES method up to a tolerance of 10^{-6} on the Euclidean norm of the residual rescaled with the Euclidean norm of the right hand side. We build $L = 4$ RB coarse operators (by adopting a fixed accuracy approach) with tolerance $\delta_{RB,k} = \delta_{RB}$, $k = 0, 1, 2, 3$, using either $\delta_{RB} = 10^{-2}$ or $\delta_{RB} = 10^{-3}$. The offline phase is carried out by using $n_s = 1500$ snapshots for constructing the RB spaces and 500 matrix snapshots for MDEIM; the singular values σ_i, $i = 1, \ldots, 500$ obtained by performing SVD on the matrix collecting these latter snapshots are reported in Figure 2.9. A similar behavior is observed for $\mathcal{P}_1$ and $\mathcal{P}_2$ FE basis functions, in both cases the decrease of the singular values is slow and reaches a plateau for $i \geq 250$.

The results during the online phase are averaged on $N_{\text{onl}} = 250$ instances of $\boldsymbol{\mu}$ randomly selected. To evaluate the accuracy of the RB approximation $\mathbf{V}_k \mathbf{y}_{N_k}(\boldsymbol{\mu})$ of the FE solution $\mathbf{y}_k(\boldsymbol{\mu})$, we compute the average coefficient

$$\eta_{RB}^{(k)}(\boldsymbol{\mu}) = \frac{\left\| \mathbf{v}_{k+\frac{1}{2}}(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu}) \mathbf{V}_k \mathbf{y}_{N_k}(\boldsymbol{\mu}) \right\|_2}{\left\| \mathbf{v}_{k+\frac{1}{2}}(\boldsymbol{\mu}) \right\|_2}, \tag{2.51}$$

which corresponds to the relative FE residual evaluated in the RB solution of problem (2.38) and related to coarse operator k. The value of $\eta_{RB}^{(k)}(\boldsymbol{\mu})$ is strictly related to the tolerances δ_{RB} and δ_{MDEIM} used for constructing the RB spaces (δ_{RB}) and the MDEIM affine approximation (δ_{MDEIM}); for this latter we report results obtained by using $\delta_{\text{MDEIM}} = 10^{-l}$, $l = 4, 5, 6, 7$. The average $\eta_{RB}^{(k)}$, $k = 0, 1, 2, 3$, obtained online is reported for $\mathcal{P}_1$ FE basis functions (Figure 2.10a-2.10b) and $\mathcal{P}_2$ FE basis functions (Figure 2.10c-2.10d) for different values of δ_{RB} and δ_{MDEIM}. As a matter of fact, when using $\delta_{RB} = 10^{-2}$ (Figure (2.10a)-(2.10c)), online the measured accuracy of the RB coarse operators is almost constant by varying δ_{MDEIM}, and a fixed decay (about $4 \cdot 10^{-2}$) is achieved in the FGMRES residual; indeed, since $\delta_{RB} > \delta_{\text{MDEIM}}$, the error generated by the RB coarse operator is mainly caused by the tolerance $\delta_{RB} = 10^{-2}$, which is the leading term between δ_{MDEIM} and itself, and determines the accuracy obtained online, for any δ_{MDEIM}. On the other hand, when using $\delta_{RB} = 10^{-3}$ (Figure (2.10b)-(2.10d)), δ_{RB} is the leading term for all MDEIM tolerances but $\delta_{\text{MDEIM}} = 10^{-4}$, since the accuracy in this latter case (blue lines) is more affected compared to the others. We remark that the same behavior is observed for both $\mathcal{P}_1$ and $\mathcal{P}_2$ FE basis functions.

The number of MDEIM basis functions Q_a, the iteration number It^{onl}, the time $t_{\text{MSRB}}^{\text{onl}}$ to compute the solution during the online phase and the time t_{off} of the offline phase are reported in Table 2.9 ($\delta_{RB} = 10^{-2}$ and $\mathcal{P}_1$ FE), Table 2.10 ($\delta_{RB} = 10^{-3}$ and $\mathcal{P}_1$ FE), Table 2.11 ($\delta_{RB} = 10^{-2}$ and $\mathcal{P}_2$ FE) and Table 2.12 ($\delta_{RB} = 10^{-2}$ and $\mathcal{P}_2$ FE). The online time $t_{\text{MSRB}}^{\text{onl}}$ and the offline time t_{off} are largely affected by δ_{MDEIM}. Indeed, the former is impacted by the assembly of the RB matrices online, while the latter is influenced during the construction of the affine RB decomposition $\mathbf{A}_{N_k}^q$, $q = 1, \ldots, Q_a$ and both these operations have a linear complexity with respect to the number of affine terms Q_a. However, the iteration counts for each simulation confirm that the convergence is always reached in 4 iterations or less, since the accuracy of the RB coarse component

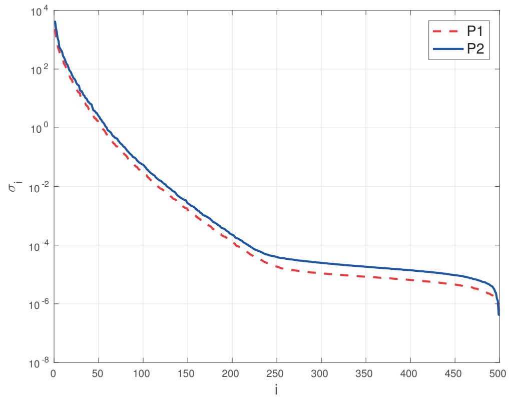
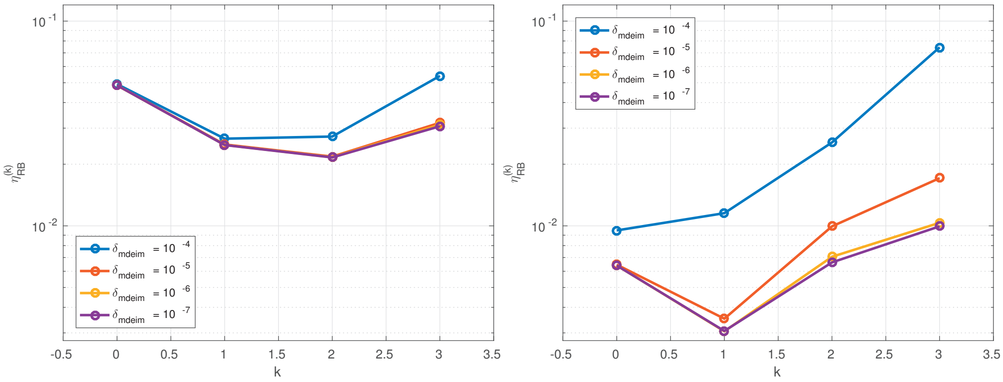
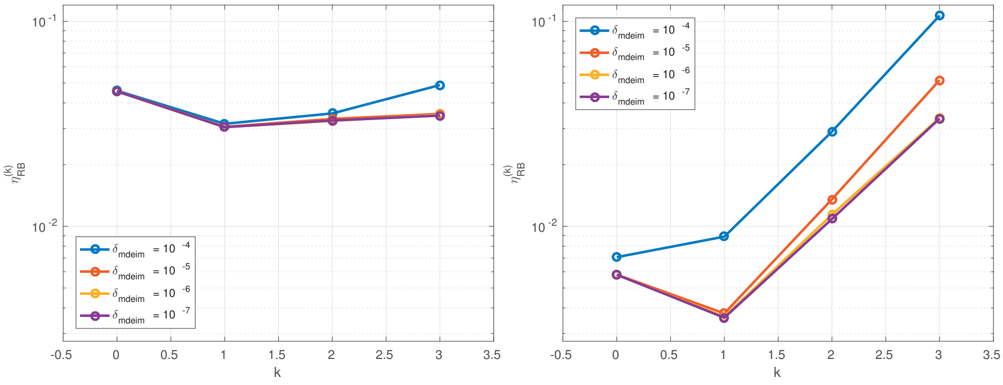


Figure 2.9 – Test case II: singular values σ_i, $i = 1, \ldots, 500$ for $\mathcal{P}_1$ and $\mathcal{P}_2$ FE basis functions computed by POD within MDEIM algorithm.



(a) $\mathcal{P}_1$ FE, $\delta_{RB,k} = 10^{-2}$ $k = 0, 1, 2, 3$.

(b) $\mathcal{P}_1$ FE, $\delta_{RB,k} = 10^{-3}$ $k = 0, 1, 2, 3$.



(c) $\mathcal{P}_2$ FE, $\delta_{RB,k} = 10^{-2}$ $k = 0, 1, 2, 3$.

(d) $\mathcal{P}_2$ FE, $\delta_{RB,k} = 10^{-3}$ $k = 0, 1, 2, 3$.

Figure 2.10 – Test case II: $\eta_{RB}^{(k)}$ as function of k with $\mathcal{P}_1$ (top) and $\mathcal{P}_2$ FE basis functions and $\delta_{RB,k} = \delta_{RB}$, $k = 0, 1, 2, 3$ with $\delta_{RB} = 10^{-2}$ (left) and $\delta_{RB} = 10^{-3}$ (right).

is very mildly affected by the value of δ_{MDEIM}, thanks to the fact that for any chosen value of δ_{MDEIM}, the tolerance δ_{RB} is the leading term in (2.30). As a matter of fact, an extremely accurate approximation of the affine decomposition is not required in practice

Table 2.9 – Test case II: FGMRES results with MSRB preconditioner for $\mathcal{P}_1$ FE basis functions, fixed accuracy approach with $\delta_{RB,k} = 10^{-2}$ and $L = 4$ varying δ_{MDEIM}.

δ_{MDEIM}	Q_a	It^{onl}	$t^{\mathrm{onl}}_{\mathrm{MSRB}}$ (sec)	t_{off} (sec)
1e-04	70	4	0.23	4084.55
1e-05	102	3	0.39	4584.39
1e-06	139	3	0.49	5123.29
1e-07	182	3	0.62	5445.38

Table 2.10 – Test case II: FGMRES results with MSRB preconditioner for $\mathcal{P}_1$ FE basis functions, fixed accuracy approach with $\delta_{RB,k} = 10^{-3}$ and $L = 4$ varying δ_{MDEIM}.

δ_{MDEIM}	Q_a	It^{onl}	$t^{\mathrm{onl}}_{\mathrm{MSRB}}$ (sec)	t_{off} (sec)
1e-04	70	3	0.74	5587.14
1e-05	102	2	0.91	8878.12
1e-06	139	2	1.27	11598.90
1e-07	182	2	1.50	14535.10

Table 2.11 – Test case II: FGMRES results with MSRB preconditioner for $\mathcal{P}_2$ FE basis functions, fixed accuracy approach with $\delta_{RB,k} = 10^{-2}$ and $L = 4$ varying δ_{MDEIM}.

δ_{MDEIM}	Q_a	It^{onl}	$t^{\mathrm{onl}}_{\mathrm{MSRB}}$ (sec)	t_{off} (sec)
1e-04	69	4	0.45	20723.80
1e-05	102	4	0.59	21544.70
1e-06	139	4	0.74	22779.90
1e-07	183	4	0.87	24916.40

Table 2.12 – Test case II: FGMRES results with MSRB preconditioner for $\mathcal{P}_2$ FE basis functions, fixed accuracy approach with $\delta_{RB,k} = 10^{-3}$ and $L = 4$ varying δ_{MDEIM}.

δ_{MDEIM}	Q_a	It^{onl}	$t^{\mathrm{onl}}_{\mathrm{MSRB}}$ (sec)	t_{off} (sec)
1e-04	69	2	1.19	24761.50
1e-05	102	2	1.45	31754.10
1e-06	139	2	1.76	35711.00
1e-07	183	2	2.23	44932.00

and it should be avoided, since it causes a significant overhead in the efficiency of the method without providing any effective benefit. Finally, as a rule of thumb used in the following numerical examples of this thesis, δ_{MDEIM} is chosen trading off between a small enough and negligible term in (2.30), but a large enough value to avoid any efficiency overhead at the same time.

2.5 MSRB preconditioners for parabolic PDEs

In this section we extend the MSRB preconditioning framework developed so far to the case of parametrized unsteady problems; for the sake of synthesis, we focus on the construction and the application in the case of the FGMRES method, however the procedures outlined are directly applicable also in the context of Richardson iterations.

2.5.1 MSRB preconditioner construction

Let us start by considering the sequence of parametrized linear systems to be solved for $n = 0, \dots, N_t - 1$

$$\left(\frac{\alpha_1}{\Delta t} \mathbf{M}(\boldsymbol{\mu}) + \mathbf{A}(\boldsymbol{\mu}) \right) \mathbf{u}^{n+1}(\boldsymbol{\mu}) = \mathbf{f}^{n+1}(\boldsymbol{\mu}) + \frac{1}{\Delta t} \mathbf{M}(\boldsymbol{\mu}) \mathbf{u}^{n,\sigma_1}(\boldsymbol{\mu}), \tag{2.52}$$

with $\mathbf{u}^0(\boldsymbol{\mu}) = \mathbf{u}_0(\boldsymbol{\mu})$, which arises from the BDF time discretization of the algebraic dynamical system (1.17). As a matter of fact, the MSRB preconditioning framework is straightforwardly applicable to problems as (2.52) by considering the matrix $(\frac{\alpha_1}{\Delta t}\mathbf{M}(\boldsymbol{\mu}) + \mathbf{A}(\boldsymbol{\mu}))$ instead of $\mathbf{A}(\boldsymbol{\mu})$ in the construction outlined in the previous sections. Following (2.10), we define our MSRB preconditioner for problem (2.52) as

$$\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu}) = \mathbf{P}^{-1}(\boldsymbol{\mu}) + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\Big(\mathbf{I}_{N_h} - \Big(\frac{\alpha_1}{\Delta t}\mathbf{M}(\boldsymbol{\mu}) + \mathbf{A}(\boldsymbol{\mu})\Big)\mathbf{P}^{-1}(\boldsymbol{\mu})\Big), \tag{2.53}$$

where now $\mathbf{P}(\boldsymbol{\mu})$ is a fine grid nonsingular preconditioner for the left hand side matrix in (2.52). When considering a parameter instance $\boldsymbol{\mu}$ and time step t_n, the FGMRES preconditioning step at iteration k approximately solves the linear system

$$\Big(\frac{\alpha_1}{\Delta t}\mathbf{M}(\boldsymbol{\mu}) + \mathbf{A}(\boldsymbol{\mu})\Big)\mathbf{c}_k^n(\boldsymbol{\mu}) = \mathbf{v}_k^n(\boldsymbol{\mu}), \tag{2.54}$$

where $\mathbf{v}_k^n(\boldsymbol{\mu})$ is the k-th Krylov basis function. Hence, the RB coarse operator $\mathbf{Q}_{N_k}(\boldsymbol{\mu})$ in $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ must be trained to approximate the solution of

$$\Big(\frac{\alpha_1}{\Delta t}\mathbf{M}(\boldsymbol{\mu}) + \mathbf{A}(\boldsymbol{\mu})\Big)\mathbf{y}_k^n(\boldsymbol{\mu}) = \Big(\mathbf{I}_{N_h} - \Big(\frac{\alpha_1}{\Delta t}\mathbf{M}(\boldsymbol{\mu}) + \mathbf{A}(\boldsymbol{\mu})\Big)\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{v}_k^n(\boldsymbol{\mu}). \tag{2.55}$$

As in the steady case, an approximation of the solution of (2.55) is sought in a N_k-dimensional RB space V_{N_k} spanned by solutions of (2.55), this time computed for properly chosen time steps and parameter values.

We consider the projection matrix $\mathbf{V}_k \in \mathbb{R}^{N_h \times N_k}$ which algebraically represents the RB space V_{N_k} and we introduce the RB matrices

$$\mathbf{M}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k^T\mathbf{M}(\boldsymbol{\mu})\mathbf{V}_k \qquad \mathbf{A}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k^T\mathbf{A}(\boldsymbol{\mu})\mathbf{V}_k, \tag{2.56}$$

we then set the RB coarse operator as the RB low-rank solver for (2.55) on the subspace $\mathbf{V}_k$, that is,

$$\mathbf{Q}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k\left(\frac{\alpha_1}{\Delta t}\mathbf{M}_{N_k}(\boldsymbol{\mu}) + \mathbf{A}_{N_k}(\boldsymbol{\mu})\right)^{-1}\mathbf{V}_k^T. \tag{2.57}$$

As a matter of fact, when using the matrix $\mathbf{M}_k^{-1}(\boldsymbol{\mu}) = \mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ defined in (2.53) in the preconditioning step, an approximation $\mathbf{y}_{N_k}^n(\boldsymbol{\mu}) \in \mathbb{R}^{N_k}$ of the solution of problem (2.55) is found by solving

$$\Big(\frac{\alpha_1}{\Delta t}\mathbf{M}_{N_k}(\boldsymbol{\mu}) + \mathbf{A}_{N_k}(\boldsymbol{\mu})\Big)\mathbf{y}_{N_k}^n(\boldsymbol{\mu}) = \mathbf{V}_k^T\mathbf{v}_{k+\frac{1}{2}}^n(\boldsymbol{\mu}), \tag{2.58}$$

where $\mathbf{v}_{k+\frac{1}{2}}^n(\boldsymbol{\mu}) = \big(\mathbf{I}_{N_h} - (\frac{\alpha_1}{\Delta t}\mathbf{M}(\boldsymbol{\mu}) + \mathbf{A}(\boldsymbol{\mu}))\,\mathbf{P}^{-1}(\boldsymbol{\mu})\big)\,\mathbf{v}_k^n(\boldsymbol{\mu})$, and the high-fidelity representation $\mathbf{V}_k\mathbf{y}^n_{N_k}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ of $\mathbf{y}^n_k(\boldsymbol{\mu})$ is computed as

$$\mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{v}^n_{k+\frac{1}{2}}(\boldsymbol{\mu}) = \mathbf{V}_k\left(\frac{\alpha_1}{\Delta t}\mathbf{M}_{N_k}(\boldsymbol{\mu}) + \mathbf{A}_{N_k}(\boldsymbol{\mu})\right)^{-1}\mathbf{V}_k^T\mathbf{v}^n_{k+\frac{1}{2}}(\boldsymbol{\mu}) = \mathbf{V}_k\mathbf{y}^n_{N_k}(\boldsymbol{\mu}). \tag{2.59}$$

For the efficient assembly of the RB matrix $(\frac{\alpha_1}{\Delta t}\mathbf{M}_{N_k}(\boldsymbol{\mu}) + \mathbf{A}_{N_k}(\boldsymbol{\mu}))$, we can rely on the affine decomposition of $\mathbf{M}_{N_k}(\boldsymbol{\mu})$ and $\mathbf{A}_{N_k}(\boldsymbol{\mu})$, which is inherited by the affine property of the FE matrices $\mathbf{M}(\boldsymbol{\mu})$ and $\mathbf{A}(\boldsymbol{\mu})$. As already discussed, such affine decomposition is not trivial to be found as a built-in property in engineering application; should $\mathbf{M}(\boldsymbol{\mu})$ and $\mathbf{A}(\boldsymbol{\mu})$ not verify such assumption, we (separately) employ MDEIM to obtain an approximate one, yielding, for $k = 0, 1, \dots,$

$$\begin{aligned}
\mathbf{M}_{N_k}(\boldsymbol{\mu}) \approx \widetilde{\mathbf{M}}_{N_k}(\boldsymbol{\mu}) &= \sum_{q=1}^{Q_m}\tilde{\Theta}^q_m(\boldsymbol{\mu})\mathbf{V}_k^T\mathbf{M}^q\mathbf{V}_k = \sum_{q=1}^{Q_a}\tilde{\Theta}^q_a(\boldsymbol{\mu})\mathbf{M}^q_{N_k}\\
\mathbf{A}_{N_k}(\boldsymbol{\mu}) \approx \widetilde{\mathbf{A}}_{N_k}(\boldsymbol{\mu}) &= \sum_{q=1}^{Q_a}\tilde{\Theta}^q_a(\boldsymbol{\mu})\mathbf{V}_k^T\mathbf{A}^q\mathbf{V}_k = \sum_{q=1}^{Q_a}\tilde{\Theta}^q_a(\boldsymbol{\mu})\mathbf{A}^q_{N_k}.
\end{aligned}$$

Then, an approximated RB coarse operator is constructed by substituting $\mathbf{A}_{N_k}(\boldsymbol{\mu})$ and $\mathbf{M}_{N_k}(\boldsymbol{\mu})$ in (2.59) with the corresponding MDEIM-approximated ones, that is by setting

$$\mathbf{Q}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k\left(\frac{\alpha_1}{\Delta t}\widetilde{\mathbf{M}}_{N_k}(\boldsymbol{\mu}) + \widetilde{\mathbf{A}}_{N_k}(\boldsymbol{\mu})\right)^{-1}\mathbf{V}_k^T.$$

2.5.2 Algorithmic procedures for unsteady problems

In this section we show how to handle the practical construction of a MSRB preconditioner, and specifically of the RB coarse operators, when dealing with time dependent problems. To this purpose, we could follow either a *fixed accuracy* or a *fixed dimension* approach, that is we can rely on an equivalent of either Algorithm 4 or 5, where both snapshots in time and with respect to $\boldsymbol{\mu}$ are collected. For the sake of synthesis, we report in Algorithm 7 the construction in the fixed dimension case, being the one employed in the numerical experiments presented. We start by computing the (eventually approximated) affine decompositions of $\mathbf{M}(\boldsymbol{\mu})$ and $\mathbf{A}(\boldsymbol{\mu})$ and the solution of (2.52) for a set of selected parameters $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$ and time steps $n = 0, \dots, N_t - 1$; these snapshots are employed to build the first RB projection matrix $\mathbf{V}_0$. Secondly, the iterative relation (2.42), which holds provided $\mathbf{A}(\boldsymbol{\mu})$ is substituted by $(\frac{\alpha_1}{\Delta t}\mathbf{M}(\boldsymbol{\mu}) + \mathbf{A}(\boldsymbol{\mu}))$, is exploited to build the snapshots for the subsequent RB spaces, until the tolerance ε_r is larger than the product of the RB tolerances $\delta_{RB,k}, k = 0, \dots, L-1$. Then, L RB coarse corrections are produced by Algorithm 7.

Algorithm 7 provides a practical way to build the MSRB preconditioner, however it can be too computationally demanding when the time steps number N_t is too large. In fact:

Algorithm 7 MSRB for unsteady problems - Fixed Dimension

```
procedure MSRB-UNSTEADY-FIXEDDIMENSION($\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}, \varepsilon_r, N$)
    Compute an (approximated) affine decomposition of $\mathbf{A}(\boldsymbol{\mu})$ and $\mathbf{M}(\boldsymbol{\mu})$
    Compute $\{\mathbf{u}^n(\boldsymbol{\mu}_1)\}_{n=1}^{N_t}, \dots, \{\mathbf{u}^n(\boldsymbol{\mu}_{n_s})\}_{n=1}^{N_t}$
    Set $\mathbf{S} = [\mathbf{u}^1(\boldsymbol{\mu}_1), \dots, \mathbf{u}^{N_t}(\boldsymbol{\mu}_1), \dots, \mathbf{u}^1(\boldsymbol{\mu}_{n_s}), \dots, \mathbf{u}^{N_t}(\boldsymbol{\mu}_{n_s})]$ and $k = 0$
    while $\prod_k \delta_{RB,k} > \varepsilon_r$ do
        $\mathbf{V}_k = \mathrm{POD}(\mathbf{S}, \mathbf{X}_h, N)$ and $k = k + 1$
        Build RB affine matrices $\{\mathbf{A}_{N_k}^q\}_{q=1}^{Q_a}$ and $\{\mathbf{M}_{N_k}^q\}_{q=1}^{Q_m}$
        Compute new snapshots $\mathbf{y}_k^n(\boldsymbol{\mu}_i)$, $n = 1, \dots, N_t$, $i = 1, \dots, n_s$, with (2.42)
        Set $\mathbf{S} = [\mathbf{y}_k^1(\boldsymbol{\mu}_1), \dots, \mathbf{y}_k^{N_t}(\boldsymbol{\mu}_1), \dots, \mathbf{y}_k^1(\boldsymbol{\mu}_{n_s}), \dots, \mathbf{y}_k^{N_t}(\boldsymbol{\mu}_{n_s})]$
    end while
end procedure
```

1. POD is performed in practice by solving the eigenvalue problem for the correlation matrix (cf. Section 1.3.1), whose cost scales with the cube of its dimension. In general, this does not represent a bottleneck in the case of steady problems, since the correlation matrix has size n_s, which usually is at most of order of $10^2 - 10^3$. On the other hand, this can be an issue when considering time dependent problems, for which the correlation matrix has dimension $N_t \cdot n_s$.
2. the need of storing the vectors $\{\mathbf{y}_k^n(\boldsymbol{\mu}_i)\}_{n=1}^{N_t}$ for $i = 1, \dots, n_s$ and $k = 0, \dots, L-1$, can lead to a huge memory consumption during the construction phase.

Both these bottlenecks can be overcome by adopting a time slab perspective. To this aim, we divide the interval [0, T] in $\mathcal{S}$ time slabs of equal size $\Delta\tau \gg \Delta t$ and we build a different preconditioner for each time slab s, for which a sequence of L_s RB spaces $\mathbf{V}_{k,s}$, $k = 0, \dots, L_s - 1$, and the corresponding coarse operators are constructed

$$\mathbf{A}_{N_k,s}(\boldsymbol{\mu}) = \mathbf{V}_{k,s}^T \mathbf{A}(\boldsymbol{\mu}) \mathbf{V}_{k,s}, \qquad \mathbf{M}_{N_k,s}(\boldsymbol{\mu}) = \mathbf{V}_{k,s}^T \mathbf{M}(\boldsymbol{\mu}) \mathbf{V}_{k,s}. \tag{2.60}$$

Then, the preconditioner for time slab s is defined as

$$\mathbf{Q}_{\mathrm{MSRB},k,s}(\boldsymbol{\mu}) = \mathbf{P}^{-1}(\boldsymbol{\mu}) + \mathbf{Q}_{N_k,s}(\boldsymbol{\mu}) \Big(\mathbf{I}_{N_h} - \Big(\frac{\alpha_1}{\Delta t} \mathbf{M}(\boldsymbol{\mu}) + \mathbf{A}(\boldsymbol{\mu}) \Big) \mathbf{P}^{-1}(\boldsymbol{\mu}) \Big), \tag{2.61}$$

where

$$\mathbf{Q}_{N_k,s}(\boldsymbol{\mu}) = \mathbf{V}_{k,s} \left(\frac{\alpha_1}{\Delta t} \mathbf{M}_{N_k,s}(\boldsymbol{\mu}) + \mathbf{A}_{N_k,s}(\boldsymbol{\mu}) \right)^{-1} \mathbf{V}_{k,s}^T. \tag{2.62}$$

We finally define as $\mathcal{N}_\tau = \Delta\tau / \Delta t$ the number of time steps contained in a time slab and express any time t_n in terms of the time slab s_n it belongs, that is,

$$s_n = \lfloor \frac{n}{\mathcal{N}_\tau} \rfloor, \qquad t_n = s_n \Delta\tau + (n - s_n \mathcal{N}_\tau) \Delta t, \qquad n = 0, \dots, N_t.$$

At time t_n, $n = 1, \dots, N_t$, the preconditioner built for the time slab s_n is then employed. The procedure for the time slab construction is reported in Algorithm 8, which essentially repeats Algorithm 7 for each time slab $s = 0, \dots, \mathcal{S} - 1$ and generates L_s RB coarse operators. We highlight that in general, the preconditioners are independent from one time slab to the other, including, for instance, a different number of RB coarse corrections L_s. By employing Algorithm 8, a more accurate preconditioner (in time) is constructed and the dimension of the POD correlation matrices used to build the RB spaces is limited to $\mathcal{N}_\tau \cdot n_s$; in practice, the number of time slabs $\mathcal{N}_\tau$ is chosen such that this dimension does not exceed a certain maximum (e.g., few thousands).

Algorithm 8 MSRB for unsteady problems - Fixed Dimension with time slabs

```
procedure MSRB-SLABS-FIXEDDIMENSION($\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}, \varepsilon_r, N, \mathcal{S}$)
    Compute an approximated affine decomposition of $\mathbf{A}(\boldsymbol{\mu})$ and $\mathbf{M}(\boldsymbol{\mu})$ and set $k = 0$
    for $s = 0, \dots, \mathcal{S} - 1$ do
        Compute $\{\mathbf{u}^n(\boldsymbol{\mu}_1)\}_{n=\mathcal{N}_\tau s+1}^{\mathcal{N}_\tau(s+1)}, \dots, \{\mathbf{u}^n(\boldsymbol{\mu}_{n_s})\}_{n=\mathcal{N}_\tau s+1}^{\mathcal{N}_\tau(s+1)}$
        Set $\mathbf{S} = [\mathbf{u}^{\mathcal{N}_\tau s+1}(\boldsymbol{\mu}_1), \dots, \mathbf{u}^{\mathcal{N}_\tau(s+1)}(\boldsymbol{\mu}_1), \dots, \mathbf{u}^{\mathcal{N}_\tau s+1}(\boldsymbol{\mu}_{n_s}), \dots, \mathbf{u}^{\mathcal{N}_\tau(s+1)}(\boldsymbol{\mu}_{n_s})]$
        while $\prod_k \delta_{RB,k} > \varepsilon_r$ do
            Build RB space $\mathbf{V}_{k,s} = \mathrm{POD}(\mathbf{S}, \mathbf{X}_h, N)$
            Build RB affine matrices $\{\mathbf{A}^q_{N_k,s}\}_{q=1}^{Q_a}$ and $\{\mathbf{M}^q_{N_k,s}\}_{q=1}^{Q_m}$ and set $k = k + 1$
            Compute $\mathbf{y}^n_k(\boldsymbol{\mu}_i)$, $n = \mathcal{N}_\tau s + 1, \dots, \mathcal{N}_\tau(s+1)$, $i = 1, \dots, n_s$ with (2.42)
            Set $\mathbf{S} = [\mathbf{y}_k^{\mathcal{N}_\tau s+1}(\boldsymbol{\mu}_1), \dots, \mathbf{y}_k^{\mathcal{N}_\tau s+1}(\boldsymbol{\mu}_{n_s}), \dots, \mathbf{y}_k^{\mathcal{N}_\tau(s+1)}(\boldsymbol{\mu}_{n_s})]$
        end while
    end for
end procedure
```

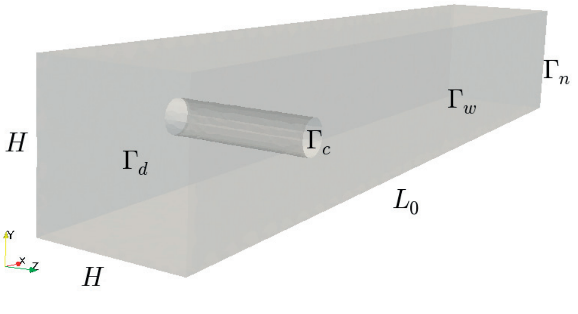


Figure 2.11 – Test case III: computational domain Ω with boundary flags, with $H = 0.41$m, $L_0 = 2.5$m. The cylinder has radius $r = 0.05$m, its center is distant 0.5m from Γ_d and 0.2m from the bottom face.

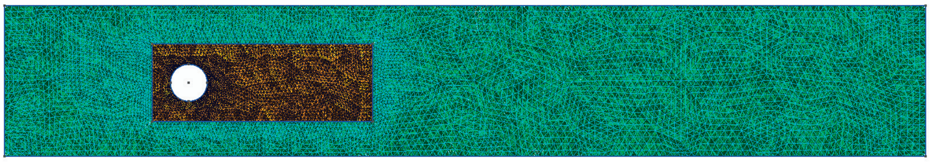


Figure 2.12 – Test case III: computational domain Ω obtained with Gmsh [Geuzaine and Remacle, 2009]. A refinement has been applied in the (orange) area around the cylinder.

2.6 Numerical results for parabolic problems

2.6.1 Test case III: three-dimensional heat transfer past a cylinder

The third test case considered is related to unsteady heat transfer past a cylinder.

Test case setting

We consider the domain in Figure 2.11 and the temperature $C(\boldsymbol{\mu})$ evolution to be governed by the following advection diffusion equation

$$\begin{cases} \dfrac{\partial C(\boldsymbol{\mu})}{\partial t} - \alpha_c \Delta C(\boldsymbol{\mu}) + \vec{v} \cdot \nabla C(\boldsymbol{\mu}) = 0 & \text{in } \Omega \times (0,T) \\ C(\boldsymbol{\mu}) = 0 & \text{on } \Gamma_d \times (0,T) \\ C(\boldsymbol{\mu}) = C_d & \text{on } \Gamma_c \times (0,T) \\ \alpha_c \nabla C(\boldsymbol{\mu}) \cdot \mathbf{n} = 0 & \text{on } \Gamma_n \cup \Gamma_w \times (0,T) \\ C(0;\boldsymbol{\mu}) = 0 & \text{in } \Omega \times (0,T), \end{cases} \tag{2.63}$$

where $T = 10$ seconds, the advection field $\vec{v}$ is the solution of an underlying steady Navier-Stokes problem describing the dynamics of the fluid velocity and pressure, α_c is the thermal diffusivity and $C_d \in \mathbb{R}$ is the given Dirichlet datum. The problem is parametrized with respect to the values of α_c and C_d, that is $\boldsymbol{\mu} = (\alpha_c, C_d) \in [10^{-3}, 10^{-2}] \times [1, 10]$. Regarding the FE discretization, we employ the computational grid shown in Figure 2.12, opting for a SUPG formulation, since the problem is highly transport dominated, see e.g. [Quarteroni and Valli, 2008] for further details on this stabilization technique. We introduce a triangulation $\mathcal{T}_h$ of Ω and a lifting function which takes into account the nonhomogeneous Dirichlet condition. Then, we obtain the following semi-dicrete problem, which reads: for all $t \in (0,T]$, find $C_h(\boldsymbol{\mu}) \in X_h$ such that

$$m_h(C_h(\boldsymbol{\mu}), w_h; t; \boldsymbol{\mu}) + a_h(C_h(\boldsymbol{\mu}), w_h; t; \boldsymbol{\mu}) = g_h(w_h; \boldsymbol{\mu}) \qquad \forall w_h \in X_h. \tag{2.64}$$

For any $C_h, w_h \in X_h$, we define

$$m_h(C_h, w_h; \boldsymbol{\mu}) = \Big(\frac{\partial C_h}{\partial t}, w_h\Big)_{L^2(\Omega)} + \sum_{K \in \mathcal{T}_h} \Big(\frac{\partial C_h}{\partial t}, \tau_K(\boldsymbol{\mu}) \vec{v}_h \cdot \nabla w_h\Big)_{L^2(K)} \tag{2.65}$$

$$\begin{aligned} a_h(C_h, w_h; \boldsymbol{\mu}) = {} & \Big(\alpha_c \nabla C_h, \nabla w_h\Big)_{L^2(\Omega)} + \Big(\vec{v}_h \cdot \nabla C_h, w_h\Big)_{L^2(\Omega)} \\ & + \sum_{K \in \mathcal{T}_h} \Big(- \alpha_c \Delta C_h + \vec{v}_h \cdot \nabla C_h, \tau_K(\boldsymbol{\mu}) \vec{v}_h \cdot \nabla w_h\Big)_{L^2(K)}. \end{aligned} \tag{2.66}$$

In (2.64), $g_h(w_h; \boldsymbol{\mu})$ encodes the right hand side and the lifting function, while the SUPG parameter $\tau_K(\boldsymbol{\mu})$ is defined as

$$\tau_K(\boldsymbol{\mu}) = \Big(\frac{4}{\Delta t^2} + \vec{v}_h \cdot \mathbf{G}_K \vec{v}_h + \alpha_c^2 \mathbf{G}_K : \mathbf{G}_K\Big)^{-1/2}, \tag{2.67}$$

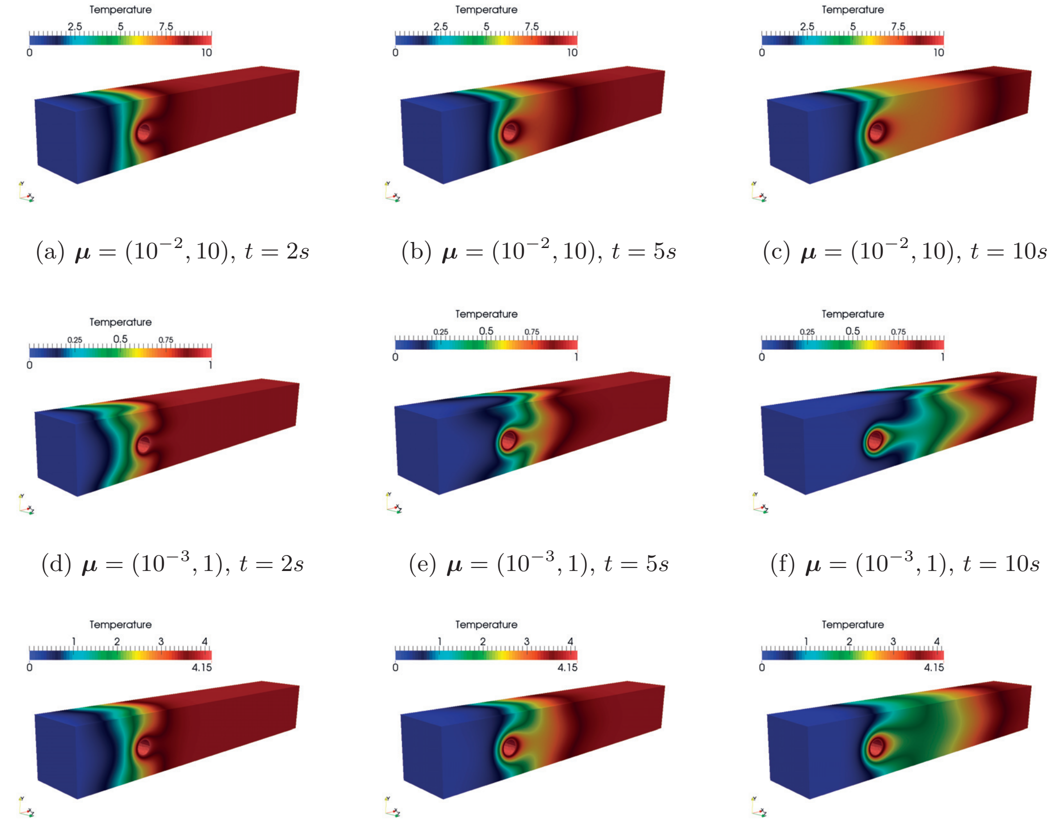
where $\mathbf{G}_K$ is the covariant metric tensor of the computational domain defined as

$$(\mathbf{G}_K)_{ij} = \sum_{l=1}^{d} \frac{\partial \xi_j}{\partial x_i} \frac{\partial \xi_j}{\partial x_j}, \qquad i, j = 1, \dots, d; \tag{2.68}$$

see [Bazilevs et al., 2007] for additional details.

From an algebraic standpoint, problem (2.64) is equivalent to the N_h-dimensional system of ordinary differential equations in (1.17). Notice, however, that in the considered case the mass matrix $\mathbf{M}(\boldsymbol{\mu})$ and the stiffness matrix $\mathbf{A}(\boldsymbol{\mu})$ take into account the stabilization terms, which arise due to the SUPG-FE formulation. As a matter of fact, the SUPG formulation yields two nonaffine matrices $\mathbf{M}(\boldsymbol{\mu})$, $\mathbf{A}(\boldsymbol{\mu})$ even if the starting differential problem (2.63) featured an affine parameter dependence.

We employ a computational grid with 802’048 elements and 147’558, with a refinement near the cylinder, cf. Figure 2.11, and $\mathcal{P}_2$ polynomial basis functions, leading to $N_h = 1'124'412$ degrees of freedom. The time discretization is carried out by employing the BDF1 method, that is by the backward Euler method, with $\Delta t = 0.01$. Examples of solutions are shown in Figure 2.13 for selected values of the parameter and times: the value of the parameter can largely influence the solution at a given time. The results are computed up to a tolerance equal to 10^{-7} on the relative residual as stopping criterion for the FGMRES method and using $N_{\text{core}} = 128, 256, 512$ cores.



(a) $\boldsymbol{\mu} = (10^{-2}, 10)$, $t = 2s$ (b) $\boldsymbol{\mu} = (10^{-2}, 10)$, $t = 5s$ (c) $\boldsymbol{\mu} = (10^{-2}, 10)$, $t = 10s$

(d) $\boldsymbol{\mu} = (10^{-3}, 1)$, $t = 2s$ (e) $\boldsymbol{\mu} = (10^{-3}, 1)$, $t = 5s$ (f) $\boldsymbol{\mu} = (10^{-3}, 1)$, $t = 10s$

(g) $\boldsymbol{\mu} = (6.85 \cdot 10^{-3}, 4.15)$, $t = 2s$ (h) $\boldsymbol{\mu} = (6.85 \cdot 10^{-3}, 4.15)$, $t = 5s$ (i) $\boldsymbol{\mu} = (6.85 \cdot 10^{-3}, 4.15)$, $t = 10s$

Figure 2.13 – Test case III: example of solutions for different parameter values.

Table 2.13 – Test case III: number of RB coarse operators L_s computed by Algorithm 8 to reach a tolerance $\varepsilon_r = 10^{-9}$. They are the same for all the values of N_{core}.

s	0	1	2	3	4	5	6	7	8	9
L_s	13	9	8	7	7	6	5	5	4	5

Numerical results

A block Jacobi preconditioner is again chosen as fine grid component for the MSRB preconditioner, that is $\mathbf{P}(\boldsymbol{\mu}) = \mathbf{P}_{\text{BJ}}(\boldsymbol{\mu})$, and the offline phase is carried out with $n_s = 25$ parameters (chosen randomly) for the construction of the MSRB precondtioner; such leading to 25000 snapshots in total. To avoid this computational load, we divide the interval $(0, T)$ in $\mathcal{S} = 10$ time slabs, each of length $\Delta\tau = 1$, and build a MSRB preconditioner for each time slab, such that each local-in-time construction is performed with 2500 snapshots, to reach $\varepsilon_r = 10^{-9}$ with $N_k = 15, k = 0, \dots, L_s - 1$, $s = 0, \dots, 9$. On the other hand, MDEIM, with tolerance $\delta_{\text{MDEIM}} = 10^{-7}$ and 75 initial matrix snapshots, is used to determine an affine approximated decomposition of the matrices $\mathbf{M}(\boldsymbol{\mu})$, $\mathbf{A}(\boldsymbol{\mu})$, yielding 1 and 4 affine terms, respectively. The SVDs used within MDEIM for the two affine approximations are reported in Figure 2.14, where the rapid decay of the singular values can be observed.

The number of RB spaces L_s produced by Algorithm 8 as function of the time slab s is reported in Table 2.13 and ranges from 4 to 13, decreasing with time, due to the fact the solution reaches the steady state (notice that source and boundary data are modeled time independent). This is confirmed by the decay of the singular values in the POD to build the RB projection matrices $\mathbf{V}_{k,s}$, $k = 0, \dots, L_s - 1$, $s = 0, \dots, 9$, reported in Figure 2.15a, 2.15b for the slabs $s = 0$ and $s = 5$, respectively. The singular values for slab $s = 0$ decrease slower than the ones for slab $s = 5$ when a fixed k for the two charts is considered, leading to the construction of a larger amount of spaces for the first slab, since we are employing a fixed dimension approach. The MSRB preconditioners are then used

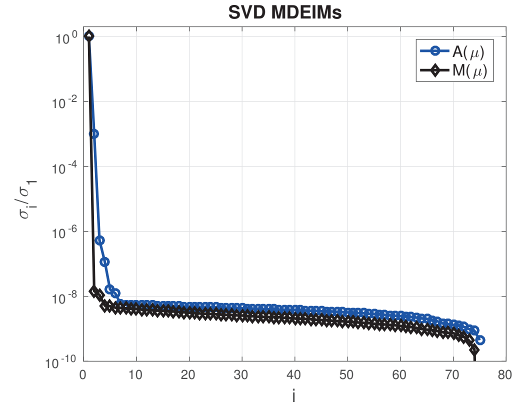
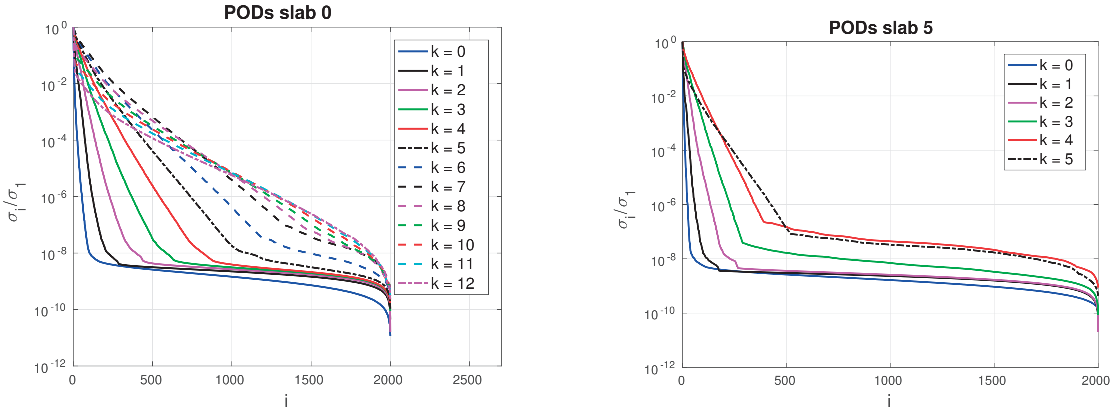


Figure 2.14 – SVD of MDEIM computed for the construction of $\mathbf{A}(\boldsymbol{\mu})$ and $\mathbf{M}(\boldsymbol{\mu})$.



(a) POD for consructing the RB projection matrices $\mathbf{V}_{k,0}$.

(b) POD for consructing the RB projection matrices $\mathbf{V}_{k,5}$.

Figure 2.15 – Test case III: POD used to build the RB projection matrices $\mathbf{V}_{k,0}$ (bottom, left) and $\mathbf{V}_{k,5}$ (bottom, right); the singular values for slab $s = 0$ decrease slower than the ones for $s = 5$ when the same k for the two charts is considered.

online on 100 parameters different from the ones used during the offline phase. We report its average performance in Table 2.14 and we compare the obtained results with the ones provided by the AMG preconditioner from `ML` package of `Trilinos`. The number of iterations needed to reach convergence is on average 4 or 5 for the MSRB-preconditioned FGMRES, that is about 3 times smaller than the one needed by AMG-preconditioned GMRES. Furthermore, the corresponding computational time decreases as the number of cores N_{core} grows, entailing good performances if compared with the AMG preconditioner, however not featuring optimally scalable performances.

The time t_{off} required for the construction of the MSRB preconditioners and the break-even point BEP are reported as well. In particular, the latter accounts for the computational time required to repay the offline phase with respect to using the reference AMG preconditioner, and is defined as

$$\text{BEP} = \frac{t_{\text{off}}}{t_{\text{GML}} - t_{\text{MSRB}}^{\text{onl}}},$$

and is a decreasing function of N_{core}.

Table 2.14 – Test case III: results with MSRB preconditioner with fixed dimension approach ($N = 15$) and $\mathcal{S} = 10$ time slabs. Computational times are expressed in seconds.

N_{core}	$t_{\text{MSRB}}^{\text{onl}}(It)$	$t_{\text{GML}}(It)$	t_{off}	BEP
128	149.93 (4)	268.03 (12)	45312.9	369
256	100.97 (5)	237.12 (14)	31030.7	228
512	90.19 (5)	290.01 (13)	27107.6	136

3 RB methods and multi space RB preconditioners for parametrized Stokes equations

In this chapter we consider the Stokes equations as relevant example of parametrized linear saddle-point problem. We propose a new way to solve the Stokes problem in the framework of RB methods, which can be regarded as an algebraic LSRB method; for this reason we refer to it as aLSRB method. The aLSRB method extends and improves the existing RB methods for Stokes equations in several directions: it does not need an enrichment of the velocity space, it exploits suitable approximations of the matrix-norm $\mathbf{X}_h(\boldsymbol{\mu})$ in the definition of the supremizing operator, the resulting aLSRB problem is *inf-sup* stable and in the case of geometrical parameters it does not require the use of an analytical map between a reference domain and the original domain $\Omega(\boldsymbol{\mu})$. We analyze it theoretically and assess its numerical performance on some problems of interest.

Then, we turn our attention to the MSRB preconditioning strategy for the Stokes equations, and propose two different variants for the construction of the RB coarse operators. A first strategy exploits an enriched velocity Galerkin-RB formulation; alternatively, we employ the newly proposed aLSRB method as coarse operator. We verify the nonsingularity of the proposed preconditioner and formulate the algorithms for its construction and application. We finally test its capabilities with Stokes problems in parametrized geometries and compare the results with the ones obtained using state-of-the-art techniques. We refer to [Dal Santo et al., 2017b, Dal Santo et al., 2018b] for additional details on the topic.

3.1 Parametrized Stokes equations

In this section we introduce the Stokes equations in parametrized domains, together with their weak formulation and the resulting FE approximation, specifically focusing on a parametrized geometric formulation. This will be useful in cardiovascular applications.

However, the Given a $\boldsymbol{\mu}-$dependent domain $\Omega(\boldsymbol{\mu}) \subset \mathbb{R}^d$, $d = 2, 3$, such that, for any $\boldsymbol{\mu} \in \mathcal{D}$, $\partial\Omega(\boldsymbol{\mu}) = \Gamma_{out}(\boldsymbol{\mu}) \cup \Gamma_{in}(\boldsymbol{\mu}) \cup \Gamma_w(\boldsymbol{\mu})$ and $\mathring{\Gamma}_{out}(\boldsymbol{\mu}) \cap \mathring{\Gamma}_{in}(\boldsymbol{\mu}) = \mathring{\Gamma}_{w}(\boldsymbol{\mu}) \cap \mathring{\Gamma}_{in}(\boldsymbol{\mu}) = \mathring{\Gamma}_{out}(\boldsymbol{\mu}) \cap \mathring{\Gamma}_{w}(\boldsymbol{\mu}) = \emptyset$, the Stokes equations read

$$\begin{cases} -\nu(\boldsymbol{\mu})\Delta\vec{u}(\boldsymbol{\mu}) + \nabla p(\boldsymbol{\mu}) = \vec{f}(\boldsymbol{\mu}) & \text{in } \Omega(\boldsymbol{\mu}) \\ \nabla \cdot \vec{u}(\boldsymbol{\mu}) = 0 & \text{in } \Omega(\boldsymbol{\mu}) \\ \vec{u}(\boldsymbol{\mu}) = \vec{g}_D(\boldsymbol{\mu}) & \text{on } \Gamma_{in}(\boldsymbol{\mu}) \\ \vec{u}(\boldsymbol{\mu}) = \vec{0} & \text{on } \Gamma_w(\boldsymbol{\mu}) \\ -p(\boldsymbol{\mu})\vec{n}(\boldsymbol{\mu}) + \nu(\boldsymbol{\mu})\dfrac{\partial\vec{u}(\boldsymbol{\mu})}{\partial\vec{n}(\boldsymbol{\mu})} = \vec{g}_N(\boldsymbol{\mu}) & \text{on } \Gamma_{out}(\boldsymbol{\mu}), \end{cases} \tag{3.1}$$

where $(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu}))$ denote the velocity and the pressure of a viscous incompressible Newtonian fluid with viscosity $\nu(\boldsymbol{\mu})$, respectively. We introduce a regular enough lifting function $\vec{r}_{\vec{g}_D}(\boldsymbol{\mu}) \in \left(H^1(\Omega(\boldsymbol{\mu}))\right)^d$ and the following $\boldsymbol{\mu}$-dependent Hilbert spaces

$$\begin{aligned} &V(\boldsymbol{\mu}) = \left\{\vec{v} \in \left(H^1(\Omega(\boldsymbol{\mu}))\right)^d : \vec{v}\big|_{\Gamma_w(\boldsymbol{\mu})} = \vec{v}\big|_{\Gamma_{in}(\boldsymbol{\mu})} = \vec{0}\right\}, \\ &Q(\boldsymbol{\mu}) = L^2(\Omega(\boldsymbol{\mu})) \quad \text{ or } \quad Q(\boldsymbol{\mu}) = L^2_0(\Omega(\boldsymbol{\mu})) \ \text{ if } \ \Gamma_{out}(\boldsymbol{\mu}) = \emptyset, \end{aligned}$$

equipped with the scalar products (and the corresponding induced norms) $(\cdot,\cdot)_{V(\boldsymbol{\mu})} = (\cdot,\cdot)_{(H^1_0(\Omega(\boldsymbol{\mu})))^d}$ and $(\cdot,\cdot)_{Q(\boldsymbol{\mu})} = (\cdot,\cdot)_{L^2(\Omega(\boldsymbol{\mu}))}$. For a given $\boldsymbol{\mu} \in \mathcal{D}$, the weak formulation of problem (3.1) reads: find $(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu})) \in V(\boldsymbol{\mu}) \times Q(\boldsymbol{\mu})$ such that

$$\begin{cases} d(\vec{u}(\boldsymbol{\mu}), \vec{v}; \boldsymbol{\mu}) + b(\vec{v}, p(\boldsymbol{\mu}); \boldsymbol{\mu}) = f(\vec{v}; \boldsymbol{\mu}) & \forall \vec{v} \in V(\boldsymbol{\mu}) \\ b(\vec{u}(\boldsymbol{\mu}), q; \boldsymbol{\mu}) = -b(\vec{r}_{\vec{g}_D}(\boldsymbol{\mu}), q; \boldsymbol{\mu}) & \forall q \in Q(\boldsymbol{\mu}), \end{cases} \tag{3.2}$$

where $\vec{r}_{\vec{g}_D}(\boldsymbol{\mu})$ is a regular enough lifting function and we define the forms in (3.2) for $\vec{u}, \vec{v} \in V(\boldsymbol{\mu})$, $q \in Q(\boldsymbol{\mu})$ as

$$\begin{aligned} d(\vec{u}, \vec{v}; \boldsymbol{\mu}) &= \int_{\Omega(\boldsymbol{\mu})} \nu(\boldsymbol{\mu})\nabla\vec{u} : \nabla\vec{v} d\Omega(\boldsymbol{\mu}), \\ b(\vec{v}, q; \boldsymbol{\mu}) &= -\int_{\Omega(\boldsymbol{\mu})} q\nabla\cdot\vec{v} d\Omega(\boldsymbol{\mu}) \\ f(\vec{v}; \boldsymbol{\mu}) &= \int_{\Omega(\boldsymbol{\mu})} \vec{f}(\boldsymbol{\mu}) \cdot \vec{v} d\Omega(\boldsymbol{\mu}) + \int_{\Gamma_{out}(\boldsymbol{\mu})} \vec{g}_N(\boldsymbol{\mu}) \cdot \vec{v} d\Gamma_{out}(\boldsymbol{\mu}) - d(\vec{r}_{\vec{g}_D}(\boldsymbol{\mu}), \vec{v}; \boldsymbol{\mu}). \end{aligned}$$

Problem (3.2) can be written as a symmetric non-coercive problem, provided we define the space $X(\boldsymbol{\mu}) = V(\boldsymbol{\mu}) \times Q(\boldsymbol{\mu})$, equipped with the scalar product

$$((\vec{u}, p), (\vec{v}, q))_{X(\boldsymbol{\mu})} = (\vec{u}, \vec{v})_{V(\boldsymbol{\mu})} + (p, q)_{Q(\boldsymbol{\mu})}, \qquad (\vec{u}, p), (\vec{v}, q) \in X(\boldsymbol{\mu}),$$

and the norm

$$\|(\vec{v}, q)\|_{X(\boldsymbol{\mu})} = \sqrt{((\vec{v}, q), (\vec{v}, q))_{X(\boldsymbol{\mu})}}, \qquad (\vec{v}, q) \in X(\boldsymbol{\mu}).$$

System (3.2) can thus be equivalently written as: find $\vec{z}(\boldsymbol{\mu}) \in X(\boldsymbol{\mu})$ such that

$$a(\vec{z}(\boldsymbol{\mu}), \vec{w}; \boldsymbol{\mu}) = f(\vec{w}; \boldsymbol{\mu}) \qquad \forall \vec{w} \in X(\boldsymbol{\mu}). \tag{3.3}$$

The well-posedness of problem (3.3) is ensured according to the general theory of saddle-point problems, see, e.g., [Boffi et al., 2013, Brezzi, 1974, Brezzi and Bathe, 1990].

3.1.1 Finite element approximation of the Stokes equations

All the preconditioning and ROM techniques considered hereafter for the efficient solution of the parametrized problem (3.3) hinge upon a high-fidelity finite element approximation, which represent a successful technique to handle the numerical approximation of (3.1), see e.g. [Elman et al., 2005, Quarteroni and Valli, 2008, Girault and Raviart, 2012, Temam, 1984]. Since we consider a domain deformation dependent on $\boldsymbol{\mu}$; the corresponding meshes are also taken as a deformation of a reference mesh, hence not affecting the mesh connectivity, that is without changing the topology of the degrees of freedom.

Let us denote by $V_h(\boldsymbol{\mu})$ and $Q_h(\boldsymbol{\mu})$ two finite dimensional FE spaces of dimension N_h^u and N_h^p, respectively, with $V_h(\boldsymbol{\mu}) \subset V$ and $Q_h(\boldsymbol{\mu}) \subset Q$. Moreover, let us set $X_h(\boldsymbol{\mu}) = V_h(\boldsymbol{\mu}) \times Q_h(\boldsymbol{\mu})$ with dimension $N_h = N_h^u + N_h^p$. The FE approximation of problem (3.3) reads: find $\vec{z}_h(\boldsymbol{\mu}) \in X_h(\boldsymbol{\mu})$ such that

$$a(\vec{z}_h(\boldsymbol{\mu}), \vec{w}_h; \boldsymbol{\mu}) = f(\vec{w}_h; \boldsymbol{\mu}) \qquad \forall \vec{w}_h \in X_h(\boldsymbol{\mu}). \tag{3.4}$$

We further assume that the following $\boldsymbol{\mu}-$uniform *inf-sup* condition holds: there exists a positive constant β_h^{min}, independent of $\boldsymbol{\mu}$, such that

$$\beta_h^{min}(\boldsymbol{\mu}) = \inf_{\vec{z}_h \in X_h(\boldsymbol{\mu})} \sup_{\vec{w}_h \in X_h(\boldsymbol{\mu})} \frac{a(\vec{z}_h, \vec{w}_h; \boldsymbol{\mu})}{\|\vec{z}_h\|_{X(\boldsymbol{\mu})} \|\vec{w}_h\|_{X(\boldsymbol{\mu})}} \geq \beta_h^{min} \qquad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{3.5}$$

A couple of FE spaces which fulfills condition (3.5) is given by $\mathcal{P}_2 - \mathcal{P}_1$ (Taylor-Hood) finite elements basis functions, for velocity and pressure, respectively. Condition (3.5) ensures the stability of problem (3.4). In algebraic form, problem (3.4) can be rewritten as

$$\mathbf{A}(\boldsymbol{\mu})\mathbf{z}(\boldsymbol{\mu}) = \mathbf{g}(\boldsymbol{\mu}), \tag{3.6}$$

featuring the saddle-point structure (1.13), that is,

$$\mathbf{A}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{D}(\boldsymbol{\mu}) & \mathbf{B}^T(\boldsymbol{\mu}) \\ \mathbf{B}(\boldsymbol{\mu}) & O \end{bmatrix}, \qquad \mathbf{z}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{u}(\boldsymbol{\mu}) \\ \mathbf{p}(\boldsymbol{\mu}) \end{bmatrix}, \qquad \mathbf{g}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{f}(\boldsymbol{\mu}) \\ \mathbf{r}(\boldsymbol{\mu}) \end{bmatrix}, \tag{3.7}$$

where $\mathbf{A}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ and $\mathbf{z}(\boldsymbol{\mu})$, $\mathbf{g}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$. More precisely $\mathbf{D}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^u \times N_h^u}$, $\mathbf{B}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^p \times N_h^u}$, $\mathbf{f}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^u}$ and finally $\mathbf{r}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^p}$. In particular, by introducing the basis functions $\{\phi_i^{\vec{u}}(\boldsymbol{\mu})\}_{i=1}^{N_h^u}$ and $\{\phi_i^{p}(\boldsymbol{\mu})\}_{i=1}^{N_h^p}$ of $V_h(\boldsymbol{\mu})$ and $Q_h(\boldsymbol{\mu})$, respectively, so that $V_h(\boldsymbol{\mu}) = span\{\phi_i^{\vec{u}}(\boldsymbol{\mu}),\, i = 1, \dots, N_h^u\}$ and $Q_h(\boldsymbol{\mu}) = span\{\phi_i^{p}(\boldsymbol{\mu}),\, i = 1, \dots, N_h^p\}$, we have that the block matrices are defined as

$$\begin{aligned} \left(\mathbf{D}(\boldsymbol{\mu})\right)_{ij} &= d(\phi_j^{\vec{u}}(\boldsymbol{\mu}), \phi_i^{\vec{u}}(\boldsymbol{\mu}); \boldsymbol{\mu}) \qquad \forall i, j = 1, \dots, N_h^u \\ \left(\mathbf{B}(\boldsymbol{\mu})\right)_{ij} &= b(\phi_j^{\vec{u}}(\boldsymbol{\mu}), \phi_i^{p}(\boldsymbol{\mu}); \boldsymbol{\mu}) \qquad \forall i = 1, \dots, N_h^p,\, j = 1, \dots, N_h^u \end{aligned} \tag{3.8}$$

and the block vectors as

$$\left(\mathbf{f}(\boldsymbol{\mu})\right)_i = f(\phi_i^{\vec{u}}(\boldsymbol{\mu}); \boldsymbol{\mu}) \quad \forall i = 1, \dots, N_h^u \tag{3.9}$$

$$\left(\mathbf{r}(\boldsymbol{\mu})\right)_i = -b(\vec{r}_{\vec{g}_D}(\boldsymbol{\mu}), \phi_i^{p}(\boldsymbol{\mu}); \boldsymbol{\mu}) \quad \forall i = 1, \dots, N_h^p. \tag{3.10}$$

The solution of system (3.6) exploits suitable preconditioned iterative methods. As discussed in Section 1.2.1, several techniques relying on domain decomposition, multilevel methods and block factorizations have been proposed as preconditioners, we refer in particular to [Elman et al., 2005, Rehman et al., 2011, ur Rehman et al., 2009, Segal et al., 2010, Toselli and Widlund, 2005] and references therein. Condition (3.5) can be algebraically expressed as follows: there exists $\beta_h^{min} > 0$ such that

$$\beta_h^{min}(\boldsymbol{\mu}) = \inf_{\vec{z}_h \in \mathbb{R}^{N_h}} \sup_{\vec{w}_h \in \mathbb{R}^{N_h}} \frac{\mathbf{w}^T \mathbf{A}(\boldsymbol{\mu}) \mathbf{z}}{\|\mathbf{z}\|_{\mathbf{X}_h(\boldsymbol{\mu})} \|\mathbf{w}\|_{\mathbf{X}_h(\boldsymbol{\mu})}} \geq \beta_h^{min} \qquad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{3.11}$$

where the symmetric and positive definite matrix $\mathbf{X}_h(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ encodes the scalar product $(\cdot, \cdot)_{X(\boldsymbol{\mu})}$ on the FE space $X_h(\boldsymbol{\mu})$, and is built as a block diagonal matrix of the form

$$\mathbf{X}_h(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{X}_u(\boldsymbol{\mu}) & 0 \\ 0 & \mathbf{X}_p(\boldsymbol{\mu}) \end{bmatrix}; \tag{3.12}$$

$\mathbf{X}_u(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^u \times N_h^u}$ and $\mathbf{X}_p(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^p \times N_h^p}$ encode the scalar products on the spaces $V_h(\boldsymbol{\mu})$ and $Q_h(\boldsymbol{\mu})$, respectively. Notice that since the computational domain is $\boldsymbol{\mu}-$dependent, also the matrix $\mathbf{X}_h(\boldsymbol{\mu})$ depends on the parameter $\boldsymbol{\mu}$. We highlight that one could alternatively ensure the well-posedness of (3.6) in terms of the matrix $\mathbf{B}(\boldsymbol{\mu})$, by requiring the existence of $\beta_p > 0$

$$\beta_{hp}(\boldsymbol{\mu}) = \inf_{\mathbf{q} \in \mathbb{R}^{N_h^p}} \sup_{\mathbf{v} \in \mathbb{R}^{N_h^u}} \frac{\mathbf{v}^T \mathbf{B}^T(\boldsymbol{\mu}) \mathbf{q}}{\|\mathbf{v}\|_{\mathbf{X}_u(\boldsymbol{\mu})} \|\mathbf{q}\|_{\mathbf{X}_p(\boldsymbol{\mu})}} \geq \beta_p \qquad \forall \boldsymbol{\mu} \in \mathcal{D}; \tag{3.13}$$

notice indeed that (3.13) together with the positive definiteness of $\mathbf{D}(\boldsymbol{\mu})$ is equivalent to (3.11), and ensures that the matrix $\mathbf{A}(\boldsymbol{\mu})$ is nonsingular, yielding a well-posed algebraic problem.

3.2 Review on RB methods for Stokes equations

The RB method represents a convenient framework for the reduction of parametrized saddle-point problems as the Stokes equations. In recent years, several works have been devoted to the analysis and the implementation of the RB method for addressing problems involving Stokes-like systems; a non-exhaustive list includes, among others: Stokes flows featuring affine parameter dependence [Rozza and Veroy, 2007, Gerner and Veroy, 2012, Rozza et al., 2013], nonaffine parameter dependence [Rozza, 2009]; Navier-Stokes flows depending on physical and/or geometrical parameters [Deparis, 2008, Quarteroni and Rozza, 2007, Deparis and Rozza, 2009, Manzoni, 2014]; parametrized optimal control problems [Negri et al., 2015b] or shape optimization problems [Manzoni et al., 2012b] involving Stokes flows. In all these cases, the RB method hinges upon:

1. a (weak) greedy algorithm for the incremental construction of the RB space, performed by selecting a new basis for velocity and pressure upon the use of a residual-based a posteriori error estimator. This latter is a $\boldsymbol{\mu}$-dependent quantity related with the FE approximation and is not always easily available or computable;

2. a Galerkin projection onto the RB space to generate the RB problem (G-RB method).

Of course, this is not the only available choice. Regarding point 1., POD, rather than greedy algorithms, can be used to build the RB spaces for velocity and pressure, either jointly or separately [Bache et al., 2010, Bergmann et al., 2009, Elman and Forstall, 2017, Kunisch and Volkwein, 2002b, Weller et al., 2010]. This option has been considered, e.g., in [Ballarin et al., 2015] where two-dimensional Navier-Stokes flows on simple geometries affinely parametrized have been treated. Moreover, we remark that other possibilities have been investigated, e.g. in [Díez et al., 2017], where proper generalized decomposition (PGD) is applied to the Stokes equations in two-dimensional parametrized geometries.

Concerning point 2., a more general Petrov-Galerkin (rather than Galerkin) projection - such as in the case of a least-squares (LS) method - can be performed, choosing a test space different from the trial space, see e.g. [Carlberg et al., 2011, Dahmen et al., 2012]. This option has been first explored in the case of two-dimensional, affinely parametrized Stokes problems on simple geometries in [Abdulle and Budáč, 2015]. Moreover, in both these cases parameter-dependent domains $\Omega(\boldsymbol{\mu})$ were obtained as images of a reference domain Ω^0 through a parameter-dependent map whose expression was known analytically. This is a relevant limitation toward the application of Petrov-Galerkin RB methods to more general domains with varying shape, not necessarily obtained in an explicit way from *a priori* known, parametrized deformations.

3.2.1 RB methods for parametrized saddle-point systems

As already discussed, the RB method is based on the idea that the solution of the parametrized system (3.6), for a certain value of the parameter $\boldsymbol{\mu}$, can be well approximated by a linear combination of basis functions obtained by orthonormalizing the solutions of the same problem for other values of the parameter. In the case of Stokes equations, the RB space V_N is defined as

$$V_N = span\{\vec{\xi}_i,\, i = 1, \dots, N\}$$

where

$$\vec{\xi}_i = (\vec{\varphi}_i^{\vec{u}}, 0) \qquad i = 1, \dots, N_u \qquad\qquad \vec{\xi}_i = (\vec{0}, \varphi_i^p) \qquad i = 1, \dots, N_p, \tag{3.14}$$

with $\{\vec{\varphi}_i^{\vec{u}}\}_{i=1}^{N_u}$ a basis for the velocity approximation and $\{\varphi_i^p\}_{i=1}^{N_p}$ for the pressure approximation. As a matter of fact, here $N = N_u + N_p$ and $V_N = V_{N_u} \times Q_{N_p}$, where

$$V_{N_u} = span\{\vec{\varphi}_i^{\vec{u}},\, i = 1, \dots, N_u\} \qquad Q_{N_p} = span\{\varphi_i^p,\, i = 1, \dots, N_p\}.$$

Then, a general PGRB approximation is constructed by introducing a set of (possibly $\boldsymbol{\mu}-$dependent) functions $\{w_i(\boldsymbol{\mu})\}_{i=1}^N$ such that a test space $W_N(\boldsymbol{\mu})$ is obtained as

$$W_N(\boldsymbol{\mu}) = span\{w_i(\boldsymbol{\mu}),\, i = 1, \dots, N\},$$

and considering the following problem: find $\vec{z}_N(\boldsymbol{\mu}) \in V_N$ such that

$$a(\vec{z}_N(\boldsymbol{\mu}), \vec{w}_N; \boldsymbol{\mu}) = f(\vec{w}_N; \boldsymbol{\mu}) \qquad \forall \vec{w}_N \in W_N(\boldsymbol{\mu}). \tag{3.15}$$

In order to obtain a well-posed RB approximation, an *inf-sup* condition at the RB level must also be satisfied. Specifically, there must exist $\beta_N^{min} > 0$, independent of $\boldsymbol{\mu}$, such that

$$\beta_N(\boldsymbol{\mu}) = \inf_{\vec{z}_N \in V_N} \sup_{\vec{w}_N \in W_N(\boldsymbol{\mu})} \frac{a(\vec{z}_N, \vec{w}_N; \boldsymbol{\mu})}{\|\vec{z}_N\|_{X(\boldsymbol{\mu})} \|\vec{w}_N\|_{X(\boldsymbol{\mu})}} \geq \beta_N^{min} > 0 \qquad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{3.16}$$

Being able to ensure this condition, as we will see in the following, essentially depends on the type of projection used to generate the RB problem and the way the RB spaces are built.

Algebraically, we recall that V_N is represented by the matrix $\mathbf{V} = [\boldsymbol{\xi}_1 | \dots | \boldsymbol{\xi}_N] \in \mathbb{R}^{N_h \times N}$, where $\boldsymbol{\xi}_i, i = 1, \dots, N$ are the FE vector representation of the basis $\vec{\xi}_i, i = 1, \dots, N$. On the other hand, $W_N(\boldsymbol{\mu})$ is represented by a matrix $\mathbf{W}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N}$, which is generally different from $\mathbf{V}$ and may be $\boldsymbol{\mu}$-dependent. If $\mathbf{W}(\boldsymbol{\mu}) \neq \mathbf{V}$ we have the more general PGRB approximation, otherwise, if $\mathbf{W}(\boldsymbol{\mu}) = \mathbf{V}$, we come up with the more usual Galerkin approximation. Problem (3.15) leads to the following algebraic RB linear system

$$\mathbf{A}_N(\boldsymbol{\mu})\mathbf{z}_N(\boldsymbol{\mu}) = \mathbf{g}_N(\boldsymbol{\mu}), \tag{3.17}$$

where the RB matrix $\mathbf{A}_N(\boldsymbol{\mu}) \in \mathbb{R}^{N\times N}$ and the RB right hand side $\mathbf{g}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ are defined as

$$\mathbf{A}_N(\boldsymbol{\mu}) = (\mathbf{W}(\boldsymbol{\mu}))^T\mathbf{A}(\boldsymbol{\mu})\mathbf{V}, \qquad \mathbf{g}_N(\boldsymbol{\mu}) = (\mathbf{W}(\boldsymbol{\mu}))^T\mathbf{g}(\boldsymbol{\mu}). \tag{3.18}$$

The nonsingularity of $\mathbf{A}_N(\boldsymbol{\mu})$ is ensured by (3.16), which at algebraic level is equivalent to

$$\beta_N(\boldsymbol{\mu}) = \inf_{\mathbf{z}_N\in\mathbb{R}^N}\sup_{\mathbf{w}_N\in\mathbb{R}^N}\frac{\mathbf{w}_N^T\mathbf{A}_N(\boldsymbol{\mu})\mathbf{z}_N}{\|\mathbf{V}\mathbf{z}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})}\|\mathbf{W}(\boldsymbol{\mu})\mathbf{w}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})}} \geq \beta_N^{min} \qquad \forall\boldsymbol{\mu}\in\mathcal{D}. \tag{3.19}$$

In the framework we developed, the matrix $\mathbf{V}$ is built by employing the POD method, which in the Stokes case turns to

$$\mathbf{V} = \begin{bmatrix}\mathbf{V}_{N_u} & 0\\ 0 & \mathbf{V}_{N_p}\end{bmatrix} = \Big[\boldsymbol{\xi}_1|\dots|\boldsymbol{\xi}_{N_u}|\boldsymbol{\xi}_{N_u+1}|\dots|\boldsymbol{\xi}_N\Big]\,; \tag{3.20}$$

here $\mathbf{V}_{N_u} \in \mathbb{R}^{N_h^u\times N_u}$ and $\mathbf{V}_{N_p} \in \mathbb{R}^{N_h^p\times N_p}$ are used to approximate the velocity $\mathbf{u}(\boldsymbol{\mu})$ and the pressure $\mathbf{p}(\boldsymbol{\mu})$, respectively. In particular,

$$\boldsymbol{\xi}_i = \begin{bmatrix}\boldsymbol{\varphi}_i^{\vec{u}}\\ \mathbf{0}\end{bmatrix} \quad i = 1,\dots,N_u, \qquad\qquad \boldsymbol{\xi}_{N_u+i} = \begin{bmatrix}\mathbf{0}\\ \boldsymbol{\varphi}_i^p\end{bmatrix} \quad i = 1,\dots,N_p,$$

where $\{\boldsymbol{\varphi}_i^{\vec{u}}\}_i^{N_u}$ and $\{\boldsymbol{\varphi}_i^p\}_i^{N_p}$ are the FE vector basis functions for the velocity and the pressure RB space, that is,

$$\mathbf{V}_{N_u} = \Big[\boldsymbol{\varphi}_1^{\vec{u}}|\dots|\boldsymbol{\varphi}_{N_u}^{\vec{u}}\Big], \qquad \mathbf{V}_{N_p} = \Big[\boldsymbol{\varphi}_1^p|\dots|\boldsymbol{\varphi}_{N_p}^p\Big],$$

and such that $\boldsymbol{\varphi}_i^{\vec{u}}$, $i = 1,\dots,N_u$, (resp. $\boldsymbol{\varphi}_i^p$, $i = 1,\dots,N_p$) are the FE vector representation of $\vec{\varphi}_i^{\vec{u}}$, $i = 1,\dots,N_u$, (resp. φ_i^p, $i = 1,\dots,N_p$).

The construction of the RB spaces is thus performed by first collecting a set of FE snapshots $\{\mathbf{u}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$, $\{\mathbf{p}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$, solutions of (3.6) for different instances of the parameters $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$, and then performing separately POD, yielding

$$\mathbf{V}_{N_u} = POD\Big(\mathbf{S}_{\vec{u}}, \mathbf{X}_u, \varepsilon_{\text{POD}}\Big), \qquad \mathbf{V}_{N_p} = POD\Big(\mathbf{S}_p, \mathbf{X}_p, \varepsilon_{\text{POD}}\Big),$$

where $\mathbf{S}_{\vec{u}} = [\mathbf{u}(\boldsymbol{\mu}_1),\dots,\mathbf{u}(\boldsymbol{\mu}_{n_s})] \in \mathbb{R}^{N_h^u\times n_s}$ and $\mathbf{S}_p = [\mathbf{p}(\boldsymbol{\mu}_1),\dots,\mathbf{p}(\boldsymbol{\mu}_{n_s})] \in \mathbb{R}^{N_h^p\times n_s}$ collect the velocity and pressure snapshots, respectively. The matrices $\mathbf{V}_{N_u}$ and $\mathbf{V}_{N_p}$ are constructed by selecting the largest N_u and N_p eigenmodes respectively, as explained in

Algorithm 1 in Section 1.3.1. We highlight that the dimension $N = N_u + N_p \ll N_h$ of the RB system is smaller than the dimension N_h of the FE linear system of several orders of magnitude, so that problem (3.17) is solved by direct methods and in genera $N_u \neq N_p$.

What makes the RB approximation of parametrized Stokes equations (and, more generally, of parametrized saddle-point problems) much harder than coercive elliptic problems, is ensuring the stability of the resulting RB problem. This is the main reason why, for instance, reduced-order models for fluid dynamics problems have sometimes focused on approximations for the velocity field uniquely, recovering then the pressure in a different way, rather than building a reduced-order approximation based on a mixed velocity-pressure formulation. Indeed, it is well-known that in the RB approximation, a stable couple of reduced subspaces for velocity and pressure, satisfying an equivalent *inf-sup* condition at the reduced level, ensures that the RB Stokes problem is well-posed. This property is not automatically fulfilled if the RB problem is constructed through a Galerkin projection employing RB spaces made solely of orthonormalized solutions of (3.6) obtained for different values of parameters. To overcome this shortcoming, two strategies have been proposed.

A. The velocity space is augmented by means of a set of *enriching* basis functions computed through the pressure supremizing operator $T_p(\cdot;\boldsymbol{\mu}) : Q_h(\boldsymbol{\mu}) \to V_h(\boldsymbol{\mu})$ such that, for any given $q_h \in Q_h(\boldsymbol{\mu})$, $T_p(q_h;\boldsymbol{\mu})$ is the solution of the following variational problem

$$(T_p(q_h;\boldsymbol{\mu}), \vec{v}_h)_{V(\boldsymbol{\mu})} = b(\vec{v}_h, q_h; \boldsymbol{\mu}) \qquad \forall \vec{v}_h \in V_h(\boldsymbol{\mu}). \tag{3.21}$$

This yields a RB problem with additional degrees of freedom for the velocity field (as many as the pressure variable), see [Rozza et al., 2013] for the details. In presence of parameter-dependent domains, the supremizing operator is $\boldsymbol{\mu}$-dependent, so that to recover computational efficiency (and avoid the construction of the pressure supremizing operator for any value of $\boldsymbol{\mu}$ online), an offline enrichment is employed. This strategy leads to a RB problem which is *inf-sup* stable in practice, however its well-posedness is not proven rigorously. Such a framework has been originally introduced in conjunction with a (weak) greedy algorithm [Rozza and Veroy, 2007, Gerner and Veroy, 2012, Rozza et al., 2013], and later exploited when dealing with POD. In the former case, for each pressure basis selected by the greedy algorithm, a supremizing function (hereon also referred to as supremizer) is used to augment the velocity space. In the latter case, however, the basis functions are not directly related to precise instances of the parameter, so that a set of enriching functions for the velocity space must be computed in advance starting from the pressure snapshots, so that (3.21) must be solved for any parameter considered offline; then POD is applied to build the enriching basis [Ballarin et al., 2015]. This technique allows to build a stable RB problem, however it is not clear, a priori, how many supremizing functions are needed to properly stabilize the problem. Taking as many enriching functions as the number of velocity and pressure basis is a working rule of thumb, however very likely this leads to an excessive number of basis functions.

B. Alternatively to strategy A, we can exploit a least squares (LS) method [Abdulle and Budáč, 2015, Quarteroni et al., 2016a] to build an automatically stable RB problem. The resulting LSRB method relies on a test space which is obtained as the image of the RB space through a global supremizing operator $T(\cdot;\boldsymbol{\mu}) : X_h(\boldsymbol{\mu}) \to X_h(\boldsymbol{\mu})$, such that

$$(T(\vec{z}_h;\boldsymbol{\mu}),\vec{w}_h)_{X(\boldsymbol{\mu})} = a(\vec{z}_h,\vec{w}_h;\boldsymbol{\mu}) \qquad \forall \vec{w}_h \in X_h(\boldsymbol{\mu}). \tag{3.22}$$

With respect to the definition of $T_p(\cdot;\boldsymbol{\mu})$ provided by (3.21), both velocity and pressure appear, together with the full Stokes operator at the right hand side. The corresponding algebraic construction of this operator substantially relies on the choice of the matrix-norm to be used for the velocity and pressure spaces. By this approach the resulting RB problem is automatically stable - that is, it satisfies the required *inf-sup* condition - as usually happens when dealing with PGRB methods for weakly coercive problems, see [Quarteroni et al., 2016a] for further details. However, the existing formulation of the LSRB method for Stokes equations proposed in [Abdulle and Budáč, 2015] presumes the existence of an explicit $\boldsymbol{\mu}-$dependent function which enables to recast the problem on a reference domain. Without this function available, as it is the case when the deformation results from the solution of an additional FE problem, the computational work to build the RB problem is unbearable, thus preventing the use of the LSRB method.

In the following sections we report both the G-RB and LSRB approximations for the parametrized Stokes equations. In Section 3.3 a novel algebraic LSRB (aLSRB) method, which overcomes the structural limitations of the existing LSRB approximation, is constructed. In Section 3.4 we report numerical results which show how the proposed aLSRB approach yields to a more accurate and efficient approximation than the one obtained by relying on state-of-the-art RB formulations.

Galerkin-RB method with velocity enrichment

A Galerkin-RB formulation is obtained by choosing $W_N(\boldsymbol{\mu}) = V_N$ (or algebraically $\mathbf{W}(\boldsymbol{\mu}) = \mathbf{V}$) in (3.18), resulting in a RB approximation whose well-posedness is guaranteed by satisfying the following assumption: there must exist $\tilde{\beta}_N^{min} > 0$ such that

$$\tilde{\beta}_N(\boldsymbol{\mu}) = \inf_{q_N \in Q_{N_p}} \sup_{\vec{v}_N \in V_{N_u}} \frac{b(\vec{v}_N, q_N;\boldsymbol{\mu})}{\|\vec{v}_N\|_{V(\boldsymbol{\mu})}\|q_N\|_{Q(\boldsymbol{\mu})}} \geq \tilde{\beta}_N^{min} > 0 \qquad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{3.23}$$

Unfortunately, as explained above, condition (3.23) is not automatically satisfied when the RB spaces V_{N_u} and Q_{N_p} are constructed by POD, or by greedy algorithms, by considering basis functions extracted from velocity and pressure snapshots only. Consequently, we consider an "enriched" velocity space formulation, as proposed in [Ballarin et al., 2015], where the velocity space V_{N_u} is augmented to guarantee the well-posedness of the resulting RB approximation. Algebraically, this is pursued by building a matrix $\mathbf{V}_{N_s} \in \mathbb{R}^{N_h^u \times N_s}$ whose columns form a basis for the enriching RB velocity space, which is properly orthonormalized with a Gram-Schmidt (G-S) procedure:

$$\mathbf{V}_{N_u} = \text{G-S}([\mathbf{V}_{N_u}, \mathbf{V}_{N_s}], \mathbf{X}_u). \tag{3.24}$$

Then, the G-RB approximation is built by considering $\mathbf{V} = \mathbf{W}(\boldsymbol{\mu})$ in (3.18) with $\mathbf{V}_{N_u}$ as in (3.24).

The enriching strategy is based upon the use of the pressure-supremizing operator defined by problem (3.21), which corresponds to a FE problem with the following algebraic structure

$$\mathbf{X}_u(\boldsymbol{\mu})\mathbf{t}_p(\mathbf{q}; \boldsymbol{\mu}) = \mathbf{B}^T(\boldsymbol{\mu})\mathbf{q}, \tag{3.25}$$

where $\mathbf{q} \in \mathbb{R}^{N_h^p}$ is the FE vector representation of $q_h \in Q_h(\boldsymbol{\mu})$. A well-posed G-RB approximation for a new parameter $\boldsymbol{\mu}$ is obtained in two ways:

- build for each pressure basis $\{\boldsymbol{\varphi}_i^p\}_{i=1}^{N_p}$ the corresponding supremizing functions $\{\mathbf{t}_p(\boldsymbol{\varphi}_i^p; \boldsymbol{\mu})\}_{i=1}^{N_p}$ and define

  $$\mathbf{V}_{N_s} = [\mathbf{t}_p(\boldsymbol{\varphi}_1^p; \boldsymbol{\mu})| \dots |\mathbf{t}_p(\boldsymbol{\varphi}_{N_p}^p; \boldsymbol{\mu})],$$

  leading to a RB formulation which by definition satisfies (3.23). However, in this way the construction of the supremizing enriching functions is not computationally feasible, because it entails (online) the solution of N_p FE linear system for each new value of $\boldsymbol{\mu}$;

- compute a set of supremizing snapshots $\{\mathbf{t}_p(\mathbf{p}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ corresponding to the pressure snapshots $\{\mathbf{p}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ through (3.25), and then build the matrix $\mathbf{V}_{N_s}$ through POD

  $$\mathbf{V}_{N_s} = POD\Big(\mathbf{S}_{\vec{t}}, \mathbf{X}_u, \varepsilon_{\text{POD}}\Big),$$

  with $\mathbf{S}_{\vec{t}} = [\mathbf{t}_p(\boldsymbol{\varphi}_1^p; \boldsymbol{\mu}_1), \dots, \mathbf{t}_p(\boldsymbol{\varphi}_{n_s}^p; \boldsymbol{\mu}_{n_s})]$. Notice that this option does not ensure that condition (3.23) (or any equivalent one) is satisfied. Moreover, the number N_s of basis functions for $\mathbf{V}_{N_s}$ is usually chosen equal to N_u, doubling the size of the RB velocity space. This looks like a reliable option which yields a stable RB problem for the steady Navier-Stokes equations, see [Ballarin et al., 2015].

LSRB method

Instead of performing a Galerkin projection onto properly enriched RB spaces, the Petrov-Galerkin (PG)-RB method uses a different test space $\mathbf{W}(\boldsymbol{\mu})$ and naturally builds an *inf-sup* stable RB problem. The PGRB method has been firstly analyzed for the affinely parametrized Stokes equations in [Abdulle and Budáč, 2015] where the RB space is built upon a greedy algorithm. In this work we deepen the analysis carried out in [Abdulle and Budáč, 2015], propose several strategies to make this method computationally efficient and use instead the POD method for the construction of the RB space, which does not need any error estimator. Moreover, we do not assume to have an analytical function which maps the reference domain Ω^0 to the physical domain $\Omega(\boldsymbol{\mu})$; the main consequence is that we consider the more general case where recasting the problem on a reference, parameter-independent domain Ω^0 is not possible. We restrict ourselves to the case of PGRB method built through the least-squares (LS) method, which automatically guarantees to obtain an *inf-sup* stable problem. With this aim, we exploit the global supremizing operator defined in (3.22), which entails a $\boldsymbol{\mu}-$dependent FE problem to be solved for $T(\vec{z}_h;\boldsymbol{\mu})$. Then, the RB problem reads as (3.15), where the test space is chosen as

$$W_N(\boldsymbol{\mu}) = span\{T(\vec{\xi}_i;\boldsymbol{\mu}),\ i=1,\dots,N\},$$

while the trial RB space is chosen as in (1.38). From an algebraic standpoint, given $\mathbf{z}\in\mathbb{R}^{N_h}$, the supremizing solution $\mathbf{t}(\mathbf{z};\boldsymbol{\mu})$ is obtained by solving the linear system

$$\mathbf{X}_h(\boldsymbol{\mu})\mathbf{t}(\mathbf{z};\boldsymbol{\mu}) = \mathbf{A}(\boldsymbol{\mu})\mathbf{z}. \tag{3.26}$$

The projection matrix $\mathbf{W}(\boldsymbol{\mu})$, whose columns are supremizers of type (3.26) and form a basis for the (parameter-dependent) test space, is then given by

$$\mathbf{W}(\boldsymbol{\mu}) = \mathbf{X}_h^{-1}(\boldsymbol{\mu})\mathbf{A}(\boldsymbol{\mu})\mathbf{V}, \tag{3.27}$$

where $\mathbf{X}_h(\boldsymbol{\mu})$ is the $\boldsymbol{\mu}-$dependent norm matrix (3.12). Finally, the linear system (3.17) representing the LSRB problem is recovered by setting

$$\mathbf{A}_N(\boldsymbol{\mu}) = \mathbf{V}^T\mathbf{A}^T(\boldsymbol{\mu})\mathbf{X}_h^{-1}(\boldsymbol{\mu})\mathbf{A}(\boldsymbol{\mu})\mathbf{V}, \qquad \mathbf{g}_N(\boldsymbol{\mu}) = \mathbf{V}^T\mathbf{A}^T(\boldsymbol{\mu})\mathbf{X}_h^{-1}(\boldsymbol{\mu})\mathbf{g}(\boldsymbol{\mu}). \tag{3.28}$$

The RB matrix as defined in (3.28) is nonsingular, leading to a well-posed RB approximation, as the following results confirms, see also [Abdulle and Budáč, 2015, Quarteroni et al., 2016a].

Lemma 3.2.1. *Assume that condition* (3.11) *holds and that* $\mathbf{W}(\boldsymbol{\mu})$ *is taken as in* (3.27). *Then, the LSRB problem* (3.17) *is inf-sup stable uniformly with respect to* $\boldsymbol{\mu}$, *that is, there exists* $\beta_N^{min} > 0$ *independent of* $\boldsymbol{\mu}$ *such that* (3.19) *holds with* $\beta_N^{min} = \beta_h^{min}$. *Moreover, it* *has a unique solution* $\mathbf{z}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ *for any* $\boldsymbol{\mu} \in \mathcal{D}$*, which satisfies*

$$\|\mathbf{z}_N(\boldsymbol{\mu})\|_{\mathbf{X}_h(\boldsymbol{\mu})} \leq \frac{1}{\beta_N(\boldsymbol{\mu})} \|\mathbf{g}(\boldsymbol{\mu})\|_{\mathbf{X}_h^{-1}(\boldsymbol{\mu})}.$$

Proof. We report an algebraic variant of the proof of the result shown in [Abdulle and Budáč, 2015]. Starting from (3.26) and the Cauchy-Schwarz inequality

$$\mathbf{w}^T\mathbf{A}(\boldsymbol{\mu})\mathbf{z} = \mathbf{w}^T\mathbf{X}_h(\boldsymbol{\mu})\mathbf{t}(\mathbf{z};\boldsymbol{\mu}) \leq \|\mathbf{t}(\mathbf{z};\boldsymbol{\mu})\|_{\mathbf{X}_h(\boldsymbol{\mu})}\|\mathbf{w}\|_{\mathbf{X}_h(\boldsymbol{\mu})} \qquad \forall \mathbf{w} \in \mathbb{R}^{N_h},$$

and the equality is reached for $\mathbf{w} = \mathbf{t}(\mathbf{z};\boldsymbol{\mu})$. Then, we have

$$\begin{aligned}\beta_N(\boldsymbol{\mu}) &= \inf_{\vec{z}_N \in \mathbb{R}^N} \sup_{\vec{w}_N \in \mathbb{R}^N} \frac{\mathbf{w}_N^T \mathbf{A}_N(\boldsymbol{\mu})\mathbf{z}_N}{\|\mathbf{V}\mathbf{z}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})}\|\mathbf{W}(\boldsymbol{\mu})\mathbf{w}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})}} = \inf_{\mathbf{z}_N \in \mathbb{R}^N} \frac{\|\mathbf{t}^{\mu}(\mathbf{V}\mathbf{z}_N)\|_{\mathbf{X}_h(\boldsymbol{\mu})}}{\|\mathbf{V}\mathbf{z}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})}} \\ &\geq \inf_{\mathbf{z} \in \mathbb{R}^{N_h}} \frac{\|\mathbf{t}(\mathbf{z};\boldsymbol{\mu})\|_{\mathbf{X}_h(\boldsymbol{\mu})}}{\|\mathbf{z}\|_{\mathbf{X}_h(\boldsymbol{\mu})}} \geq \beta_h^{min}.\end{aligned}$$

The proof is concluded by employing an algebraic variant of the Babuška theorem for non-coercive problems satisfying an *inf-sup* stability property, see [Babuška, 1971]. □

Remark 3.2.1. *The solution* $\mathbf{z}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ *of problem* (3.17) *solves the following minimization problem*

$$\mathbf{z}_N(\boldsymbol{\mu}) = \arg \min_{\mathbf{v}_N \in \mathbb{R}^N} \|\mathbf{g}(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu})\mathbf{V}\mathbf{v}_N\|^2_{\mathbf{X}_h^{-1}(\boldsymbol{\mu})}, \tag{3.29}$$

i.e. the RB solution minimizes the residual in the norm induced by the symmetric positive definite matrix $\mathbf{X}_h^{-1}(\boldsymbol{\mu})$*, see [Quarteroni et al., 2016a] for further details.*

3.3 Algebraic LSRB method for parametrized saddle-point problems

The LSRB method described in Section 3.2.1 requires to build the $\boldsymbol{\mu}$-dependent matrix $\mathbf{X}_h^{-1}(\boldsymbol{\mu})$ or to solve approximately the N linear systems (3.26) associated with the matrix $\mathbf{X}_h(\boldsymbol{\mu})$ to construct a stable RB problem for any new parameter instances $\boldsymbol{\mu} \in \mathcal{D}$ considered online. If an analytical map is available, one can recast problem (3.3) over the reference domain Ω^0 by using the Jacobian of the map. In this way, the LSRB problem would be built with respect to the reference domain, and the independence of the norm matrix $\mathbf{X}_h(\boldsymbol{\mu})$ on $\boldsymbol{\mu}$ would be easily achieved. However, if the displacement of the domain is not analitically available, it is not possible to rely on this strategy, due to the huge assembling costs of $\mathbf{A}_N(\boldsymbol{\mu})$.

In this section we propose a purely algebraic PGRB method which can be viewed as an algebraic LSRB (aLSRB) method described above for parametrized saddle-point problems as (3.6), see [Dal Santo et al., 2017b]. Compared to the approximate enrichment of the velocity space described in Section 3.2.1, the aLSRB method allows to build a RB problem which is automatically and rigorously *inf-sup* stable and henceforth it does not require to enrich the velocity space doubling the degrees of freedom of the velocity.

The underlying idea is to substitute the matrix $\mathbf{X}_h(\boldsymbol{\mu})$ appearing in the definition of the test space (3.27) by a properly chosen surrogate $\mathbf{P}_X \in \mathbb{R}^{N_h \times N_h}$. To this aim, we suppose the following assumption to hold.

Assumption 3.3.1. *The matrix* $\mathbf{P}_X \in \mathbb{R}^{N_h \times N_h}$ *is symmetric and positive definite and induces a scalar product* $(\mathbf{x}, \mathbf{y})_{\mathbf{P}_X} = \mathbf{x}^T \mathbf{P}_X \mathbf{y}$ *and a norm* $\|\mathbf{x}\|^2_{\mathbf{P}_X} = (\mathbf{x}, \mathbf{x})_{\mathbf{P}_X} = \mathbf{x}^T \mathbf{P}_X \mathbf{x}$ *for any* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{N_h}$*. Moreover, there exist two positive constants* c *and* C *such that*

$$c\|\mathbf{x}\|_{\mathbf{P}_X} \leq \|\mathbf{x}\|_{\mathbf{X}_h(\boldsymbol{\mu})} \leq C\|\mathbf{x}\|_{\mathbf{P}_X} \qquad \forall \mathbf{x} \in \mathbb{R}^{N_h}. \tag{3.30}$$

Next, we introduce a slightly modified supremizing operator $T_{\mathbf{P}_X}(\cdot;\boldsymbol{\mu}) : V_h(\boldsymbol{\mu}) \times V_h(\boldsymbol{\mu}) \to V_h(\boldsymbol{\mu})$ defined by the following problem

$$(T_{\mathbf{P}_X}(\vec{z}_h;\boldsymbol{\mu}), \vec{w}_h)_{\mathbf{P}_X} = a(\vec{z}_h, \vec{w}_h;\boldsymbol{\mu}) \qquad \forall \vec{w}_h \in V_h(\boldsymbol{\mu}), \tag{3.31}$$

where the difference with respect to (3.22) is the choice of the scalar product with respect to which the operator is built. Reasoning as in the previous section, we introduce a PG problem under the form: find $\vec{z}_N \in V_N$ such that

$$a(\vec{z}_N, \vec{w}_N;\boldsymbol{\mu}) = f(\vec{w}_N;\boldsymbol{\mu}) \qquad \forall \vec{w}_N \in W_{N,\mathbf{P}_X}(\boldsymbol{\mu}), \tag{3.32}$$

where now the test space is chosen as

$$W_{N,\mathbf{P}_X}(\boldsymbol{\mu}) = span\{T_{\mathbf{P}_X}(\vec{\xi}_i;\boldsymbol{\mu}),\ i = 1,\dots,N\},$$

where $\{\vec{\xi}_i\}_{i=1}^N$ are the RB functions defining V_N in (1.38). Problem (3.31) is algebraically equivalent to solving

$$\mathbf{P}_X \mathbf{t}_{\mathbf{P}_X}(\mathbf{z};\boldsymbol{\mu}) = \mathbf{A}(\boldsymbol{\mu})\mathbf{z}, \tag{3.33}$$

and yields a projection matrix of the following form

$$\mathbf{W}_{\mathbf{P}_X}(\boldsymbol{\mu}) = \mathbf{P}_X^{-1}\mathbf{A}(\boldsymbol{\mu})\mathbf{V}. \tag{3.34}$$

Finally, the corresponding RB system is

$$\widehat{\mathbf{A}}_N(\boldsymbol{\mu})\mathbf{z}_N(\boldsymbol{\mu}) = \hat{\mathbf{g}}_N(\boldsymbol{\mu}), \tag{3.35}$$

where the RB matrix $\widehat{\mathbf{A}}_N(\boldsymbol{\mu}) \in \mathbb{R}^{N \times N}$ and the RB right hand side $\hat{\mathbf{g}}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ are defined as

$$\widehat{\mathbf{A}}_N(\boldsymbol{\mu}) = \mathbf{V}^T\mathbf{A}^T(\boldsymbol{\mu})\mathbf{P}_X^{-1}\mathbf{A}(\boldsymbol{\mu})\mathbf{V}, \qquad \hat{\mathbf{g}}_N(\boldsymbol{\mu}) = \mathbf{V}^T\mathbf{A}^T(\boldsymbol{\mu})\mathbf{P}_X^{-1}\mathbf{g}(\boldsymbol{\mu}). \tag{3.36}$$

Remark 3.3.1. *Equations* (3.36) *are similar to the ones in* (3.28), *provided that* $\mathbf{X}_h(\boldsymbol{\mu})$ *is substituted with* $\mathbf{P}_X$.

Similarly to (3.7), the following block structure can be devised for the aLSRB problem:

$$\widehat{\mathbf{A}}_N(\boldsymbol{\mu}) = \begin{bmatrix} \widehat{\mathbf{D}}_N(\boldsymbol{\mu}) & \widehat{\mathbf{B}}_N^T(\boldsymbol{\mu}) \\ \widehat{\mathbf{B}}_N(\boldsymbol{\mu}) & \widehat{\mathbf{C}}_N(\boldsymbol{\mu}) \end{bmatrix}, \tag{3.37}$$

and

$$\mathbf{z}_N(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{u}_N(\boldsymbol{\mu}) \\ \mathbf{p}_N(\boldsymbol{\mu}) \end{bmatrix}, \qquad \hat{\mathbf{g}}_N(\boldsymbol{\mu}) = \begin{bmatrix} \hat{\mathbf{f}}_N(\boldsymbol{\mu}) \\ \hat{\mathbf{r}}_N(\boldsymbol{\mu}) \end{bmatrix}, \tag{3.38}$$

where the block (2,2) is filled as in the standard LSRB method. The RB block matrices are defined as

$$\begin{aligned} \widehat{\mathbf{D}}_N(\boldsymbol{\mu}) &= (\mathbf{V}_{N_u})^T\mathbf{D}^T(\boldsymbol{\mu})\mathbf{P}_{\mathbf{X}_u}^{-1}\mathbf{D}(\boldsymbol{\mu})\mathbf{V}_{N_u} + (\mathbf{V}_{N_u})^T\mathbf{B}^T(\boldsymbol{\mu})\mathbf{P}_{\mathbf{X}_p}^{-1}\mathbf{B}(\boldsymbol{\mu})\mathbf{V}_{N_u}, \\ \widehat{\mathbf{B}}_N(\boldsymbol{\mu}) &= (\mathbf{V}_{N_p})^T\mathbf{B}(\boldsymbol{\mu})\mathbf{P}_{\mathbf{X}_u}^{-1}\mathbf{D}(\boldsymbol{\mu})\mathbf{V}_{N_u}, \\ \widehat{\mathbf{C}}_N(\boldsymbol{\mu}) &= (\mathbf{V}_{N_p})^T\mathbf{B}(\boldsymbol{\mu})\mathbf{P}_{\mathbf{X}_u}^{-1}\mathbf{B}^T(\boldsymbol{\mu})\mathbf{V}_{N_p}, \end{aligned}$$

and the RB block vectors are

$$\begin{aligned} \hat{\mathbf{f}}_N(\boldsymbol{\mu}) &= (\mathbf{V}_{N_u})^T\mathbf{D}^T(\boldsymbol{\mu})\mathbf{P}_{\mathbf{X}_u}^{-1}\mathbf{f}(\boldsymbol{\mu}) + (\mathbf{V}_{N_u})^T\mathbf{B}^T(\boldsymbol{\mu})\mathbf{P}_{\mathbf{X}_p}^{-1}\mathbf{r}(\boldsymbol{\mu}), \\ \hat{\mathbf{r}}_N(\boldsymbol{\mu}) &= (\mathbf{V}_{N_p})^T\mathbf{B}(\boldsymbol{\mu})\mathbf{P}_{\mathbf{X}_u}^{-1}\mathbf{f}(\boldsymbol{\mu}). \end{aligned}$$

3.3.1 Well-posedness of the aLSRB problem

In the following, we provide results showing the stability of system (3.35) and the optimality properties satisfied by the solution $\mathbf{z}_N(\boldsymbol{\mu})$ of (3.35).

Proposition 3.3.1. *Assume that condition* (3.11) *holds, that* $\mathbf{W}(\boldsymbol{\mu})$ *is taken as in* (3.34) *and let assumption 3.3.1 hold. Then problem* (3.35) *is inf-sup stable, more precisely, for any* $\boldsymbol{\mu} \in \mathcal{D}$

$$\beta_{\mathbf{P}_X,N}(\boldsymbol{\mu}) = \inf_{\mathbf{z}_N \in \mathbb{R}^N} \sup_{\mathbf{w}_N \in \mathbb{R}^N} \frac{\mathbf{w}_N^T \widehat{\mathbf{A}}_N(\boldsymbol{\mu})\mathbf{z}_N}{\|\mathbf{V}\mathbf{z}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})}\|\mathbf{W}_{\mathbf{P}_X}(\boldsymbol{\mu})\mathbf{w}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})}} \geq \frac{c}{C}\beta_h^{min}. \tag{3.39}$$

Moreover, problem (3.35) *has a unique solution* $\mathbf{z}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ *for any* $\boldsymbol{\mu} \in \mathcal{D}$, *which* *satisfies*

$$\|\mathbf{z}_N(\boldsymbol{\mu})\|_{\mathbf{X}_h(\boldsymbol{\mu})} \leq \frac{1}{\beta_{\mathbf{P}_X,N}(\boldsymbol{\mu})}\|\mathbf{g}(\boldsymbol{\mu})\|_{\mathbf{X}_h^{-1}(\boldsymbol{\mu})}.$$

Proof. Starting from (3.33), it holds

$$\mathbf{w}^T\mathbf{A}(\boldsymbol{\mu})\mathbf{z} = \mathbf{w}^T\mathbf{P}_X\mathbf{t}_{\mathbf{P}_X}(\mathbf{z};\boldsymbol{\mu}) \leq \|\mathbf{t}_{\mathbf{P}_X}(\mathbf{z};\boldsymbol{\mu})\|_{\mathbf{P}_X}\|\mathbf{w}\|_{\mathbf{P}_X} \qquad \forall \mathbf{w}\in\mathbb{R}^{N_h},$$

where the equality is reached for $\mathbf{w} = \mathbf{t}_{\mathbf{P}_X}(\mathbf{z};\boldsymbol{\mu})$. Consequently, using the inequalities in (3.30) we have

$$\begin{aligned}
\beta_{\mathbf{P}_X,N}(\boldsymbol{\mu}) &= \inf_{\mathbf{z}_N\in\mathbb{R}^N}\sup_{\mathbf{w}_N\in\mathbb{R}^N}\frac{\mathbf{w}_N^T\widehat{\mathbf{A}}_N(\boldsymbol{\mu})\mathbf{z}_N}{\|\mathbf{V}\mathbf{z}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})}\|\mathbf{W}_{\mathbf{P}_X}(\boldsymbol{\mu})\mathbf{w}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})}}\\
&\geq \frac{1}{C}\inf_{\mathbf{z}_N\in\mathbb{R}^N}\sup_{\mathbf{w}_N\in\mathbb{R}^N}\frac{\mathbf{w}_N^T\widehat{\mathbf{A}}_N(\boldsymbol{\mu})\mathbf{z}_N}{\|\mathbf{V}\mathbf{z}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})}\|\mathbf{W}_{\mathbf{P}_X}(\boldsymbol{\mu})\mathbf{w}_N\|_{\mathbf{P}_X}}\\
&= \frac{1}{C}\inf_{\mathbf{z}_N\in\mathbb{R}^N}\frac{\|\mathbf{t}_{\mathbf{P}_X}(\mathbf{V}\mathbf{z}_N;\boldsymbol{\mu})\|_{\mathbf{P}_X}}{\|\mathbf{V}\mathbf{z}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})}}\\
&\geq \frac{1}{C}\inf_{\mathbf{z}\in\mathbb{R}^{N_h}}\frac{\|\mathbf{t}_{\mathbf{P}_X}(\mathbf{z};\boldsymbol{\mu})\|_{\mathbf{P}_X}}{\|\mathbf{z}\|_{\mathbf{X}_h(\boldsymbol{\mu})}}\\
&= \frac{1}{C}\inf_{\mathbf{z}\in\mathbb{R}^{N_h}}\sup_{\mathbf{w}\in\mathbb{R}^{N_h}}\frac{\mathbf{w}^T\mathbf{A}(\boldsymbol{\mu})\mathbf{z}}{\|\mathbf{z}\|_{\mathbf{X}_h(\boldsymbol{\mu})}\|\mathbf{w}\|_{\mathbf{P}_X}}\\
&\geq \frac{c}{C}\inf_{\mathbf{z}\in\mathbb{R}^{N_h}}\sup_{\mathbf{w}\in\mathbb{R}^{N_h}}\frac{\mathbf{w}^T\mathbf{A}(\boldsymbol{\mu})\mathbf{z}}{\|\mathbf{z}\|_{\mathbf{X}_h(\boldsymbol{\mu})}\|\mathbf{w}\|_{\mathbf{X}_h(\boldsymbol{\mu})}}\\
&= \frac{c}{C}\beta_h^{min}(\boldsymbol{\mu}) \geq \frac{c}{C}\beta_h^{min}.
\end{aligned}$$

By applying an algebraic equivalent of the Babuška theorem for non-coercive problems satisfying an *inf-sup* stability property, see [Babuška, 1971], concludes the proof. □

Proposition 3.3.2. *Let assumption 3.3.1 hold, then problem* (3.35) *corresponds to solving the minimization problem*

$$\mathbf{z}_N(\boldsymbol{\mu}) = \arg\min_{\mathbf{v}_N\in\mathbb{R}^N}\|\mathbf{g}(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu})\mathbf{V}\mathbf{v}_N\|^2_{\mathbf{P}_X^{-1}}. \tag{3.40}$$

Proof. We consider the quadratic functional

$$J(\mathbf{v}_N) = \|\mathbf{g}(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu})\mathbf{V}\mathbf{v}_N\|^2_{\mathbf{P}_X^{-1}}, \quad \mathbf{v}_N\in\mathbb{R}^N,$$

which has a unique minimum in $\mathbf{u}_N\in\mathbb{R}^N$ thanks to the nonsingularity of the matrices $\mathbf{P}_X$ and $\mathbf{A}(\boldsymbol{\mu})$. We impose its gradient with respect to $\mathbf{v}_N$ and evaluated at $\mathbf{u}_N$ to vanish.

By employing the definition of the norm $\|\cdot\|_{\mathbf{P}_X^{-1}}$ we obtain

$$\begin{aligned}
\mathbf{0} = \frac{\partial\{J(\mathbf{v}_N)\}}{\partial \mathbf{v}_N}(\mathbf{u}_N) &= \frac{\partial}{\partial \mathbf{v}_N}\Big\{(\mathbf{g}(\boldsymbol{\mu}))^T\mathbf{P}_X^{-1}\mathbf{g}(\boldsymbol{\mu}) \\
&\quad + \mathbf{v}_N^T\mathbf{V}^T\mathbf{A}^T(\boldsymbol{\mu})\mathbf{P}_X^{-1}\mathbf{A}(\boldsymbol{\mu})\mathbf{V}\mathbf{v}_N - 2(\mathbf{g}(\boldsymbol{\mu}))^T\mathbf{P}_X^{-1}\mathbf{A}(\boldsymbol{\mu})\mathbf{V}\mathbf{v}_N\Big\}(\mathbf{u}_N) \\
&= 2\mathbf{V}^T\mathbf{A}^T(\boldsymbol{\mu})\mathbf{P}_X^{-1}\mathbf{A}(\boldsymbol{\mu})\mathbf{V}\mathbf{u}_N - 2(\mathbf{g}(\boldsymbol{\mu}))^T\mathbf{P}_X^{-1}\mathbf{A}(\boldsymbol{\mu})\mathbf{V}\mathbf{u}_N \\
&= 2\widehat{\mathbf{A}}_N(\boldsymbol{\mu})\mathbf{u}_N - 2\hat{\mathbf{g}}_N(\boldsymbol{\mu}).
\end{aligned}$$

Therefore, $\mathbf{u}_N$ is such that

$$\widehat{\mathbf{A}}_N(\boldsymbol{\mu})\mathbf{u}_N = \hat{\mathbf{g}}_N(\boldsymbol{\mu}),$$

hence it coincides with the RB solution $\mathbf{z}_N(\boldsymbol{\mu})$, since the matrix $\widehat{\mathbf{A}}_N(\boldsymbol{\mu})$ is invertible. □

3.3.2 Assembling the aLSRB problem

When building the aLSRB approximation, it is essential to assume the affine dependence on $\boldsymbol{\mu}$ in the FE arrays (3.6), that is $\mathbf{D}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$, $\mathbf{f}(\boldsymbol{\mu})$, $\mathbf{r}(\boldsymbol{\mu})$ (separately) satisfy the affine assumption (1.44). In this way, the aLSRB algebraic structures can be written as

$$\begin{aligned}
\widehat{\mathbf{A}}_N(\boldsymbol{\mu}) &= \sum_{q_1,q_2=1}^{Q_a} \Theta_a^{q_1}(\boldsymbol{\mu})\Theta_a^{q_2}(\boldsymbol{\mu})\mathbf{V}^T(\mathbf{A}_h^{q_1})^T\mathbf{P}_X^{-1}\mathbf{A}_h^{q_2}\mathbf{V} \qquad (3.41)\\
&= \sum_{q_1,q_2=1}^{Q_a} \Theta_a^{q_1}(\boldsymbol{\mu})\Theta_a^{q_2}(\boldsymbol{\mu})\mathbf{A}_N^{q_1,q_2}\\
\hat{\mathbf{g}}_N(\boldsymbol{\mu}) &= \sum_{q_1=1}^{Q_a}\sum_{q_2=1}^{Q_g} \Theta_a^{q_1}(\boldsymbol{\mu})\Theta_g^{q}(\boldsymbol{\mu})\mathbf{V}^T(\mathbf{A}_h^{q_1})^T\mathbf{P}_X^{-1}\mathbf{g}^{q_2} \qquad (3.42)\\
&= \sum_{q_1=1}^{Q_a}\sum_{q_2=1}^{Q_g} \Theta_a^{q_1}(\boldsymbol{\mu})\Theta_g^{q_2}(\boldsymbol{\mu})\mathbf{g}_N^{q_1,q_2}.
\end{aligned}$$

In the G-RB case, the algebraic RB structures can be instead obtained similarly to (1.45)-(1.46). The matrices $\mathbf{A}_N^q$, $q = 1,\dots,Q_a$, $\mathbf{A}_N^{q_1,q_2} \in \mathbb{R}^{N\times N}$, $q_1, q_2 = 1,\dots,Q_a$, and the vectors $\mathbf{g}_N^q \in \mathbb{R}^N$, $q = 1,\dots,Q_g$, $\mathbf{g}_N^{q_1,q_2} \in \mathbb{R}^N$, $q_1 = 1,\dots,Q_a$, $q_2 = 1,\dots,Q_g$ can be precomputed and stored during the offline phase. During the online phase, only the sums in (3.41)-(3.42) and (1.61)-(1.62) must be calculated out to assemble the RB problem. Notice that the number of operations for building $\mathbf{A}_N(\boldsymbol{\mu})$ and $\mathbf{g}_N(\boldsymbol{\mu})$ in (1.61)-(1.62) depends linearly on the number of affine terms Q_a and Q_g for the G-RB method. On the other hand, the corresponding aLSRB structures $\widehat{\mathbf{A}}_N(\boldsymbol{\mu})$ and $\hat{\mathbf{g}}_N(\boldsymbol{\mu})$ in (3.41)-(3.42) depend quadratically Q_a and Q_g. Practically, employing the G-RB method softens the dependence on the number of affine terms, since less RB structures must be assembled and stored with respect to the aLSRB method. This advantage is also visible in the online phase, since the construction of the RB matrix and right hand side scales linearly with respect to Q_a and Q_g. However, the aLSRB matrices and right hand sides have a smaller dimension, since the velocity basis is not augmented, entailing a lower cost for computing and storing each array and for computing the solution of the RB system. Finally, notice that the affine decomposition (1.44) would not be exploitable in the case of standard LSRB method, due to the $\boldsymbol{\mu}-$dependence of the matrix $\mathbf{X}_h(\boldsymbol{\mu})$. Indeed, one would need also an affine decomposition of $\mathbf{X}_h^{-1}(\boldsymbol{\mu})$, which is generally not available since it is never explicitly assembled and its application is performed by solving a linear system where $\mathbf{X}_h(\boldsymbol{\mu})$ is at the left hand side.

In the numerical examples considered in this work, as well as in almost every problem of applied interest, the geometrical dependence of the computational domain on the parameter $\boldsymbol{\mu}$ is generally nonaffine, therefore an affine representation of $\mathbf{A}(\boldsymbol{\mu})$ and $\mathbf{g}(\boldsymbol{\mu})$ cannot be computed. To circumvent this problem both the empirical interpolation method (EIM) or its discrete variants DEIM and Matrix-DEIM offer, as in the case of elliptic problems, the possibility to recover an approximate affine decomposition. Furthermore, in our practical implementation, we run separately MDEIM on the matrices $\mathbf{D}(\boldsymbol{\mu})$ and $\mathbf{B}(\boldsymbol{\mu})$, meaning that we compute two basis, $\mathbf{D}^q \in \mathbb{R}^{N_h^u \times N_h^u}$, $q = 1, \dots, Q_d$ and $\mathbf{B}^q \in \mathbb{R}^{N_h^p \times N_h^u}$, $q = 1, \dots, Q_b$, of $\boldsymbol{\mu}-$independent matrices such that the following relations hold

$$\mathbf{D}(\boldsymbol{\mu}) \approx \sum_{q=1}^{Q_d} \tilde{\Theta}_d^q(\boldsymbol{\mu})\mathbf{D}^q, \qquad \mathbf{B}(\boldsymbol{\mu}) \approx \sum_{q=1}^{Q_b} \tilde{\Theta}_b^q(\boldsymbol{\mu})\mathbf{B}^q, \tag{3.43}$$

where the functions $\tilde{\Theta}_d^q : \mathcal{D} \to \mathbb{R}$, $q = 1, \dots, Q_d$ and $\tilde{\Theta}_b^q : \mathcal{D} \to \mathbb{R}$, $q = 1, \dots, Q_b$ are $\boldsymbol{\mu}-$dependent. Similarly for the right hand sides it holds

$$\mathbf{f}(\boldsymbol{\mu}) \approx \sum_{q=1}^{Q_f} \tilde{\Theta}_f^q(\boldsymbol{\mu})\mathbf{f}^q, \qquad \mathbf{r}(\boldsymbol{\mu}) \approx \sum_{q=1}^{Q_r} \tilde{\Theta}_r^q(\boldsymbol{\mu})\mathbf{r}^q, \tag{3.44}$$

with $\boldsymbol{\mu}-$dependent functions $\tilde{\Theta}_f^q : \mathcal{D} \to \mathbb{R}$, $q = 1, \dots, Q_f$ and $\tilde{\Theta}_r^q : \mathcal{D} \to \mathbb{R}$, $q = 1, \dots, Q_r$ and $\boldsymbol{\mu}$-independent basis functions $\mathbf{f}^q \in \mathbb{R}^{N_h^u}$, $q = 1, \dots, Q_f$ and $\mathbf{r}^q \in \mathbb{R}^{N_h^p}$, $q = 1, \dots, Q_r$. Then, the block matrices in (3.37) are such that

$$\widehat{\mathbf{D}}_N(\boldsymbol{\mu}) \approx \sum_{q_1,q_2=1}^{Q_d} \tilde{\Theta}_d^{q_1}(\boldsymbol{\mu})\tilde{\Theta}_d^{q_2}(\boldsymbol{\mu})\widehat{\mathbf{D}}_{N1}^{q_1,q_2} + \sum_{q_1,q_2=1}^{Q_b} \tilde{\Theta}_b^{q_1}(\boldsymbol{\mu})\tilde{\Theta}_b^{q_2}(\boldsymbol{\mu})\widehat{\mathbf{D}}_{N2}^{q_1,q_2}, \tag{3.45}$$

$$\widehat{\mathbf{B}}_N(\boldsymbol{\mu}) \approx \sum_{q_d=1}^{Q_d} \sum_{q_b=1}^{Q_b} \tilde{\Theta}_d^{q_d}(\boldsymbol{\mu})\tilde{\Theta}_b^{q_b}(\boldsymbol{\mu})\widehat{\mathbf{B}}_N^{q_b q_d}, \tag{3.46}$$

$$\widehat{\mathbf{C}}_N(\boldsymbol{\mu}) \approx \sum_{q_1,q_2=1}^{Q_b} \tilde{\Theta}_b^{q_1}(\boldsymbol{\mu})\tilde{\Theta}_b^{q_2}(\boldsymbol{\mu})\widehat{\mathbf{C}}_N^{q_1,q_2}, \tag{3.47}$$

and the $\boldsymbol{\mu}$-independent RB matrices $\widehat{\mathbf{D}}_{N1}^{q_1,q_2} \in \mathbb{R}^{N_u \times N_u}$, $\widehat{\mathbf{D}}_{N2}^{q_1,q_2} \in \mathbb{R}^{N_u \times N_u}$, $\widehat{\mathbf{B}}_{N}^{q_b q_d} \in \mathbb{R}^{N_p \times N_u}$ and $\widehat{\mathbf{C}}_{N}^{q_1,q_2}$, $q_1, q_2 = 1, \dots, Q_b, \in \mathbb{R}^{N_p \times N_p}$ are defined similarly to (3.41) by exploiting the approximated affine decompositions (3.44) and can be precomputed and stored during the offline phase, such that only the evaluation of the coefficients and the sums in (3.45)-(3.46)-(3.47) need to be performed online.

3.3.3 On the choice of $\mathbf{P}_X$

A natural question arising in this context regards the choice of the matrix $\mathbf{P}_X$, since this directly affects the values of the constants c and C and the computational efficiency of the proposed aLSRB method. These constants play indeed a relevant role in the conditioning of the aLSRB approximation. Moreover, it is clear that $c/C \leq 1$ and by taking $\mathbf{P}_X = \mathbf{X}_h(\boldsymbol{\mu})$ we would have the optimal case $c/C = 1$, hence recovering the standard LSRB method. Therefore, $\mathbf{P}_X$ should be chosen as close as possible to $\mathbf{X}_h(\boldsymbol{\mu})$, however it has to be $\boldsymbol{\mu}$-independent for the sake of computational efficiency. The following results offer some insights on how to properly choose the matrix $\mathbf{P}_X$.

Lemma 3.3.1. *Let the assumption 3.3.1 hold. The optimal values for the constants* $C \geq c$ *satisfying* (3.30) *are*

$$C = \|\mathbf{P}_X^{-1/2}(\mathbf{X}_h(\boldsymbol{\mu}))^{1/2}\|_{\mathbf{P}_X}, \qquad c = \|(\mathbf{X}_h(\boldsymbol{\mu}))^{-1/2}\mathbf{P}_X^{1/2}\|_{\mathbf{X}_h(\boldsymbol{\mu})}^{-1}. \tag{3.48}$$

Proof. Since $\mathbf{X}_h$ and $\mathbf{P}_X$ are symmetric and positive definite, for any $\mathbf{y} \in \mathbb{R}^{N_h}$ it holds

$$\begin{aligned} \|\mathbf{y}\|_{\mathbf{X}_h}^2 &= \left(\mathbf{X}_h^{1/2}\mathbf{y}, \mathbf{X}_h^{1/2}\mathbf{y}\right)_2 = \left(\mathbf{P}_X^{-1/2}\mathbf{P}_X\mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\mathbf{y}, \mathbf{X}_h^{1/2}\mathbf{y}\right)_2 \\ &= \left(\mathbf{P}_X\mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\mathbf{y}, \mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\mathbf{y}\right)_2 = \left(\mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\mathbf{y}, \mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\mathbf{y}\right)_{\mathbf{P}_X} \\ &= \|\mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\mathbf{y}\|_{\mathbf{P}_X}^2 \leq \|\mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\|_{\mathbf{P}_X}^2 \|\mathbf{y}\|_{\mathbf{P}_X}^2, \end{aligned}$$

and there exists an element $\mathbf{y}_0 \in \mathbb{R}^{N_h}$ where equality is reached. This leads to an optimal $C = \|\mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\|_{\mathbf{P}_X}$. Similarly, by inverting the roles of $\mathbf{P}_X$ and $\mathbf{X}_h$ and following the same argument, we have that $c = \|\mathbf{X}_h^{-1/2}\mathbf{P}_X^{1/2}\|_{\mathbf{X}_h}^{-1}$. $\square$

Hereon, we set C and c equal to their optimal values (3.48).

Lemma 3.3.2. *Let the assumption 3.3.1 hold. The two constants* $C \geq c > 0$ *satisfying* (3.30) *and* (3.48) *are such that*

$$\frac{c}{C} = \left[\mathcal{K}_{\mathbf{X}_h}(\mathbf{P}_X^{-1}\mathbf{X}_h(\boldsymbol{\mu}))\right]^{-1/2} = \left[\mathcal{K}_2(\mathbf{P}_X^{-1/2}\mathbf{X}_h\mathbf{P}_X^{-1/2})\right]^{-1/2}. \tag{3.49}$$

Proof. We rewrite the optimal values for C and c as it follows

$$\begin{aligned}
C^2 &= \|\mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\|^2_{\mathbf{P}_X} = \sup_{\mathbf{y}\in\mathbb{R}^{N_h},\mathbf{y}\neq\mathbf{0}} \frac{\|\mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\mathbf{y}\|^2_{\mathbf{P}_X}}{\|\mathbf{y}\|^2_{\mathbf{P}_X}} \\
&= \sup_{\mathbf{y}\in\mathbb{R}^{N_h},\mathbf{y}\neq\mathbf{0}} \frac{(\mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\mathbf{y},\mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\mathbf{y})_{\mathbf{P}_X}}{(\mathbf{y},\mathbf{y})_{\mathbf{P}_X}} = \sup_{\mathbf{y}\in\mathbb{R}^{N_h},\mathbf{y}\neq\mathbf{0}} \frac{(\mathbf{X}_h^{1/2}\mathbf{y},\mathbf{X}_h^{1/2}\mathbf{y})_2}{(\mathbf{P}_X^{1/2}\mathbf{y},\mathbf{P}_X^{1/2}\mathbf{y})_2} \\
&= \sup_{\mathbf{y}\in\mathbb{R}^{N_h},\mathbf{y}\neq\mathbf{0}} \frac{(\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2}\mathbf{P}_X^{1/2}\mathbf{y},\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2}\mathbf{P}_X^{1/2}\mathbf{y})_2}{(\mathbf{P}_X^{1/2}\mathbf{y},\mathbf{P}_X^{1/2}\mathbf{y})_2} \\
&= \sup_{\mathbf{w}\in\mathbb{R}^{N_h},\mathbf{w}\neq\mathbf{0}} \frac{(\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2}\mathbf{w},\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2}\mathbf{w})_2}{(\mathbf{w},\mathbf{w})_2} = \sup_{\mathbf{w}\in\mathbb{R}^{N_h},\mathbf{w}\neq\mathbf{0}} \frac{\|\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2}\mathbf{w}\|_2^2}{\|\mathbf{w}\|_2^2} \\
&= \|\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2}\|_2^2. \qquad (3.50)
\end{aligned}$$

Similarly, we have that $\|\mathbf{X}_h^{-1/2}\mathbf{P}_X^{1/2}\|_{\mathbf{X}_h} = \|\mathbf{P}_X^{1/2}\mathbf{X}_h^{-1/2}\|_2$, yielding

$$\begin{aligned}
\frac{c}{C} &= \Big(\|\mathbf{X}_h^{-1/2}\mathbf{P}_X^{1/2}\|_{\mathbf{X}_h}\|\mathbf{P}_X^{-1/2}\mathbf{X}_h^{1/2}\|_{\mathbf{P}_X}\Big)^{-1} = \Big(\|\mathbf{P}_X^{1/2}\mathbf{X}_h^{-1/2}\|_2\|\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2}\|_2\Big)^{-1} \\
&= \Big[\mathcal{K}_2(\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2})\Big]^{-1} = \Big[\mathcal{K}_2(\mathbf{P}_X^{1/2}\mathbf{X}_h^{-1/2})\Big]^{-1},
\end{aligned}$$

where the last two relations are both used to find different equalities. Next, by recalling the definition of condition number (with respect to the Euclidean norm) $\mathcal{K}_2$ for a matrix, we obtain

$$\begin{aligned}
\mathcal{K}_2(\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2}) &= \sqrt{\frac{\lambda_{\max}\Big((\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2})^T\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2}\Big)}{\lambda_{\min}\Big((\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2})^T\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1/2}\Big)}} = \sqrt{\frac{\lambda_{\max}\Big(\mathbf{P}_X^{-1/2}\mathbf{X}_h\mathbf{P}_X^{-1/2}\Big)}{\lambda_{\min}\Big(\mathbf{P}_X^{-1/2}\mathbf{X}_h\mathbf{P}_X^{-1/2}\Big)}} \\
&= \sqrt{\mathcal{K}_2\Big(\mathbf{P}_X^{-1/2}\mathbf{X}_h\mathbf{P}_X^{-1/2}\Big)},
\end{aligned}$$

which verifies the second equality of (3.49). On the other hand we have

$$\begin{aligned}
\mathcal{K}_2(\mathbf{P}_X^{1/2}\mathbf{X}_h^{-1/2}) &= \sqrt{\mathcal{K}_2\Big(\mathbf{X}_h^{-1/2}\mathbf{P}_X\mathbf{X}_h^{-1/2}\Big)} \\
&= \sqrt{\|\mathbf{X}_h^{-1/2}\mathbf{P}_X\mathbf{X}_h^{-1/2}\|_2\|\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1}\mathbf{X}_h^{1/2}\|_2} \\
&= \sqrt{\|\mathbf{X}_h^{-1}\mathbf{P}_X\|_{\mathbf{X}_h}\|\mathbf{P}_X^{-1}\mathbf{X}_h\|_{\mathbf{X}_h}} = \sqrt{\mathcal{K}_{\mathbf{X}_h}(\mathbf{P}_X^{-1}\mathbf{X}_h)},
\end{aligned}$$

where we used that

$$\|\mathbf{X}_h^{-1/2}\mathbf{P}_X\mathbf{X}_h^{-1/2}\|_2 = \|\mathbf{X}_h^{-1}\mathbf{P}_X\|_{\mathbf{X}_h}, \qquad \|\mathbf{X}_h^{1/2}\mathbf{P}_X^{-1}\mathbf{X}_h^{1/2}\|_2 = \|\mathbf{P}_X^{-1}\mathbf{X}_h\|_{\mathbf{X}_h};$$

these latter relationships are verified similarly to (3.50), and their proof can therefore be omitted. □

Lemma 3.3.2 provides further insights about the choice of the matrix $\mathbf{P}_X$. Indeed, this latter should be chosen in such a way that the condition number of the preconditioned matrix $\mathbf{P}_X^{-1}\mathbf{X}_h(\boldsymbol{\mu})$ does not depend on the mesh size h; in other words, $\mathbf{P}_X$ should be an optimal preconditioner for $\mathbf{X}_h(\boldsymbol{\mu})$. If this is not the case, the value of the stability constant of the RB approximation $\beta_{\mathbf{P}_X,N}(\boldsymbol{\mu})$ may depend on h. Furthermore, if we set up our RB approximation in a HPC environment, employing a mesh partitioner to divide the computational domain among the processors, it is also advisable to choose $\mathbf{P}_X$ such that $\frac{c}{C}$ does not depend on the size H of the subdomains, i.e. $\mathbf{P}_X$ should be a scalable preconditioner for $\mathbf{X}_h(\boldsymbol{\mu})$.

In our numerical experiments $\mathbf{P}_X$ is chosen either as $\mathbf{P}_X = \mathbf{X}_h^0$, i.e. as the norm matrix in the reference domain, or as a block diagonal preconditioner of $\mathbf{X}_h^0$, where the two blocks are generated as symmetric and positive definite preconditioners $\mathbf{P}_{\mathbf{X}_u} \in \mathbb{R}^{N_h^u \times N_h^u}$ of $\mathbf{X}_u^0$ and $\mathbf{P}_{\mathbf{X}_p} \in \mathbb{R}^{N_h^p \times N_h^p}$ of $\mathbf{X}_p^0$, respectively.

3.4 Numerical experiments with aLSRB approximation

In this section we provide numerical results showing the capabilities of the proposed aLSRB approximation. In particular, we compare the G-RB method (with velocity enrichment) and the aLSRB method in the case of large-scale Stokes flows in a cylindrical domain which is nonaffinely parametrized. The deformation is not analitically known, since it is retrieved as the solution of an additional FE problem which harmonically extends in the interior of the domain the datum prescribed on a Dirichlet boundary. In the following sections, we present the setup of the problem.

3.4.1 Test case setting: Stokes problem in a parametrized cylinder

We consider the Stokes equations in a parameter dependent domain $\Omega(\boldsymbol{\mu}) \subset \mathbb{R}^3$, which is obtained by deforming a reference domain

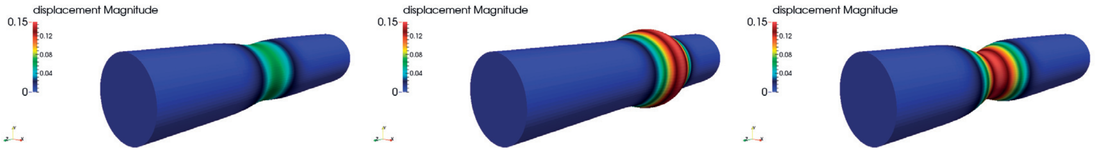
$$\Omega^0 = \{\vec{x} \in \mathbb{R}^3 : \ x_1^2 + x_1^2 < 0.25, \, x_3 \in (0,5)\}$$

by means of a displacement $\vec{d}(\boldsymbol{\mu})$ obtained as the harmonic extension of a boundary displacement. More specifically, we set

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

where $\vec{d}(\boldsymbol{\mu})$ solves the following (vector) Laplace equation

$$\begin{cases} -\Delta \vec{d}(\boldsymbol{\mu}) = \vec{0} & \text{in } \Omega^0 \\ \vec{d}(\boldsymbol{\mu}) = \vec{h}(\boldsymbol{\mu}) & \text{on } \partial\Omega^0. \end{cases} \tag{3.51}$$



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

Figure 3.1 – Displacement for different values of $\boldsymbol{\mu}$.

In our numerical experiments we take $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathcal{D} = [-0.3,\, 0.3] \times [2,\, 3]$ and a Dirichlet datum of the form

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

entailing a deformation of the cylinder by narrowing or enlarging (according to the sign of μ_1) its section in different positions along the coordinate x_3 (according to the value of μ_2). Since the solution $\vec{d}(\boldsymbol{\mu})$ of (3.51) is not known a-priori, we compute its numerical approximation $\vec{d}_h(\boldsymbol{\mu})$ by writing the variational form of problem (3.51) and by employing the FE method. We denote by $\mathbf{d}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^d}$ the solution of the corresponding FE linear system.

Moreover, 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

$$\begin{cases} -\Delta \vec{r}_{\vec{g}_D}(\boldsymbol{\mu}) = \vec{0} & \text{in } \Omega(\boldsymbol{\mu}) \\ \vec{r}_{\vec{g}_D}(\boldsymbol{\mu}) = \vec{g}_D(\boldsymbol{\mu}) & \text{on } \Gamma_{in}(\boldsymbol{\mu}) \\ \vec{r}_{\vec{g}_D}(\boldsymbol{\mu}) = \vec{0} & \text{on } \Gamma_{w}(\boldsymbol{\mu}) \\ \dfrac{\partial \vec{r}_{\vec{g}_D}(\boldsymbol{\mu})}{\partial \vec{n}(\boldsymbol{\mu})} = \vec{0} & \text{on } \Gamma_{out}(\boldsymbol{\mu}), \end{cases} \tag{3.52}$$

which is an harmonic extension of the Dirichlet data in (3.1). Here $\vec{g}_D$ is a parabolic profile such that the flow rate at the inlet is equal to 1. Problem (3.52) as well is discretized with the FE method with second order polynomials ($\mathcal{P}_2$) basis functions, leading to a parametrized linear system whose solution $\mathbf{r}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^u}$ is the approximated lifting functions. In Figure 3.1, the deformation $\mathbf{d}(\boldsymbol{\mu})$ is reported for three different values of $\boldsymbol{\mu} \in \mathcal{D}$. In the numerical experiments we present, Taylor-Hood FE ($\mathcal{P}_2 - \mathcal{P}_1$), with a mesh leading to $N_h = N_h^u + N_h^p = 1'503'280 + 64'943 = 1'568'223$ degrees of freedom, are employed for the Stokes equations.

FE simulation setup

For any parameter $\boldsymbol{\mu}$ considered, we solve the FE problems to approximate the displacement $\vec{d}(\boldsymbol{\mu})$ of problem (3.51) and the lifting function $\vec{r}_{\vec{g}_D}(\boldsymbol{\mu})$ of problem (3.52). The associated algebraic systems are solved by the preconditioned CG method, with a tolerance on the Euclidean norm of the residual (rescaled with the Euclidean norm of the right hand side) of 10^{-8}. An algebraic multigrid (AMG) preconditioner from the ML package of `Trilinos`, see [Gee et al., 2006], is employed in this respect. Once computed the deformation for a new parameter value $\boldsymbol{\mu}$, we employ a move-mesh tool to shape the computational domain and assemble the FE arrays. This ensures that the meshes for different instances of the parameter $\boldsymbol{\mu}$ are topologically equivalent and there is a one-to-one correspondence between degrees of freedom.

The linear system (3.6) resulting from the FE discretization of the Stokes equations is solved with the preconditioned flexible GMRES (FGMRES) method, where the preconditioner is the Pressure Mass Matrix (PMM) operator, which exploits the block structure of (3.7) and employs the rescaled mass matrix in pressure to approximate the Schur complement. It entails at every iteration the solution of a problem for the velocity (involving the velocity stiffness matrix) and one for the pressure (involving the pressure mass matrix). Both linear systems are solved inexactly with the preconditioned CG method, where the preconditioner is still the AMG preconditioner from the ML package of Trilinos, this time with a tolerance on the Euclidean norm of the residual (rescaled with the Euclidean norm of the right hand side) of 10^{-5}. Notice that we employed the FGMRES (instead of regular GMRES) due to the use of inner iterations for the problems involving the velocity and the pressure, which, as a matter of fact, yield an iteration dependent preconditioner. The PMM preconditioner provides satisfactory results in the case of the Stokes equations, cf. [Rehman et al., 2011, Elman et al., 2005, Elman and Silvester, 1996]. Finally, in order to compute the FE solution with the flexible GMRES method, up to a final tolerance of 10^{-8}, our solver requires on average of 38.0 seconds, which also accounts for the time for deforming the domain, building the lifting function, the PMM preconditioner and the FE solution. In particular, computing the deformation $\mathbf{d}(\boldsymbol{\mu})$ and the lifting function $\mathbf{r}(\boldsymbol{\mu})$ requires 2.5 seconds (6.5% of the FE simulation).

RB simulation setup

During the *offline* phase, we explore the parameter domain $\mathcal{D}$ for building our RB approximation. In particular we perform the following steps:

- we randomly choose a set of $n_s = 150$ parameters $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s} \subset \mathcal{D}$; then we compute the velocity snapshots $\{\mathbf{u}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ and pressure snapshots $\{\mathbf{p}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ by solving the corresponding linear system (3.6). Next, we build the RB space V_N by separately computing a basis $\mathbf{V}_{N_u}$ for the velocity and $\mathbf{V}_{N_p}$ for the pressure, by plugging in the POD the same tolerance $\varepsilon_{\text{POD}} = \delta_{RB}$ in both cases. If the Galerkin-RB method with velocity enrichment is employed, we also compute $n_s = 150$ pressure supremizer snapshots $\{\mathbf{t}_p(\mathbf{p}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i)\}_{i=1}^{n_s}$. Since in general we do not retain the same number of basis functions for the velocity and pressure RB spaces, we use a tolerance also for computing the pressure supremizer basis functions. With this aim, we employ POD with $\varepsilon_{\text{POD}} = \delta_{RB}/10$ to build the supremizer basis $\mathbf{V}_{N_s}$, which represents a heuristic criterion to provide a stable G-RB problem.

- we compute a basis to affinely approximate $\mathbf{f}(\boldsymbol{\mu})$, $\mathbf{r}(\boldsymbol{\mu})$ (with DEIM) and $\mathbf{D}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$ (with MDEIM), by taking $n_s = 100$ snapshots for each of these quantities and a tolerance δ_{DEIM} to be used in the POD;

In the *online* phase, we solve the RB linear system resulting from either the G-RB or aLSRB methods, and analyzed their performances in terms of accuracy and efficiency with respect to the tolerances δ_{RB} (or the number of basis functions N) and δ_{DEIM}, by choosing δ_{RB}, $\delta_{\text{DEIM}} = 10^{-l}$, $l = 2, 3, 4, 5, 6$. We evaluate the accuracy of the RB solutions $\mathbf{z}_N(\boldsymbol{\mu})$ in terms of the rescaled RB residual

$$r_{\text{RB}}(\boldsymbol{\mu}) = \frac{\|\mathbf{g}(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu})\mathbf{V}\mathbf{z}_N(\boldsymbol{\mu})\|_{\mathbf{X}_h^{-1}(\boldsymbol{\mu})}}{\|\mathbf{g}(\boldsymbol{\mu})\|_{\mathbf{X}_h^{-1}(\boldsymbol{\mu})}},$$

averaging the results obtained for $N_{onl} = 100$ parameters (denoting the average r_{RB}), different from the one employed within the offline phase. For the aLSRB method, we present the results for two choices of the matrix $\mathbf{P}_X$:

- $\mathbf{P}_X = \mathbf{X}_h^0$, i.e. we approximate $\mathbf{X}_h(\boldsymbol{\mu})$ with the matrix norm on the reference domain Ω^0. With this aim, in the offline phase, we need to solve Q_a FE linear systems with $\mathbf{X}_h^0$ on the left hand side to compute the affine terms $\mathbf{A}_N^{q_1,q_2}$, $q_1, q_2 = 1, \dots, Q_a$. These linear systems are solved with the CG method preconditioned with AMG, up to a tolerance of 10^{-8} on the Euclidean norm of the relative residual;

- $\mathbf{P}_X = \mathbf{P}_{\mathbf{X}_h^0}$, i.e. we take the preconditioner $\mathbf{P}_{\mathbf{X}_u^0}$ of $\mathbf{X}_h(\boldsymbol{\mu})$, which has a block structure $\mathbf{P}_{\mathbf{X}_h^0} = diag(\mathbf{P}_{\mathbf{X}_u^0}, \mathbf{P}_{\mathbf{X}_p^0})$, where $\mathbf{P}_{\mathbf{X}_u} \in \mathbb{R}^{N_h^u \times N_h^u}$ (resp. $\mathbf{P}_{\mathbf{X}_p} \in \mathbb{R}^{N_h^p \times N_h^p}$) is a symmetric and positive definite AMG preconditioner of $\mathbf{X}_u^0$ (resp. $\mathbf{X}_p^0$).

Both choices lead to a matrix $\mathbf{P}_X$ which does not depend on $\boldsymbol{\mu}$. Notice that for any new parameter $\boldsymbol{\mu}$ considered online, we solve the FE linear system for computing the deformation $\mathbf{d}(\boldsymbol{\mu})$ and the lifting function $\mathbf{r}(\boldsymbol{\mu})$. Alternatively, we could compute the RB approximations of $\mathbf{d}(\boldsymbol{\mu})$ and $\mathbf{r}(\boldsymbol{\mu})$, to be exploited during the online phase, similarly to what has been proposed in [Manzoni and Negri, 2017] for the parametrized Helmholtz scattering problem.

3.4.2 Numerical results

Offline phase

In Tables 3.1-3.2 we show the offline time required to build the structures of the RB approximations when using $\delta_{RB} = \delta_{\mathrm{DEIM}} = 10^{-6}$ (comparable results hold when bigger tolerances are used). We recall that δ_{RB} is used within POD to build the velocity and pressure RB spaces, while δ_{DEIM} for building an affine approximation of the FE blocks of $\mathbf{A}(\boldsymbol{\mu})$ and $\mathbf{g}$ in the (M)DEIM algorithm.

In the first table, we report the computational times to build the (M)DEIM basis functions which provide an affine approximation of the FE matrices and right hand sides; this effort is common to both the G-RB and aLSRB methods. In the second table, the total time of the offline computation is reported, together with its splitting into: snapshots computation, POD and RB affine arrays construction. Snapshots computation is the most demanding phase, and is particularly expensive if the G-RB method is employed, since it entails the additional computation of n_s pressure supremizer snapshots $\{\mathbf{t}_p(\mathbf{p}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i)\}_{i=1}^{n_s}$. The second phase, involving the POD to build the RB spaces, only requires a tiny percentage of the offline time for all the three methods considered, however also in this case, the two variants of the aLSRB method need a shorter time than the G-RB method, because they require only the construction of the velocity and pressure spaces $\mathbf{V}_{N_u}$ and $\mathbf{V}_{N_p}$, while in the G-RB case the pressure supremizer space $\mathbf{V}_{N_s}$ must also be constructed. Concerning the construction of the affine RB matrices and vectors, the G-RB method scales linearly with the number (Q_a and Q_f) of affine terms of the FE matrices and right hand sides, yielding a computational time which is shorter than the one obtained with the aLSRB methods for this phase. However, there is also a significant difference between the two variants of aLSRB method. By employing $\mathbf{P}_X = \mathbf{X}_h^0$, for assembling the affine terms $\mathbf{A}_N^{q_1,q_2}$, $q_1, q_2 = 1, \dots, Q_a$, a FE linear system must be solved for each combination of the N RB functions $\{\boldsymbol{\xi}_i\}_{i=1}^N$ and Q_a affine terms $\{\mathbf{A}_h^q\}_{q=1}^{Q_a}$, leading to $N \cdot Q_a$ FE linear systems. On the other hand, by employing $\mathbf{P}_X = \mathbf{P}_{\mathbf{X}_h^0}$, only $N \cdot Q_a$ applications of $\mathbf{P}_{\mathbf{X}_h^0}^{-1}$ need to be performed, boosting the computation of the affine RB structures. Finally, the lowest offline time is required by the aLSRB method where $\mathbf{P}_X = \mathbf{P}_{\mathbf{X}_h^0}$ is employed, performing the offline phase in about 81% of the time required by the aLSRB method with $\mathbf{P}_X = \mathbf{X}_h^0$ and 96% of the time required by the G-RB method. This is due to the fact that it does not require the construction of the pressure supremizing snapshots to augment the velocity RB space and to cheaply construct the RB affine arrays.

In Figure 3.2 the number of RB functions (left) and (M)DEIM affine terms are reported as function of the tolerances δ_{RB} and δ_{DEIM}, respectively. The number of pressure RB functions is the same for the G-RB and aLSRB method, however the number of velocity basis functions doubles in the former case, due to the velocity enrichment required to ensure the well-posedness of the resulting G-RB approximation.

Table 3.1 – Computational time (seconds) to build (M)DEIM basis with $\delta_{\text{DEIM}} = 10^{-6}$.

MDEIM - $\mathbf{D}(\boldsymbol{\mu})$	**MDEIM - $\mathbf{B}(\boldsymbol{\mu})$**	**DEIM - $\mathbf{f}(\boldsymbol{\mu})$**	**DEIM - $\mathbf{r}(\boldsymbol{\mu})$**	**Total (M-)DEIM**
362.6	249.4	326.7	321.3	1260.0

Table 3.2 – Computational time (seconds) to build RB approximation with $\delta_{RB} = 10^{-6}$.

	G-RB	**aLSRB ($\mathbf{X}_h^0$)**	**aLSRB ($\mathbf{P}_{\mathbf{X}_h^0}$)**
Snapshots computation	6102.2	5699.4	5699.4
POD	3.5	2.1	2.1
Affine arrays construction	19.6	1789.8	153.3
Total (M)DEIM	1260.0	1260.0	1260.0
Total offline phase	7385.3	8751.3	7114.8

Online phase

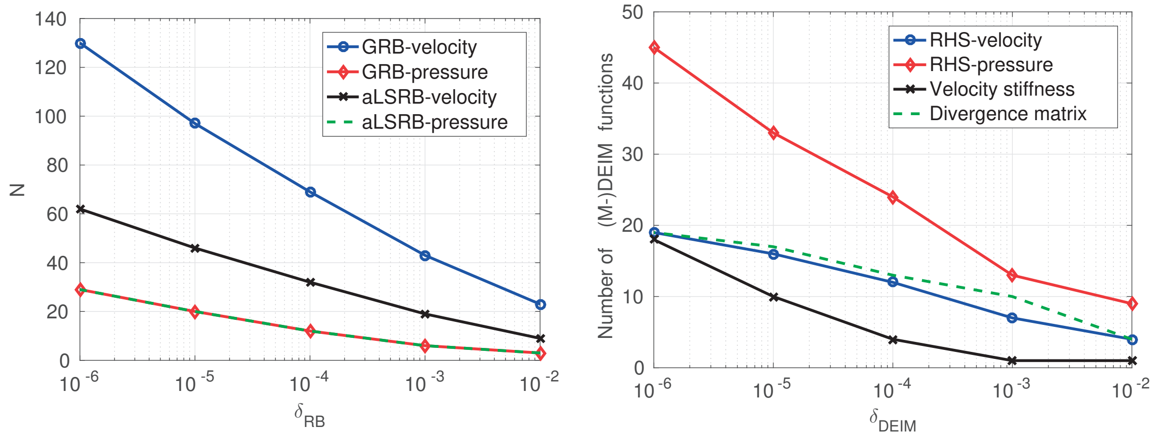
In Figure 3.3, 3.4 and 3.5 the FE solution computed for different values of the parameter and the corresponding errors obtained with the G-RB method and the aLSRB method with $\mathbf{P}_X = \mathbf{X}_h^0$ are shown (the aLSRB method with $\mathbf{P}_X = \mathbf{P}_{\mathbf{X}_h^0}$ provides similar results).

The proposed aLSRB method, either with $\mathbf{P}_X = \mathbf{X}_h^0$ or $\mathbf{P}_X = \mathbf{P}_{\mathbf{X}_h^0}$, allows to obtain an exponential decay of the residual r_{RB} with respect to the number of RB functions N, see Figure 3.6. A tolerance $\delta_{\text{DEIM}} = 10^{-8}$ has been used for (M)DEIM algorithms, in order to make the error induced by affinely approximating the FE arrays negligible.

An analysis of the convergence of the residual r_{RB} with varying both the tolerances δ_{RB}, $\delta_{\text{DEIM}} = 10^{-l}$, $l = 2, 3, 4, 5, 6$ is reported in Figure 3.7 for the G-RB and the two variants of the aLSRB methods. By using the same tolerances δ_{DEIM} and δ_{RB}, the aLSRB method allows to compute a more accurate solution of about 1 order of magnitude during the online phase. Moreover, notice that by using the same δ_{DEIM} for the aLSRB methods and the G-RB method, the latter requires a lower tolerance δ_{RB} to reach a solution with the same accuracy, yielding a much larger number N of RB functions. Obtaining a more accurate solution with the aLSRB method is an expected result, since the standard LSRB method seeks a RB approximation minimizing the $\mathbf{X}_h^{-1}(\boldsymbol{\mu})$ norm of the residual, and the aLSRB method provides a RB approximation minimizing its $\mathbf{P}_X^{-1}$ norm, where $\mathbf{P}_X^{-1} \approx \mathbf{X}_h^{-1}(\boldsymbol{\mu})$, as shown in Proposition 3.3.2.

In Figure 3.7, the computational time required to assemble and solve the RB problem is reported for the three methods by varying both the tolerances δ_{RB}, $\delta_{\text{DEIM}} = 10^{-l}$, $l = 2, 3, 4, 5, 6$. Depending on the desired level of accuracy and the RB method employed, the computational time required to solve the RB problem online ranges from 3.75 to 4.3 seconds. Therefore, a solution accurate up to an error of 0.01% on the FE residual r_{RB} is computed in a time ranging from 10% to 12% of the time required by the FE simulation.

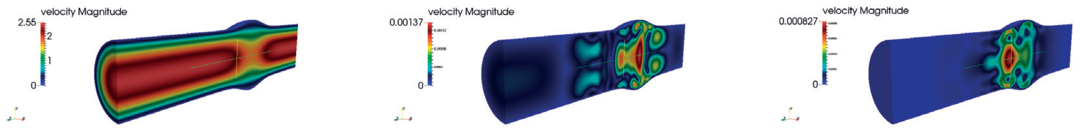
Notice however that the online computational time accounts also for the time employed for assembling and solving the FE problems to compute the deformation $\mathbf{d}(\boldsymbol{\mu})$ and the lifting



(a) RB.

(b) (M)DEIM.

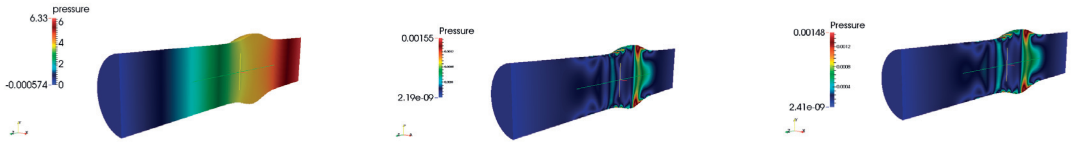
Figure 3.2 – RB and (M)DEIM functions vs δ_{RB}, $\delta_{\text{DEIM}} = 10^{-l}$, $l = 2, 3, 4, 5, 6$.



(a) FE velocity magnitude.

(b) G-RB velocity error magnitude.

(c) aLSRB velocity error magnitude.



(d) FE pressure.

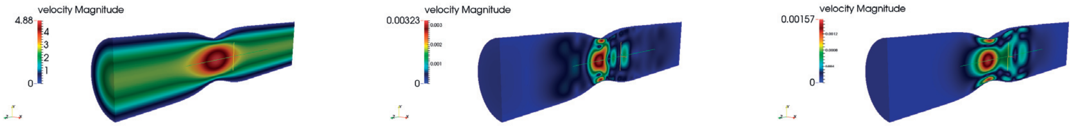
(e) G-RB norm of pressure error.

(f) aLSRB norm of pressure error.

Figure 3.3 – FE solution and G-RB and aLSRB errors for $\boldsymbol{\mu} = (2, -0.3)$.

function $\mathbf{r}(\boldsymbol{\mu})$, which on average requires 2.5 seconds in total. In our implementation this is included in the assembly of the RB matrix, whose required computational time is reported in Figure 3.9. By substituting in the simulation pipeline the assembly and solution of the FE problems to compute $\mathbf{d}(\boldsymbol{\mu})$ and $\mathbf{r}(\boldsymbol{\mu})$ with a less expensive model, e.g. by using a cheap RB approximation, one can compute an accurate solution with the aLSRB method, which needs only 5% of the time required by the FE simulation.

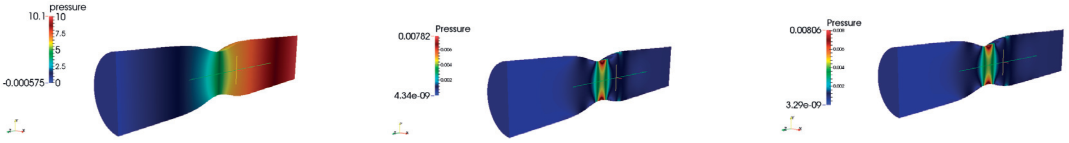
In Table 3.3, for the three methods examined, we compare the minimum time to determine a RB approximation whose residual r_{RB} is lower than a fixed target accuracy. The two versions of the aLSRB method again reach a better accuracy in a lower time. The 'x' in the G-RB column states that the accuracy 10^{-4} cannot be reached when this method



(a) FE velocity magnitude.

(b) G-RB velocity error magnitude.

(c) aLSRB velocity error magnitude.

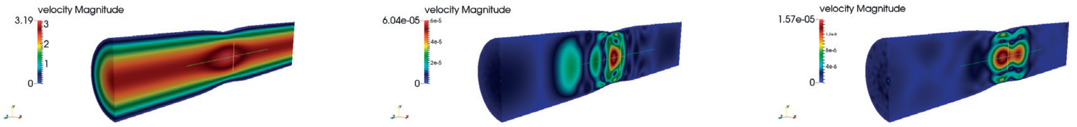


(d) FE pressure.

(e) G-RB norm of pressure error.

(f) aLSRB norm of pressure error.

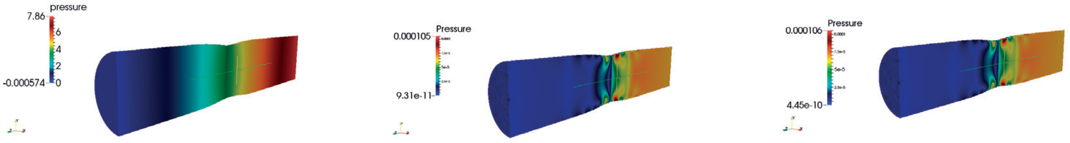
Figure 3.4 – FE solution and G-RB and aLSRB errors for $\boldsymbol{\mu} = (3, 0.3)$.



(a) FE velocity magnitude.

(b) G-RB velocity error magnitude.

(c) aLSRB velocity error magnitude.



(d) FE pressure.

(e) G-RB norm of pressure error.

(f) aLSRB norm of pressure error.

Figure 3.5 – FE solution and G-RB and aLSRB errors for $\boldsymbol{\mu} = (2.7, 0.12)$.

is used with the given tolerance values δ_{RB}, $\delta_{\text{DEIM}} = 10^{-l}$, $l = 2, 3, 4, 5, 6$. Therefore, one should further decrease δ_{RB} and δ_{DEIM} to compute a more accurate solution when employing the G-RB method, yes increasing assembling and solving costs even further.

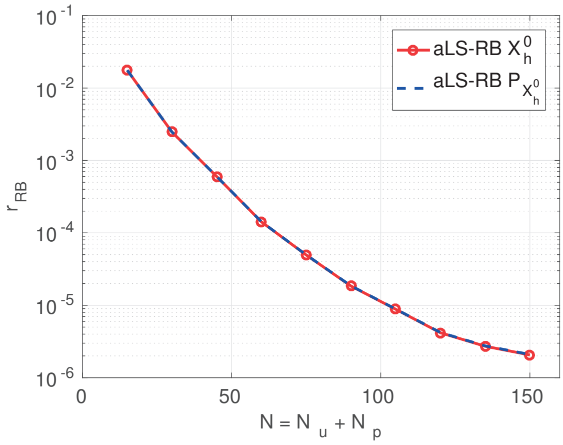
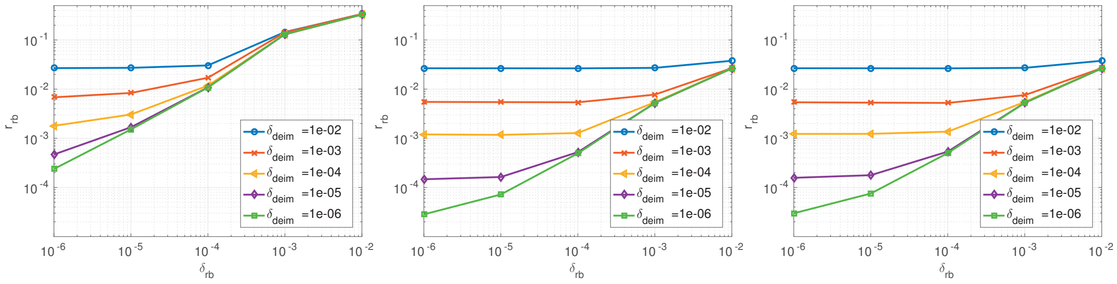
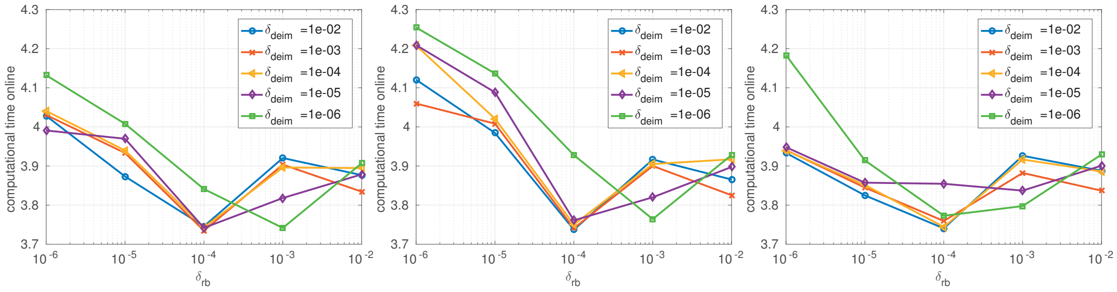


Figure 3.6 – Convergence of the residual in norm $\mathbf{X}_h^{-1}(\boldsymbol{\mu})$ vs the number of basis functions $N = N_u + N_p$ for the two case aLSRB (with $\mathbf{X}_h^0$ and $\mathbf{P}_{\mathbf{X}_h^0}$). Results computed with $\delta_{\text{DEIM}} = 10^{-8}$.



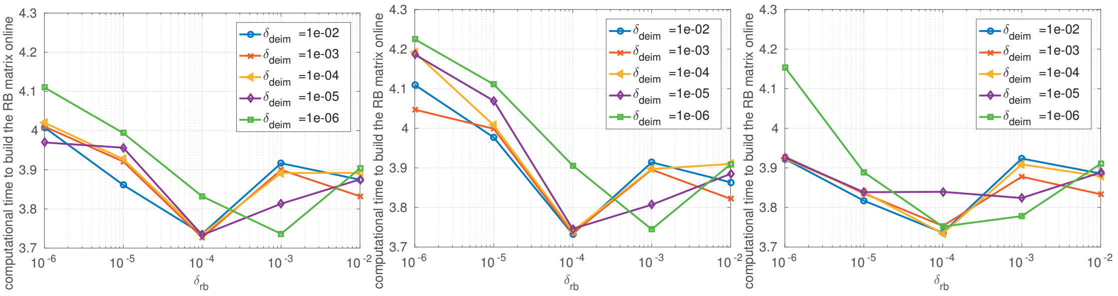
(a) GRB. (b) aLSRB ($\mathbf{X}_h^0$). (c) aLSRB ($\mathbf{P}_{\mathbf{X}_h^0}$).

Figure 3.7 – Residuals in norm $\mathbf{X}_h^{-1}(\boldsymbol{\mu})$ vs δ_{RB} for $\delta_{\text{DEIM}} = 10^{-l}$, $l = 2, 3, 4, 5, 6$.



(a) GRB. (b) aLSRB ($\mathbf{X}_h^0$). (c) aLSRB ($\mathbf{P}_{\mathbf{X}_h^0}$).

Figure 3.8 – Computational times (seconds) vs δ_{RB} for $\delta_{\text{DEIM}} = 10^{-l}$, $l = 2, 3, 4, 5, 6$.



(a) GRB.   (b) aLSRB ($\mathbf{X}_h^0$).   (c) aLSRB ($\mathbf{P}_{\mathbf{X}_h^0}$).

Figure 3.9 – Building time (seconds) for $\mathbf{A}_N(\boldsymbol{\mu})$ vs δ_{RB} for $\delta_{\text{DEIM}} = 10^{-l}$, $l = 2, 3, 4, 5, 6$.

Table 3.3 – Computational time (seconds) required by the RB methods to compute a solution satisfying a target accuracy.

Accuracy	**G-RB**	**aLS-RB ($\mathbf{X}_h^0$)**	**aLS-RB ($\mathbf{P}_X^0$)**
1e-01	3.73	3.74	3.74
1e-02	3.93	3.74	3.74
1e-03	3.99	3.76	3.77
1e-04	x	4.14	3.91

3.5 MSRB preconditioners for the Stokes equations

After having introduced the RB approximation of the parametrized Stokes equations and having conceived and developed a new aLSRB method, we focus now on the construction of a MSRB preconditioner for the same problem.

Similarly to the elliptic case, we employ a multiplicative combination of an iteration dependent RB coarse operator $\mathbf{Q}_{N_k}(\boldsymbol{\mu})$ and a fine grid preconditioner $\mathbf{P}(\boldsymbol{\mu})$, hence exploiting the structure (2.10):

$$\mathbf{Q}_{\text{MSRB},k}(\boldsymbol{\mu}) = \mathbf{P}^{-1}(\boldsymbol{\mu}) + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\Big(\mathbf{I}_{N_h} - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big). \tag{3.53}$$

In the following, we will focus on the FGMRES iterative method, even though a similar construction can be done for the Richardson method. Furthermore, we devise a MSRB preconditioning strategy with a RB coarse operator which is constructed by relying on either a G-RB or aLSRB formulation; subsequently, to set up the MSRB preconditioner in a fairly general way, we consider a general PGRB formulation to build $\mathbf{Q}_{N_k}(\boldsymbol{\mu})$. To this aim, we recall that at iteration k of the FGMRES method we aim at computing a RB approximation $\mathbf{y}_{N_k}(\boldsymbol{\mu})$ of the solution $\mathbf{y}_k(\boldsymbol{\mu})$ of the following problem

$$\mathbf{A}_h(\boldsymbol{\mu})\mathbf{y}_k(\boldsymbol{\mu}) = \Big(\mathbf{I}_{N_h} - \mathbf{A}_h(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{v}_k(\boldsymbol{\mu}). \tag{3.54}$$

To this aim, we introduce the matrices $\mathbf{V}_k = [\boldsymbol{\xi}_1^k | \dots | \boldsymbol{\xi}_N^k] \in \mathbb{R}^{N_h \times N_k}$, $k = 1, 2, \dots$, such that the basis $\{\boldsymbol{\xi}_i^k\}_i^{N_k}$ is tailored to provide $\mathbf{V}_k\mathbf{y}_{N_k}(\boldsymbol{\mu}) \approx \mathbf{y}_k(\boldsymbol{\mu})$. We remark that the RB coarse component for the MSRB preconditioner is obtained, similarly to (1.41), by enforcing the projection of the FE residual of (3.54) evaluated for the RB coarse operator $\mathbf{V}_k\mathbf{y}_{N_k}(\boldsymbol{\mu})$ onto a test space generated by the columns of a matrix $\mathbf{W}_k(\boldsymbol{\mu})$ to vanish, that is by requiring

$$(\mathbf{W}_k(\boldsymbol{\mu}))^T\Big(\mathbf{v}_{k+\frac{1}{2}}(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu})\mathbf{V}_k\mathbf{y}_{N_k}(\boldsymbol{\mu})\Big) = 0. \tag{3.55}$$

In general, $\mathbf{W}_k(\boldsymbol{\mu})$ depends on both k and $\boldsymbol{\mu}$; if $\mathbf{W}_k(\boldsymbol{\mu}) \neq \mathbf{V}_k$, we build a PGRB coarse operator; otherwise, the choice $\mathbf{W}_k(\boldsymbol{\mu}) = \mathbf{V}_k$ leads to a G-RB coarse operator. This procedure leads to the following RB problem, to be solved at iteration $k = 1, 2, \dots,$ for any $\boldsymbol{\mu}$

$$(\mathbf{W}_k(\boldsymbol{\mu}))^T\mathbf{A}(\boldsymbol{\mu})\mathbf{V}_k\mathbf{y}_{N_k}(\boldsymbol{\mu}) = (\mathbf{W}_k(\boldsymbol{\mu}))^T\Big(\mathbf{I}_{N_h} - \mathbf{A}(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{v}_k(\boldsymbol{\mu}), \tag{3.56}$$

whose solution $\mathbf{y}_{N_k}(\boldsymbol{\mu}) \in \mathbb{R}^{N_k}$ is the RB approximation to the solution $\mathbf{y}_k(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ of (3.54) where $\mathbf{v}_k(\boldsymbol{\mu})$ is the k-th Krylov basis generated by the FGMRES method.

Accordingly with the construction in Section 3.2, the RB matrices $\mathbf{A}_{N_k}(\boldsymbol{\mu}) \in \mathbb{R}^{N_k \times N_k}$, $k = 1, 2, \dots$ are built as

$$\mathbf{A}_{N_k}(\boldsymbol{\mu}) = (\mathbf{W}_k(\boldsymbol{\mu}))^T\mathbf{A}(\boldsymbol{\mu})\mathbf{V}_k. \tag{3.57}$$

The FE representation $\mathbf{V}_k\mathbf{y}_{N_k}(\boldsymbol{\mu})$ of the RB approximation is then recovered as in equation (2.39)

$$\mathbf{V}_k\mathbf{y}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k(\mathbf{A}_{N_k}(\boldsymbol{\mu}))^{-1}(\mathbf{W}_k(\boldsymbol{\mu}))^T\Big(\mathbf{I}_{N_h} - \mathbf{A}(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{v}_k(\boldsymbol{\mu}),$$

from which we set the coarse operator as $\mathbf{Q}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k(\mathbf{A}_{N_k}(\boldsymbol{\mu}))^{-1}(\mathbf{W}_k(\boldsymbol{\mu}))^T$.

In the case of the parametrized Stokes equations, the solution of equation (3.54) is made of both velocity and pressure components, that is, $\mathbf{y}_k(\boldsymbol{\mu}) = [\mathbf{y}_{u,k}(\boldsymbol{\mu}),\, \mathbf{y}_{p,k}(\boldsymbol{\mu})]^T$, $k = 1, \dots$. Consequently, we build the RB spaces for these two variables separately by setting

$$\mathbf{V}_{uk} = POD\Big(\mathbf{S}_{\vec{u}}^{(k)}, \mathbf{X}_u, \delta_{RB,k}\Big), \tag{3.58}$$

$$\mathbf{V}_{pk} = POD\Big(\mathbf{S}_{p}^{(k)}, \mathbf{X}_p, \delta_{RB,k}\Big), \tag{3.59}$$

where $\mathbf{S}_{\vec{u}}^{(k)} = [\mathbf{y}_{u,k}(\boldsymbol{\mu}_1), \dots, \mathbf{y}_{u,k}(\boldsymbol{\mu}_{n_s})] \in \mathbb{R}^{N_h^u \times n_s}$, $\mathbf{S}_{p}^{(k)} = [\mathbf{y}_{p,k}(\boldsymbol{\mu}_1), \dots, \mathbf{y}_{p,k}(\boldsymbol{\mu}_{n_s})] \in \mathbb{R}^{N_h^p \times n_s}$ and $\delta_{RB,k} > 0$ is a prescribed tolerance (possibly depending on k). Here $\{\mathbf{y}_{u,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ and $\{\mathbf{y}_{p,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ are error snapshots for the velocity and the pressure for properly chosen instances of the parameters. Notice that POD on velocities $\{\mathbf{y}_{u,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$, $k = 1, \dots$ is performed with respect to the scalar product induced by the norm matrix $\mathbf{X}_u$. On the other hand, POD on pressures $\{\mathbf{y}_{p,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ is performed with respect to the scalar product induced by the norm matrix $\mathbf{X}_p$. Finally, the matrix $\mathbf{V}_k$ has the following form

$$\mathbf{V}_k = \begin{bmatrix} \mathbf{V}_{uk} & 0 \\ 0 & \mathbf{V}_{pk} \end{bmatrix}. \tag{3.60}$$

Remark 3.5.1. *An inf-sup condition similar to* (3.16) *must hold in order to guarantee the nonsingularity of the matrices* $\mathbf{A}_{N_k}(\boldsymbol{\mu})$ *for* $k = 1, 2, \ldots$, *that is, for any* $k = 1, 2, \ldots$ *there must exist* $\beta_{N_k}^{min} > 0$ *such that*

$$\beta_{N_k}(\boldsymbol{\mu}) = \inf_{\mathbf{z}_N \in \mathbb{R}^N} \sup_{\mathbf{w}_N \in \mathbb{R}^N} \frac{\mathbf{w}_N^T \mathbf{A}_{N_k}(\boldsymbol{\mu}) \mathbf{z}_N}{\|\mathbf{V}\mathbf{z}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})} \|\mathbf{W}_k(\boldsymbol{\mu})\mathbf{w}_N\|_{\mathbf{X}_h(\boldsymbol{\mu})}} \geq \beta_{N_k}^{min} \qquad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{3.61}$$

Remark 3.5.2. *Instead of providing the tolerances* $\delta_{RB,k}$, *we could prescribe the dimensions* N_k^u *and* N_k^p *of the RB spaces for the velocity and the pressure, respectively, at each iteration.*

In the following we devise two alternative techniques to build a well-posed RB coarse operator, according to two different choices of $\mathbf{W}_k(\boldsymbol{\mu})$, $k = 1, 2 \ldots$ which reflect the choice between a G-RB or an algebraic LSRB method discussed above.

3.5.1 MSRB preconditioners with enriched G-RB coarse operators

A G-RB approximation to build the $k-$th coarse operator is obtained by choosing $\mathbf{W}_k(\boldsymbol{\mu}) = \mathbf{V}_k$, $k = 1, 2, \ldots$. However, the resulting RB approximation is not guaranteed to fulfill (3.61), similarly to what happens for the standard G-RB approximation for the Stokes problem. Consequently, we consider an enriched velocity space formulation, where the velocity space spanned by the columns of $\mathbf{V}_{uk}$ is augmented by a set of N_k^s enriching basis functions. Given the pressure snapshots $\{\mathbf{y}_{p,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$, we build the *pressure supremizing snapshots* $\{\mathbf{y}_{t,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ by solving the following problems

$$\mathbf{X}_u(\boldsymbol{\mu})\mathbf{y}_{t,k}(\boldsymbol{\mu}_i) = \mathbf{B}^T(\boldsymbol{\mu}_i)\mathbf{y}_{p,k}(\boldsymbol{\mu}_i) \qquad i = 1, \ldots, n_s. \tag{3.62}$$

Next, we run POD on the set of pressure supremizing snapshots $\{\mathbf{y}_{t,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ and obtain $\mathbf{V}_{sk} \in \mathbb{R}^{N_h \times N_k^s}$ as

$$\mathbf{V}_{sk} = \mathrm{POD}\Big(\mathbf{S}_{\vec{t}}^{(k)}, \mathbf{X}_u, \delta_{RB,k}^s\Big),$$

with $\mathbf{S}_{\vec{t}}^{(k)} = [\mathbf{y}_{t,k}(\boldsymbol{\mu}_1), \ldots, \mathbf{y}_{t,k}(\boldsymbol{\mu}_{n_s})] \in \mathbb{R}^{N_h^u \times n_s}$ and $\delta_{RB,k}^s$ a prescribed tolerance. The columns of $\mathbf{V}_{sk}$ form a N_k^s-dimensional space employed to augment the velocity space after a proper Gram-Schmidt procedure:

$$\mathbf{V}_{uk} = \text{G-S}([\mathbf{V}_{uk}, \mathbf{V}_{sk}], \mathbf{X}_u). \tag{3.63}$$

By using (3.63) and setting $\mathbf{W}_k(\boldsymbol{\mu}) = \mathbf{V}_k$, $k = 1, \dots$, in (3.57), we obtain a well-posed G-RB coarse operator. Notice that a velocity enrichment is required for every coarse operator, leading to solve n_s additional problems of the form of (3.62) for each coarse operator $\mathbf{Q}_{N_k}(\boldsymbol{\mu})$, $k = 1, 2, \dots$ which has to be built, leading to a couple of RB spaces which proves to be numerically stable, even though a rigorous stability result cannot be proven, see e.g. [Dal Santo et al., 2017b].

3.5.2 MSRB preconditioners with aLSRB coarse operators

Compared to the approximate enrichment of the velocity space described in Section 3.5.1, the aLSRB method features a smaller dimension of the RB spaces (a lower number of RB functions), since in this case the velocity space is not augmented. This yields a remarkable advantage when the RB coarse operators and the inverse matrices of $\mathbf{A}_{N_k}(\boldsymbol{\mu})$, $k = 1, 2, \dots$, are constructed for a new parameter. Furthermore, the resulting RB formulation is automatically *inf-sup* stable, i.e. (3.61) is fulfilled. To build a LSRB approximation, we take advantage of the matrix $\mathbf{P}_X$ verifying Assumption 3.3.1. The aLSRB coarse operator is constructed by taking $\mathbf{W}_k(\boldsymbol{\mu}) = \mathbf{P}_X^{-1}\mathbf{A}(\boldsymbol{\mu})\mathbf{V}_k$ in (3.57), leading to the following definition

$$\mathbf{A}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k^T(\mathbf{A}(\boldsymbol{\mu}))^T\mathbf{P}_X^{-1}\mathbf{A}(\boldsymbol{\mu})\mathbf{V}_k, \qquad k = 1, \dots. \tag{3.64}$$

The RB problem with $\mathbf{A}_{N_k}(\boldsymbol{\mu})$ so chosen yields a naturally well-posed problem, verifying (3.61) with $\beta_{N_k}^{min} = \beta_h^{min} c/C$.

3.5.3 Nonsingularity of the preconditioner

When a G-RB approximation is employed to build the coarse operators, as in the case where an augmented velocity space is used, the MSRB preconditioner operator $\mathbf{Q}_{\text{MSRB},k}(\boldsymbol{\mu})$ is invertible, with proper assumptions on $\mathbf{P}(\boldsymbol{\mu})$ and the basis $\mathbf{V}_k$, see Section 2.2.2. In the following we extend these results, showing that $\mathbf{Q}_{\text{MSRB},k}(\boldsymbol{\mu})$ is invertible when a more general PGRB approach is used to build the RB coarse operators.

Let $W_1 = span\{\mathbf{w}_j^1\}_{j=1}^M$ and $W_2 = span\{\mathbf{w}_j^2\}_{j=1}^M \subset \mathbb{R}^{N_h}$ be two subspaces such that $dim(W_1) = dim(W_2) = M$. We denote by $W_1^\perp$ and $W_2^\perp$ the orthogonal complement of W_1 and W_2, respectively, and by $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{N_h \times M}$ the matrices of basis vectors such that $\mathbf{W}_1 = [\mathbf{w}_1^1, \dots, \mathbf{w}_M^1]$, $\mathbf{W}_2 = [\mathbf{w}_1^2, \dots, \mathbf{w}_M^2]$. Moreover, given a subspace $W \subset \mathbb{R}^{N_h}$

and a nonsingular matrix $\mathbf{B} \in \mathbb{R}^{N_h \times N_h}$, we define the following spaces

$$
\begin{aligned}
\mathbf{B}W &= \Big\{ \mathbf{x} \in \mathbb{R}^{N_h} : \ \mathbf{B}^{-1}\mathbf{x} \in W \quad \Big\} = \Big\{ \mathbf{x} \in \mathbb{R}^{N_h} : \ \ \mathbf{x} = \mathbf{B}\mathbf{z}, \, \mathbf{z} \in W \Big\}, \\
\mathbf{B}W^{\perp} &= \Big\{ \mathbf{x} \in \mathbb{R}^{N_h} : \ \mathbf{B}^{-1}\mathbf{x} \in W^{\perp} \Big\} = \Big\{ \mathbf{x} \in \mathbb{R}^{N_h} : \ \ \mathbf{x} = \mathbf{B}\mathbf{z}, \, \mathbf{z} \in W^{\perp} \Big\}.
\end{aligned}
$$

We remark that $\mathbb{R}^{N_h} = \mathbf{B}W \oplus \mathbf{B}W^{\perp}$, because of the nonsingularity of $\mathbf{B}$.

Lemma 3.5.1. *Let W_1 and W_2 be two M-dimensional subspaces of $\mathbb{R}^{N_h}$, $\{\mathbf{w}_j^1\}_{j=1}^M$ and $\{\mathbf{w}_j^2\}_{j=1}^M$ their basis and $\mathbf{W}_1 = [\mathbf{w}_1^1, \dots, \mathbf{w}_M^1] \in \mathbb{R}^{N_h \times M}$, $\mathbf{W}_2 = [\mathbf{w}_1^2, \dots, \mathbf{w}_M^2] \in \mathbb{R}^{N_h \times M}$. Moreover, let $\mathbf{B}$ be a nonsingular $N_h \times N_h$ matrix and assume that $\mathbf{W}_2^T \mathbf{B} \mathbf{W}_1$ is nonsingular. Then the following implication holds:*

$$
\mathbf{x} \in \mathbf{B}W_1 \quad \text{and} \quad \mathbf{W}_2^T \mathbf{x} = \mathbf{0} \quad \Rightarrow \quad \mathbf{x} = \mathbf{0}.
$$

Proof. We take $\mathbf{x} \in \mathbf{B}W_1$ such that $\mathbf{W}_2^T\mathbf{x} = \mathbf{0}$ and show that it must be $\mathbf{x} = \mathbf{0}$. By definition of $\mathbf{B}W_1$, $\mathbf{B}^{-1}\mathbf{x} = \mathbf{W}_1\mathbf{z}_M$ for some $\mathbf{z}_M \in \mathbb{R}^M$. Thanks to the nonsingularity of $\mathbf{B}$, we obtain

$$
\mathbf{0} = \mathbf{W}_2^T\mathbf{x} = \mathbf{W}_2^T\mathbf{B}\mathbf{B}^{-1}\mathbf{x} = \mathbf{W}_2^T\mathbf{B}\mathbf{W}_1\mathbf{z}_M,
$$

which implies $\mathbf{z}_M = \mathbf{0}$, due to the nonsingularity of $\mathbf{W}_2^T\mathbf{B}\mathbf{W}_1 \in \mathbb{R}^{M \times M}$. Finally, we have

$$
\mathbf{0} = \mathbf{W}_1\mathbf{z}_M = \mathbf{B}^{-1}\mathbf{x},
$$

which, thanks to the nonsingularity of $\mathbf{B}$, ends the proof. □

In the following we employ Lemma 3.5.1 by taking $\mathbf{W}_1 = \mathbf{V}_k$, $\mathbf{W}_2 = \mathbf{W}_k(\boldsymbol{\mu})$, $\mathbf{B} = \mathbf{P}(\boldsymbol{\mu})$ in order to prove that $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ is nonsingular. To this aim, we define

$$
V_{N_k}^{\mathbf{P}/\!/} = \Big\{ \mathbf{x} \in \mathbb{R}^{N_h} : \quad \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x} \in V_{N_k} \Big\}, \quad V_{N_k}^{\mathbf{P}\perp} = \Big\{ \mathbf{x} \in \mathbb{R}^{N_h} : \quad \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x} \in V_{N_k}^{\perp} \Big\}.
$$

Theorem 3.5.1. *For any $\boldsymbol{\mu} \in \mathcal{D}$, assume that $\mathbf{P}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ is a nonsingular matrix such that the matrix $(\mathbf{W}_k(\boldsymbol{\mu}))^T\mathbf{P}(\boldsymbol{\mu})\mathbf{V}_k$ is nonsingular. Then the matrix $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ is nonsingular.*

Proof. The proof is similar to the one outlined in Section 2.2.2. Given $\mathbf{x} = \mathbf{x}_{/\!/} + \mathbf{x}_{\perp}$, where $\mathbf{x}_{/\!/} \in V_{N_k}^{\mathbf{P}/\!/}$, $\mathbf{x}_{\perp} \in V_{N_k}^{\mathbf{P}\perp}$, such that $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{x} = \mathbf{0}$, then it must be $\mathbf{x} = \mathbf{0}$. Then we have

$$
\begin{aligned}
\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{x}_{/\!/} &= \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{/\!/} + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\Big(\mathbf{I}_{N_h} - \mathbf{A}(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{x}_{/\!/} \\
&= \mathbf{V}_k\mathbf{z}_N^{\boldsymbol{\mu}} + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{x}_{/\!/} - \mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{A}(\boldsymbol{\mu})\mathbf{V}_k\mathbf{z}_N^{\boldsymbol{\mu}} = \mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{x}_{/\!/},
\end{aligned}
$$

where $\mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{/\!/} = \mathbf{V}_k\mathbf{z}_N^{\boldsymbol{\mu}}$ for some $\mathbf{z}_{N_k}^{\boldsymbol{\mu}} \in \mathbb{R}^{N_k}$. Then

$$\begin{aligned}\mathbf{0} &= \mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{x} = \mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{x}_{/\!/} + \mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})\mathbf{x}_{\perp} \\ &= \mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{x}_{/\!/} + \mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{\perp} + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\Big(\mathbf{I}_{N_h} - \mathbf{A}(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\mathbf{x}_{\perp}\end{aligned}$$

which leads to

$$\mathbf{Q}_{N_k}(\boldsymbol{\mu})\Big(\mathbf{x}_{/\!/} + \mathbf{x}_{\perp} + \mathbf{A}(\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{\perp}\Big) = -\mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{\perp}. \tag{3.65}$$

The left hand side is an element of V_{N_k}, the right hand side is an element of $V_{N_k}^{\perp}$, therefore the only way for them to be equal is when they are both zero. Being $\mathbf{P}^{-1}(\boldsymbol{\mu})\mathbf{x}_{\perp} = \mathbf{0}$, implies $\mathbf{x}_{\perp} = \mathbf{0}$ thanks to the nonsingularity of $\mathbf{P}(\boldsymbol{\mu})$, leading to

$$\mathbf{0} = \mathbf{Q}_{N_k}(\boldsymbol{\mu})\mathbf{x}_{/\!/} = \mathbf{V}_k\mathbf{A}_{N_k}^{-1}(\boldsymbol{\mu})(\mathbf{W}_k(\boldsymbol{\mu}))^T\mathbf{x}_{/\!/} \tag{3.66}$$

which, thanks to linear independence of the columns of $\mathbf{V}_k$ and the non singularity of $\mathbf{A}_{N_k}(\boldsymbol{\mu})$ yields

$$(\mathbf{W}_k(\boldsymbol{\mu}))^T\mathbf{x}_{/\!/} = \mathbf{0}.$$

Finally, by applying Lemma 3.5.1 with $W_1 = V_{N_k}$, $\mathbf{W}_1 = \mathbf{V}_k$, $\mathbf{W}_2 = \mathbf{W}_k(\boldsymbol{\mu})$ and $\mathbf{B} = \mathbf{P}(\boldsymbol{\mu})$, we obtain that $\mathbf{x}_{/\!/} = \mathbf{0}$. □

As in the Galerkin-RB case, thanks to the nonsingularity of $\mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ invertible, we can define the MSRB preconditioner as

$$\mathbf{P}_{\mathrm{MSRB},k}(\boldsymbol{\mu}) = \mathbf{Q}_{\mathrm{MSRB},k}^{-1}(\boldsymbol{\mu}).$$

3.5.4 Algorithmic procedures

In this section we detail the procedures required to build and use the MSRB preconditioner for the Stokes problem. As in the elliptic case, we split the computation in an offline and an online phase.

Offline phase

First, the RB spaces as in (3.58)-(3.59) are constructed. To this goal, we solve the FE problem (3.6) for n_s instances of $\boldsymbol{\mu}$ to build the snapshots for velocity $\{\mathbf{u}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ and pressure $\{\mathbf{p}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$, and set

$$\mathbf{y}_{u0}^{\boldsymbol{\mu}_i} = \mathbf{u}^{\boldsymbol{\mu}_i}, \quad \mathbf{y}_{p0}^{\boldsymbol{\mu}_i} = \mathbf{p}^{\boldsymbol{\mu}_i}, \qquad i = 1, \ldots, n_s.$$

These snapshots are used to build the spaces $\mathbf{V}^u_{N_0^u}$ and $\mathbf{V}^p_{N_0^p}$, respectively, which in turn are used to provide the initial guess to the FGMRES algorithm. As a matter of fact they are the usual spaces used for the standard RB approximation, that is, $\mathbf{V}^u_{N_0^u} = \mathbf{V}_{N_u}$ and $\mathbf{V}^p_{N_0^p} = \mathbf{V}_{N_p}$. For each new RB space $\mathbf{V}_k$, $k = 1, 2, \ldots$, the new snapshots $\{\mathbf{y}_{u,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ and $\{\mathbf{y}_{p,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$, $k = 1, 2, \ldots$, solution of (3.54) for particular instances of $\boldsymbol{\mu}$, are computed taking advantage of (2.42). An (approximated) affine dependence of $\mathbf{A}(\boldsymbol{\mu})$ can also be exploited: if (1.44) is verified, the RB matrix $\mathbf{A}_{N_k}(\boldsymbol{\mu})$ can be constructed as

$$\mathbf{A}_{N_k}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) \mathbf{V}_k^T \mathbf{A}_h^q \mathbf{V}_k = \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) \mathbf{A}^q_{N_k} \tag{3.67}$$

in the G-RB case and as

$$\begin{aligned} \mathbf{A}_{N_k}(\boldsymbol{\mu}) &= \sum_{q_1,q_2=1}^{Q_a} \Theta_a^{q_1}(\boldsymbol{\mu}) \Theta_a^{q_2}(\boldsymbol{\mu}) \mathbf{V}_k^T (\mathbf{A}_h^{q_1})^T \mathbf{P}_X^{-1} \mathbf{A}_h^{q_2} \mathbf{V}_k \\ &= \sum_{q_1,q_2=1}^{Q_a} \Theta_a^{q_1}(\boldsymbol{\mu}) \Theta_a^{q_2}(\boldsymbol{\mu}) \mathbf{A}^{q_1,q_2}_{N_k}. \end{aligned} \tag{3.68}$$

in the aLSRB case. The matrices $\mathbf{A}^q_{N_k}$, $q = 1, \ldots, Q_a$, $\mathbf{A}^{q_1,q_2}_{N_k} \in \mathbb{R}^{N \times N}$, $q_1, q_2 = 1, \ldots, Q_a$, depending on the chosen RB approximation, can be precomputed and stored once the RB spaces $\mathbf{V}_k$ are constructed. Then, given a new value $\boldsymbol{\mu}$ of parameter, only the sum in (3.67) or (3.68) must be carried out to build $\mathbf{A}_{N_k}(\boldsymbol{\mu})$. If only an approximated affine decomposition is available (e.g. reconstructed with MDEIM), then the left equalities in (3.67)-(3.68) hold approximately. Furthermore, we remark that the 2×2 RB matrices $\mathbf{A}_{N_k}(\boldsymbol{\mu})$ read

$$\mathbf{A}_{N_k}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{D}_{N_k^u}(\boldsymbol{\mu}) & \mathbf{B}^T_{N_k}(\boldsymbol{\mu}) \\ \mathbf{B}_{N_k}(\boldsymbol{\mu}) & \mathbf{C}_{N_k^p}(\boldsymbol{\mu}) \end{bmatrix}. \tag{3.69}$$

In the G-RB coarse operator case this is a saddle-point matrix, since $\mathbf{C}_{N_k^p}(\boldsymbol{\mu}) = O$, whereas, in the aLSRB one, $\mathbf{C}_{N_k^p}(\boldsymbol{\mu})$ is a symmetric and positive definite matrix yielding the symmetry and the positive definiteness of $\mathbf{A}_{N_k}(\boldsymbol{\mu})$.

The offline construction of the MSRB preconditioner is outlined in Algorithm 9 for the G-RB case and in Algorithm 10 for the aLSRB case. We provide a set of sampling parameters $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$, a final tolerance ε_r and the tolerances to construct each RB space $\{\delta_{RB,k}\}_k$; then, at first we compute an affine decomposition $\{\mathbf{A}_h^q\}_{q=1}^{Q_a}$ of the matrix $\mathbf{A}(\boldsymbol{\mu})$ with M-DEIM algorithm [Negri et al., 2015a] (step 2), and we construct the snapshots required to build the first space (step 3). Then, we iteratively build the necessary RB spaces through POD (steps 5-8) and the affine RB decomposition matrices $\{\mathbf{A}^{q_1,q_2}_{N_k}\}_{q_1,q_2=1}^{Q_a}$ (step 9). The final number of RB spaces constructed is L. In the G-RB case, the construction of the snapshots is more demanding, since it requires to build also the supremizer snapshots and an additional POD for each RB space, which also leads to RB coarse components of larger dimension due to the enrichment of the velocity space. However, the number of affine structures to be computed and stored is Q_a in the G-RB case, but increases to Q_a^2 in the aLSRB case. Hence, taking into account all these factors and depending on the application at hand, one should decide between a G-RB or aLSRB approach and prefer the latter if the number of affine terms is not excessive.

Algorithm 9 MSRB Preconditioner with G-RB coarse operator - Offline phase

```
procedure MSRB-PRECONDITIONER-G-OFFLINE($\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$, $\varepsilon_r$, $\{\delta_{RB,k}\}_k$, $\delta_{\mathrm{MDEIM}}$)
    Compute an (approximated) affine decomposition of $\mathbf{A}(\boldsymbol{\mu})$
    Compute the FE solutions $\{\mathbf{z}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ and pressure supremizers $\{\mathbf{t}_p(\mathbf{p}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i)\}_{i=1}^{n_s}$
    Set $\mathbf{S}_{\vec{u}}^{(0)} = [\mathbf{u}^{\boldsymbol{\mu}_1}, \dots, \mathbf{u}^{\boldsymbol{\mu}_{n_s}}]$, $\mathbf{S}_p^{(0)} = [\mathbf{p}^{\boldsymbol{\mu}_1}, \dots, \mathbf{p}^{\boldsymbol{\mu}_{n_s}}]$, $\mathbf{S}_{\vec{t}}^{(0)} = [\mathbf{t}_p(\boldsymbol{\mu}_1), \dots, \mathbf{t}_p(\boldsymbol{\mu}_{n_s})]$ and $k = 0$
    while $\prod_k \delta_{RB,k} > \varepsilon_r$ do
        $\mathbf{V}_{uk} = \mathrm{POD}(\mathbf{S}_{\vec{u}}^{(k)}, \mathbf{X}_u, \delta_{RB,k})$
        $\mathbf{V}_{pk} = \mathrm{POD}(\mathbf{S}_p^{(k)}, \mathbf{X}_p, \delta_{RB,k})$
        $\mathbf{V}_{sk} = \mathrm{POD}(\mathbf{S}_{\vec{t}}^{(k)}, \mathbf{X}_u, \frac{\delta_{RB,k}}{10})$
        $\mathbf{V}_{uk} = \mathrm{G\text{-}S}(\mathbf{V}_{uk}, \mathbf{V}_{sk}, \mathbf{X}_u)$
        Build RB affine matrices $\{\mathbf{A}_{N_k}^q\}_{q=1}^{Q_a}$
        Compute new snapshots $\{\mathbf{y}_{u,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ and $\{\mathbf{y}_{p,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ with (2.42)
        Compute new supremizer snapshots $\{\mathbf{y}_{t,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ with (3.62)
        $\mathbf{S}_{\vec{u}}^{(k+1)} = [\mathbf{y}_{u,k+1}(\boldsymbol{\mu}_1), \dots, \mathbf{y}_{u,k+1}(\boldsymbol{\mu}_{n_s})]$,
        $\mathbf{S}_p^{(k+1)} = [\mathbf{y}_{p,k+1}(\boldsymbol{\mu}_1), \dots, \mathbf{y}_{p,k+1}(\boldsymbol{\mu}_{n_s})]$
        $\mathbf{S}_{\vec{t}}^{(k+1)} = [\mathbf{y}_{t,k+1}(\boldsymbol{\mu}_1), \dots, \mathbf{y}_{t,k+1}(\boldsymbol{\mu}_{n_s})]$
        $k = k + 1$
    end while
end procedure
```

Notice that instead of providing a set of tolerances $\{\delta_{RB,k}\}_k$, we can also provide a set of dimensions $\{N_k\}_k$. Following the elliptic case, two strategies have been employed to build in practice the RB coarse operators:

- *fixed space accuracy*: we build each RB space prescribing the same tolerance δ_{RB}, i.e. $\delta_{RB,k} = \delta_{RB}$ for any k. If a G-RB method approach is employed, then the tolerance provided to POD for the construction of the enriching basis functions $\mathbf{V}_{sk}$ is $\delta_{RB,k}/10$, which empirically results in a well-posed G-RB approximation;

- *fixed space dimension*: the dimensions $\{N_k^u\}_k$, $\{N_k^p\}_k$ and, eventually, $\{N_k^s\}_k$ of the RB spaces are set to a fixed value N, that is $N_k^u = N_k^p = N (= N_k^s)$ for any k.

Algorithm 10 MSRB Preconditioner with aLSRB coarse operator - Offline phase

```
1: procedure MSRB-PRECONDITIONER-ALS-OFFLINE($\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}, \varepsilon_r, \{\delta_{RB,k}\}_k, \delta_{\text{MDEIM}}$)
2:     Compute an affine approximation $\{\mathbf{A}_h^q\}_{q=1}^{Q_a}$
3:     Compute the FE solutions $\{\mathbf{z}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$
4:     Set $\mathbf{S}_{\vec{u}}^{(0)} = [\mathbf{u}(\boldsymbol{\mu}_1), \dots, \mathbf{u}(\boldsymbol{\mu}_{n_s})]$, $\mathbf{S}_p^{(0)} = [\mathbf{p}(\boldsymbol{\mu}_1), \dots, \mathbf{p}(\boldsymbol{\mu}_{n_s})]$ and $k = 0$
5:     while $\prod_k \delta_{RB,k} > \varepsilon_r$ do
6:         $\mathbf{V}_{uk} = POD(\mathbf{S}_{\vec{u}}^{(k)}, \delta_{RB,k})$
7:         $\mathbf{V}_{pk} = POD(\mathbf{S}_p^{(k)}, \delta_{RB,k})$
8:         Build RB affine matrices $\{\mathbf{A}_{N_k}^{q_1,q_2}\}_{q_1,q_2=1}^{Q_a}$
9:         Compute new snapshots $\{\mathbf{y}_{u,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ and $\{\mathbf{y}_{p,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ with (2.42)
10:        $\mathbf{S}_{\vec{u}}^{(k+1)} = [\mathbf{y}_{u,k+1}(\boldsymbol{\mu}_1), \dots, \mathbf{y}_{u,k+1}(\boldsymbol{\mu}_{n_s})]$,
11:        $\mathbf{S}_p^{(k+1)} = [\mathbf{y}_{p,k+1}(\boldsymbol{\mu}_1), \dots, \mathbf{y}_{p,k+1}(\boldsymbol{\mu}_{n_s})]$
12:        $k = k + 1$
13:    end while
14: end procedure
```

Sequential RB coarse operator construction

The offline phase, and especially the computation of the set of snapshots $\{\mathbf{z}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ in step 3 of Algorithm 9 and 10, can be particularly expensive. In order to speed up the process, we can alternatively opt for a sequential construction of the RB coarse operators. With this aim, we partition the snapshot set by introducing M subsets $\mathcal{Z}_m$, $m = 1, \dots, M$, of $\{\mathbf{z}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$, of dimension n_s^m, respectively, and such that

$$\{\mathbf{z}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s} = \bigcup_{m=1}^{M} \mathcal{Z}_m, \qquad n_s = \sum_{m=1}^{M} n_s^m, \qquad \mathcal{Z}_m = \{\mathbf{z}_h^{\boldsymbol{\mu}_i}\}_{1+i_{m-1}}^{i_m},$$

where $i_m = \sum\limits_{l=1}^{m} n_s^l$. Then, the k-th RB matrix $\mathbf{V}_k$ is built using $\bigcup_{m=1}^{k} \mathcal{Z}_m$ as snapshots set. We remark that there is not any correspondence between the choice of the number of RB spaces L and the number of snapshot partition M. Exploiting only part of the snapshots allows to use the MSRB preconditioner developed up to iteration k for the computation of the new snapshots $\mathcal{Z}_j$, $j > k$, which will be employed to construct the RB spaces $\mathbf{V}_j$, $j > k$. This technique yields a reduction of the overall time required by the snapshot computation, since the speed up provided by the MSRB preconditioner is sequentially used to build part of the snapshots. M and $\mathcal{Z}_m$, $m = 1, \dots, M$ are empirically chosen such that the accuracies obtained by the RB coarse operators do not change if compared with the ones obtained with the RB coarse operators built with the complete set of snapshot.

Online phase

In the online phase, we aim at computing the solutions of (3.6) for new instances of the parameter $\boldsymbol{\mu}$, which have not been considered during the offline phase. We thus need to compute the weights $\{\Theta_a^q(\mu)\}_{q=1}^{Q_a}$ of the affine decomposition of $\mathbf{A}(\boldsymbol{\mu})$, and build the coarse operators $\{\mathbf{Q}_{N_k}^{\mu}\}_k$. The assembly of $\mathbf{A}_{N_k}(\boldsymbol{\mu})$ is performed blockwise by relying on the (eventually approximated) affine decomposition of the matrices $\mathbf{D}(\boldsymbol{\mu})$ and $\mathbf{B}(\boldsymbol{\mu})$, for the aLSRB coarse operators similarly to what explained in Section 3.3.2. Finally, we apply FGMRES algorithm 3 with $\mathbf{M}_k^{-1}(\boldsymbol{\mu}) = \mathbf{Q}_{\mathrm{MSRB},k}(\boldsymbol{\mu})$ in the preconditioning step.

3.6 Numerical results

In this section we show numerical results where the proposed MSRB preconditioner, based on either a G-RB or an aLSRB method, is employed to solve Stokes equations in parametrized geometries. Parameter dependent domains are obtained by considering a map from a reference domain to the physical domain which can be provided either analytically (test case I) or by computing the solution of an additional FE problem (test case II), e.g. when a solid extension mesh moving technique is employed, see [Manzoni and Negri, 2017]. However, we highlight that the proposed strategy is applicable also to the case where physical parameters are considered.

We employ Taylor-Hood ($\mathcal{P}_2 - \mathcal{P}_1$) finite element spaces for velocity and pressure, respectively, as high-fidelity discretization, which are proven to provide an *inf-sup* stable FE problem. The lifting function $\vec{r}_{\vec{g}_D}(\boldsymbol{\mu})$ is computed as harmonic extension of the Dirichlet data $\vec{g}_D(\boldsymbol{\mu})$ in (3.1), which is chosen as a parabolic profile such that the flow rate at the inlet is equal to 1. An approximation of $\vec{r}_{\vec{g}_D}(\boldsymbol{\mu})$ is computed by employing the FE method, with second order polynomials basis functions. This leads to a parametrized linear system whose solution $\mathbf{r}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^u}$ is the approximated lifting functions computed with the preconditioned conjugate gradient (PCG) method, exploiting an Algebraic Multigrid (AMG) preconditioner from the ML package of Trilinos [Gee et al., 2006].

As fine component $\mathbf{P}(\boldsymbol{\mu})$ we employ the Pressure Mass Matrix (PMM) preconditioner defined as

$$\mathbf{P}(\boldsymbol{\mu}) = \mathbf{P}_{\mathbf{M}}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{D}(\boldsymbol{\mu}) & \mathbf{B}^T(\boldsymbol{\mu}) \\ 0 & -\frac{1}{\nu(\boldsymbol{\mu})}\mathbf{X}_p(\boldsymbol{\mu}) \end{bmatrix}, \tag{3.70}$$

where the Schur complement $\mathbf{S}(\boldsymbol{\mu})$ is approximated with the rescaled pressure mass matrix, that is $\widetilde{\mathbf{S}}(\boldsymbol{\mu}) = \frac{1}{\nu(\boldsymbol{\mu})}\mathbf{X}_p(\boldsymbol{\mu})$ (which is spectrally equivalent to $\mathbf{S}(\boldsymbol{\mu})$ at least for two-dimensional problems). The PMM preconditioner (3.70) allows to obtain extremely satisfactory results both in terms of optimality and scalability, see e.g. [Rehman et al., 2011] and results therein. Specifically, the application of $\mathbf{P}_{\mathbf{M}}(\boldsymbol{\mu})$ is detailed in Algorithm 11, where the application of $\mathbf{P}_{\mathbf{M}}^{-1}(\boldsymbol{\mu})$ to the $k-$th Krylov basis function $\mathbf{v}_k = [\mathbf{v}_{u,k},\, \mathbf{v}_{p,k}]^T$

(at step k of the Krylov method) is summarized. Steps 1 and 3 are solved inexactly by inner iterations up to a tolerance of 10^{-5} on the Euclidean norm of the residual rescaled with the Euclidean norm of the right hand side. An algebraic multigrid (AMG) preconditioner from the ML package of Trilinos [Gee et al., 2006] is employed for the inner iterations.

Algorithm 11 Computation of $\mathbf{P}_{\mathbf{M}}^{-1}(\boldsymbol{\mu})\mathbf{v}_k$

```
1: solve the pressure problem $-\frac{1}{\nu(\boldsymbol{\mu})}\mathbf{X}_p(\boldsymbol{\mu})\mathbf{z}_{p,k} = \mathbf{v}_{p,k}$ (solved inexactly by inner iterations);
2: update the velocity $\mathbf{v}_{u,k} = \mathbf{v}_{u,k} - \mathbf{B}^T(\boldsymbol{\mu})\mathbf{z}_{p,k}$;
3: solve the velocity problem $\mathbf{D}(\boldsymbol{\mu})\mathbf{z}_{u,k} = \mathbf{v}_{u,k}$ (solved inexactly by inner iterations).
```

In the following, we compare the results obtained with the MSRB preconditioner with the ones obtained by using only the PMM preconditioner $\mathbf{P}_{\mathbf{M}}(\boldsymbol{\mu})$.

3.6.1 Test case I: parametrized cylinder

The first test case concerns a Stokes flow in a three-dimensional cylinder whose shape varies according to a set of parameters. We introduce a reference domain

$$\Omega^0 = \{\vec{x} \in \mathbb{R}^3 : \; x_1^2 + x_2^2 < 0.25, \, x_3 \in (0, 5)\},$$

and obtain the computational domain $\Omega(\boldsymbol{\mu})$ as

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

where $\vec{d}(\boldsymbol{\mu})$ is an analytical displacement

$$\vec{d}(\boldsymbol{\mu}) = \begin{bmatrix} -x_1\mu_1 \exp\{-\frac{(x_3-2.5)^2}{\mu_2}\} \\ -x_2\mu_1 \exp\{-\frac{(x_3-2.5)^2}{\mu_2}\} \\ 0 \end{bmatrix}.$$

Here the parameter $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathcal{D} = (0, 0.3) \times (0.5, 1)$. The cylinder is narrowed in the central section by a factor $\mu_1/2$, whereas μ_2 determines how the narrowing effect propagates towards the inlet and outlet sections. An example of deformation is shown in Figure 3.10. Compared to the example considered for the aLSRB solver in Section 3.4, in this case the displacement is assigned analytically in whole the domain (instead of being an harmonic extension of a boundary condition) and the position of the largest narrowing section is fixed at $x_3 = 2.5$ (instead of being parameter dependent).

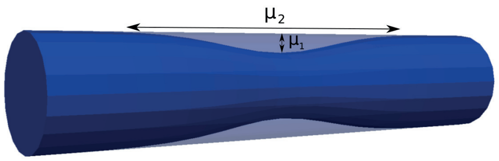
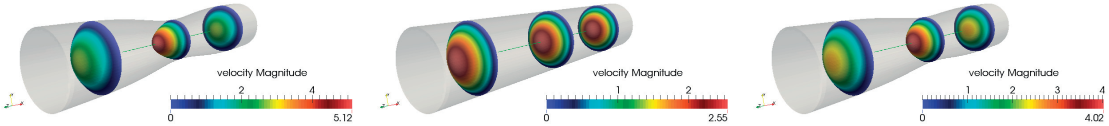


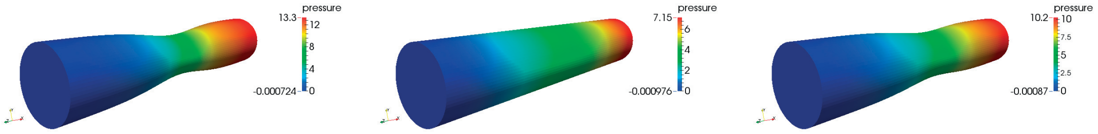
Figure 3.10 – Deformation of the domain for test case I.

Simulation setup

We show numerical results obtained for three different meshes, leading to a finite element problem with dimension $N_h = 52'152$, $320'338$, $1'568'223$, respectively, computed with $N_{\text{core}} = 36$, 180, 900 cores, thus distributing about 1800 dofs per CPU. The FE solution for different values of the parameter $\boldsymbol{\mu}$ is reported in Figure 3.11.



(a) Velocity $\boldsymbol{\mu} = (0.3, 1)$ (b) Velocity $\boldsymbol{\mu} = (0.0, 0.5)$ (c) Velocity $\boldsymbol{\mu} = (0.21, 0.85)$



(d) Pressure $\boldsymbol{\mu} = (0.3, 1)$ (e) Pressure $\boldsymbol{\mu} = (0.0, 0.5)$ (f) Pressure $\boldsymbol{\mu} = (0.21, 0.85)$

Figure 3.11 – Test case I, numerical solution for three values of $\boldsymbol{\mu}$ obtained with the MSRB preconditioning technique.

As RB coarse component, we show results for both the *fixed accuracy* and *fixed dimension* approaches in the following configurations:

- **GRB**: G-RB coarse operators;
- **aLSRB-$\mathbf{X_h^0}$**: aLSRB coarse operators where $\mathbf{P}_X = \mathbf{X}_h^0$, i.e. the matrix norm (3.12) on the reference domain;
- **aLSRB-$\mathbf{P_{X_h^0}}$**: aLSRB coarse operators where $\mathbf{P}_X = \mathbf{P}_{\mathbf{X}_h^0}$, where $\mathbf{P}_{\mathbf{X}_h^0}$ is a symmetric and positive definite preconditioner for $\mathbf{X}_h^0$ with a block structure $\mathbf{P}_{\mathbf{X}_h^0} = diag(\mathbf{P}_{\mathbf{X}_u^0}, \mathbf{P}_{\mathbf{X}_p^0})$, where $\mathbf{P}_{\mathbf{X}_u} \in \mathbb{R}^{N_h^u \times N_h^u}$ (resp. $\mathbf{P}_{\mathbf{X}_p} \in \mathbb{R}^{N_h^p \times N_h^p}$) is a symmetric and positive definite AMG preconditioner of $\mathbf{X}_u^0$ (resp. $\mathbf{X}_p^0$).

Table 3.4 – Test case I, MDEIM offline results, $\delta_{\text{MDEIM}} = 10^{-6}$.

N_h	Q_d	Q_b	$\mathbf{D}(\boldsymbol{\mu})$ offline time (s)	$\mathbf{B}(\boldsymbol{\mu})$ offline time (s)
52152	7	10	24.65	5.25
320338	6	10	37.29	8.11
1568223	6	10	54.37	11.71

For the offline phase, we take $n_s = 100$ snapshots for both the construction of the RB coarse operators and the MDEIM algorithm, which is employed to provide an affine approximation of the matrices $\mathbf{D}(\boldsymbol{\mu})$ and $\mathbf{B}(\boldsymbol{\mu})$. Specifically, for MDEIM we set $\delta_{\text{MDEIM}} = 10^{-6}$. Regarding the construction of the RB spaces, we take as final tolerance $\varepsilon_r = 10^{-9}$ for all the test cases. For the fixed accuracy approach we construct $L = 4$ RB spaces, yielding $\delta_{RB,k} = \delta_{RB} = 10^{-9/4} \approx 5.6 \cdot 10^{-3}$ for each k. For the fixed dimension approach, we take $N_k = 10$ for each k.

During the online phase, we test the proposed MSRB preconditioners with the three different RB coarse operators (**GRB**, **aLSRB-$\mathbf{X_h^0}$** and **aLSRB-$\mathbf{P_{X_h^0}}$**). We solve the FE linear system with the FGMRES method on 150 online parameters different from the ones employed during the offline phase to build the RB coarse operators. We use a stopping criterion based on the Euclidean norm of the residual, rescaled with the Euclidean norm of the right hand side, and require this quantity to be lower than 10^{-6}.

Numerical results

The computational time required to compute the approximate affine decomposition of the matrices $\mathbf{D}(\boldsymbol{\mu})$ and $\mathbf{B}(\boldsymbol{\mu})$ with the MDEIM algorithm and the number of basis functions Q_a are reported in Table 3.4. The number of required basis functions Q_a mainly depends on the parameter dependence of the PDE, consequently it does not vary with the FE dimension, and ranges from 6 to 10 with a tolerance $\delta_{\text{MDEIM}} = 10^{-6}$.

The results obtained with the MSRB preconditioner during the online phase, i.e. for new instances of the parameter, for the fixed accuracy approach with **GRB**, **aLSRB-$\mathbf{X_h^0}$** and **aLSRB-$\mathbf{P_{X_h^0}}$** are reported in Table 3.5, 3.6 and 3.7, respectively. For the fixed dimension approach, the results are reported in Table 3.8, 3.9 and 3.10, respectively. For each case, we report the number of RB coarse operators L and the total number of basis functions N_k for the space k, as the sum of the velocity, pressure and supremizer RB functions, this latter only if **GRB** is employed. We underline that the number of basis functions is larger in the **GRB** case, due to the velocity enrichment. Furthermore, the detailed results concerning the time required to compute the solution by employing the PMM preconditioner t_{PMM} and the MSRB preconditioner $t_{\text{MSRB}}^{\text{onl}}$, together with the corresponding iteration counts It_{PMM} and It^{onl}, are reported.

The number of iterations It^{onl} required to reach convergence in the FGMRES algorithm is lower than or equal to 6 for all the tests carried out with the MSRB preconditioner, it does not significantly vary with the FE dimension and, depending on the simulation, it is between 5% and 15% of that obtained by using the PMM preconditioner only, see Figure 3.12a. The computational times $t^{\text{onl}}_{\text{MSRB}}$ required to solve the FE linear system by employing the MSRB preconditioner is reduced of about 85% with respect to the one needed by employing only the PMM preconditioner t_{PMM} for the **GRB** and **aLSRB-P**$_{\mathbf{X^0_h}}$ cases, and is reduced of about 70% in the **aLSRB-X**$^{\mathbf{0}}_{\mathbf{h}}$, see Figure 3.12b. The additional time required by this latter approach is caused by the application of the matrix $\mathbf{X}_h^{-1}(\boldsymbol{\mu})$ to the vector $\mathbf{v}_{k+\frac{1}{2}}$ at each iteration of the FGMRES method (see step 3 in Alg. 6); this is practically performed by solving the corresponding linear system where $\mathbf{X}_h(\boldsymbol{\mu})$ is at the left hand side and $\mathbf{v}_{k+\frac{1}{2}}$ is at the right hand side. The **GRB** and **aLSRB-P**$_{\mathbf{X^0_h}}$ approaches entail a cheaper computation of such a step since in the former we rely on a G-RB method, while in the latter only the (fast) application of $\mathbf{P}_X^{-1}$ is required.

The computational time t_{off} required by the offline phase is reported for all tests, together with the break even point (BEP), that is, the number of online evaluations required to repay the offline phase. Our critreion is based on the wall time comparison:

$$\text{BEP} = \frac{t_{\text{off}}}{t_{\text{PMM}} - t^{\text{onl}}_{\text{MSRB}}},$$

where we indicate by t_{off} the wall time required by the offline computation, i.e. the construction of the RB coarse components. We highlight that the **GRB** case entails a larger offline time than the one required by the other options, due to the need of computing the pressure supremizer snapshots $\mathbf{S}_{\tilde{t}}$ and performing an additional POD. On the other hand, the offline time in the case of **aLSRB-X**$^{\mathbf{0}}_{\mathbf{h}}$ is larger than the one obtained with **aLSRB-P**$_{\mathbf{X^0_h}}$ due to the construction of the RB affine matrices $\mathbf{A}^{q_1,q_2}_{N_k}$, $q_1, q_2 = 1, \dots, Q_a$, because in the former case a FE linear system needs to be solved for each combination of the N_k RB functions $\{\boldsymbol{\xi}_i\}_{i=1}^N$ and Q_a affine terms $\{\mathbf{A}_h^q\}_{q=1}^{Q_a}$, leading to $N \cdot Q_a$ FE linear systems, while by employing $\mathbf{P}_X = \mathbf{P}_{\mathbf{X}_h^0}$, only $N \cdot Q_a$ applications of $\mathbf{P}^{-1}_{\mathbf{X}_h^0}$ need to be performed, boosting the computation of the affine RB structures. By inspecting the BEP values, it emerges that the most convenient approach is obtained by adopting the **aLSRB-P**$_{\mathbf{X^0_h}}$ method. Indeed, such a strategy allows to solve the problem online in a computational time comparable to the one obtained with the **GRB** approach, however entailing a cheaper offline phase, especially when the FE dimension increases. We highlight that this confirms the results obtained in Section 3.4, where the aLSRB solver with $\mathbf{P}_X = \mathbf{P}_{\mathbf{X}_h^0}$ has been shown to be the most accurate and efficient choice among the considered RB solvers.

Table 3.5 – Test case I, fixed accuracy with **GRB**, $L = 4$, $\delta_{RB,k} \approx 5.6 \cdot 10^{-3}\, \forall k$.

N_h	N_k	$t^{\text{onl}}_{\text{MSRB}}$ (sec)	It^{onl}	t_{PMM} (sec)	It_{PMM}	t_{off} (sec)	BEP
52152	9 24 50 113	0.72	3	4.70	40	1514.78	374
320338	9 24 48 118	1.30	3	11.32	42	2951.76	291
1568223	9 23 48 116	5.10	3	30.65	42	9548.40	372

Table 3.6 – Test case I, fixed accuracy with **aLSRB-$\mathbf{X_h^0}$**, $L = 4$, $\delta_{RB,k} \approx 5.6 \cdot 10^{-3}\,\forall k$.

N_h	N_k	$t^{\text{onl}}_{\text{MSRB}}$ (sec)	It^{onl}	t_{PMM} (sec)	It_{PMM}	t_{off} (sec)	BEP
52152	5 13 24 54	1.97	4	4.70	40	1493.10	535
320338	5 13 23 56	4.82	6	11.32	42	3411.82	519
1568223	5 13 23 52	11.25	6	30.65	42	8542.47	437

Table 3.7 – Test case I, fixed accuracy with **aLSRB-$\mathbf{P_{X_h^0}}$**, $L = 4$, $\delta_{RB,k} \approx 5.6 \cdot 10^{-3}\,\forall k$.

N_h	N_k	$t^{\text{onl}}_{\text{MSRB}}$ (sec)	It^{onl}	t_{PMM} (sec)	It_{PMM}	t_{off} (sec)	BEP
52152	5 13 24 53	1.29	4	4.70	40	1374.38	395
320338	5 13 23 55	2.57	6	11.32	42	2727.60	307
1568223	5 13 23 52	5.36	6	30.65	42	6975.20	274

Table 3.8 – Test case I, fixed dimension with **GRB**, $N_k^u = N_k^p = N_k^s = 10\,\forall k$.

N_h	L	$t^{\text{onl}}_{\text{MSRB}}$ (sec)	It^{onl}	t_{PMM} (sec)	It_{PMM}	t_{off} (sec)	BEP
52152	9	0.51	2	4.70	40	2476.72	584
320338	7	1.24	3	11.32	42	4546.66	447
1568223	8	4.74	3	30.65	42	18369.68	707

Table 3.9 – Test case I, fixed dimension with **aLSRB-$\mathbf{X_h^0}$**, $N_k^u = N_k^p = 10\,\forall k$.

N_h	L	$t^{\text{onl}}_{\text{MSRB}}$ (sec)	It^{onl}	t_{PMM} (sec)	It_{PMM}	t_{off} (sec)	BEP
52152	9	1.51	5	4.70	40	2507.38	776
320338	8	5.12	6	11.32	42	6770.73	1086
1568223	8	10.14	5	30.65	42	15121.68	735

Table 3.10 – Test case I, fixed dimension with **aLSRB-$\mathbf{P_{X_h^0}}$**, $N_k^u = N_k^p = 10\,\forall k$.

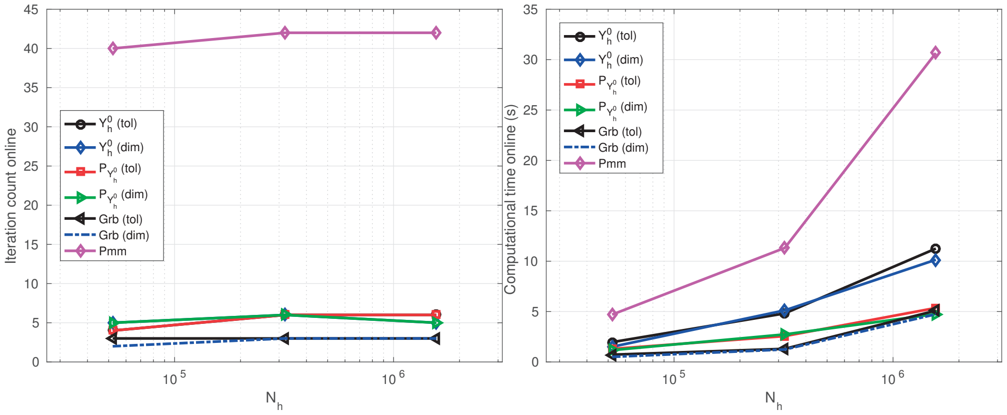
N_h	L	$t^{\text{onl}}_{\text{MSRB}}$ (sec)	It^{onl}	t_{PMM} (sec)	It_{PMM}	t_{off} (sec)	BEP
52152	9	1.17	5	4.70	40	2886.91	810
320338	8	2.74	6	11.32	42	5475.75	633
1568223	8	4.75	5	30.65	42	11489.38	442

3.6.2 Test case II: parametrized carotid bifurcations

In the second test case, we consider parametrized Stokes flows in a carotid bifurcation, whose shape varies according to a set of parameters. Even though the Stokes equations are not suited for simulating the blood flow in an artery as large as the carotid, we are interested in measuring the capabilities of the proposed MSRB preconditioning technique when dealing with flows complex geometries, since our final goal is to deal with cardiovascular applications and compare the results with the ones obtained with the standard RB method.

The computational domain $\Omega(\boldsymbol{\mu})$ is obtained by deforming a reference domain Ω^0, such that $\partial\Omega^0 = \Gamma_w \cup \Gamma_{in} \cup \Gamma_{out}$, by setting

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



(a) Iterations vs N_h. (b) Computational times (sec) vs N_h.

Figure 3.12 – Test case I, iteration number and computational times vs N_h.

where the displacement $\vec{d}(\boldsymbol{\mu})$ is computed as the solution of the following parametrized elliptic problem

$$\begin{cases} -\Delta \vec{d}(\boldsymbol{\mu}) = \vec{0} & \text{in } \Omega^0 \\ \vec{d}(\boldsymbol{\mu}) = \vec{0} & \text{on } \Gamma_{in} \cup \Gamma_{out} \\ \dfrac{\partial \vec{d}(\boldsymbol{\mu})}{\partial \vec{n}} = \vec{h}(\boldsymbol{\mu}) & \text{on } \Gamma_w. \end{cases} \tag{3.71}$$

The parametrized datum $\vec{h}(\boldsymbol{\mu})$ represents a stress load entailing a deformation leading to the narrowing of one of the branches of the bifurcation. We consider as parameter $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathcal{D} = [4,\, 5] \times [0,\, 0.5]$ and introduce a $\boldsymbol{\mu}-$dependent radius $r(\vec{x}; \boldsymbol{\mu}) = r(\boldsymbol{\mu}) = \sqrt{(x_1 + 0.8)^2 + (x_2 - \mu_1)^2 + (x_3)^2}$, $R = 0.65$ and the region $A(\boldsymbol{\mu})$, such that

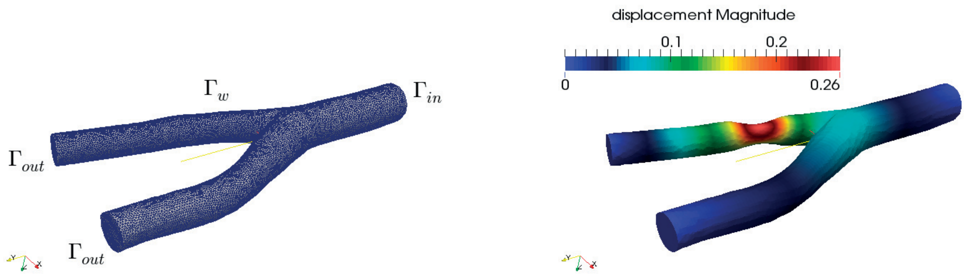
$$A(\boldsymbol{\mu}) = \{\vec{x} \in \mathbb{R}^3 : \; (r(\boldsymbol{\mu}))^2 < R^2\},$$

which identifies the portion of volume where $\vec{h}(\boldsymbol{\mu})$ is loaded as follows

$$\vec{h}(\boldsymbol{\mu}) = \vec{h}(\vec{x}; \boldsymbol{\mu}) = -\mu_2 \Big(1 - \frac{r^2(\vec{x})}{R^2}\Big) \vec{n} \mathcal{X}_{A(\boldsymbol{\mu})}(\vec{x}), \qquad \vec{x} \in \mathbb{R}^3,$$

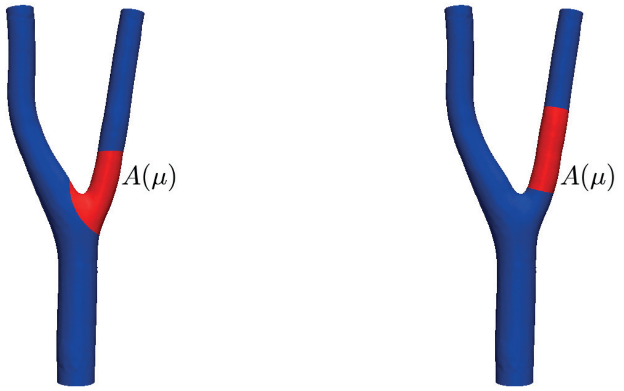
where and $\mathcal{X}_{A(\boldsymbol{\mu})}(\vec{x})$ is the indicator function over the set $A(\boldsymbol{\mu})$; two examples of the region identified by $a(\boldsymbol{\mu})$ for two values of the parameter $\boldsymbol{\mu}$ are reported in Figure 3.13c and 3.13d. This parametrization entails a narrowing of the straight branch in different positions along the coordinate x_2 (according to the value of μ_2) and simulates an occlusion. The reference domain Ω^0 is shown in Figure 3.13a, whereas an example of deformation computed for $\boldsymbol{\mu} = (5.0, 0.5)$ is given in Figure 3.13b. Examples of solutions for different values of the parameter $\boldsymbol{\mu}$ are shown in Figure 3.14a-3.14b and 3.14c-3.14d.

We remark that the solution $\vec{d}(\boldsymbol{\mu})$ of problem (3.71) is not known analytically; conse-



(a) Reference domain Ω^0.

(b) Displacement field for $\boldsymbol{\mu} = (5.0, 0.5)$.



(c) $A(\boldsymbol{\mu})$ with $\mu_1 = 4$.

(d) $A(\boldsymbol{\mu})$ with $\mu_1 = 5$.

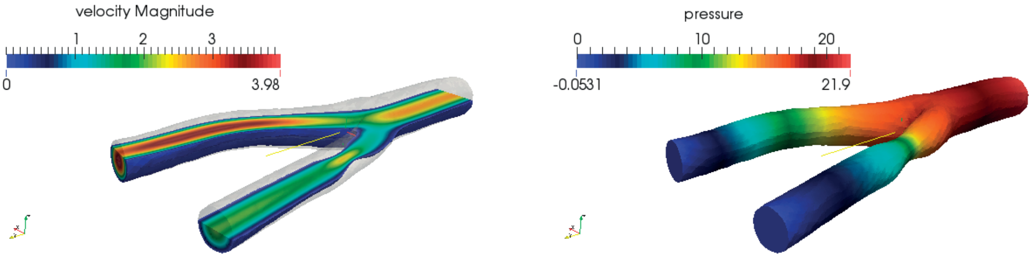
Figure 3.13 – Test case II, top row: reference domain Ω^0 (left) and displacement $\mathbf{d}(\boldsymbol{\mu})$ for $\boldsymbol{\mu} = (5.0, 0.5)$; bottom row: example of region $A(\boldsymbol{\mu})$ for two values of $\boldsymbol{\mu}$.

quently, its numerical approximation $\vec{d}_h(\boldsymbol{\mu})$ is computed employing the FE method on its corresponding variational formulation. We denote by $\mathbf{d}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^d}$ the solution of the corresponding FE linear system.

In our computations, Taylor-Hood FE ($\mathcal{P}_2 - \mathcal{P}_1$), with a mesh leading to $N_h = N_h^u + N_h^p = 3'198'820$ degrees of freedom, are employed for the FE discretization of the Stokes problem, on 360 computing cores.

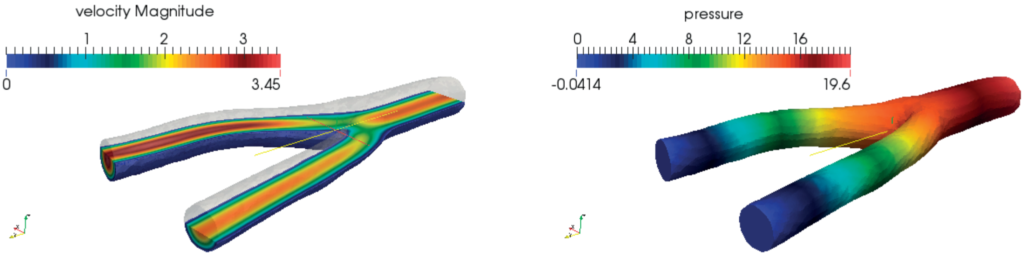
Simulation setup

When considering a new instance of the parameter $\boldsymbol{\mu}$, we compute $\mathbf{d}(\boldsymbol{\mu})$ by solving the corresponding FE linear system with the PCG method, preconditioned with the AMG preconditioner. The system is solved up to a tolerance 10^{-8} on the Euclidean norm of the residual rescaled with the Euclidean norm of the right hand side. The computation of the deformation $\mathbf{d}(\boldsymbol{\mu})$ requires on average 1.9 seconds and this time is not included in the results reported, since it does not vary in the different scenarios presented. Notice that we could accelerate the computation of $\mathbf{d}(\boldsymbol{\mu})$ by employing



(a) Slice of velocity field for $\boldsymbol{\mu} = (5.0, 0.5)$.

(b) Pressure field for $\boldsymbol{\mu} = (5.0, 0.5)$.



(c) Slice of velocity field for $\boldsymbol{\mu} = (4.0, 0.0)$.

(d) Pressure field for $\boldsymbol{\mu} = (4.0, 0.0)$.

Figure 3.14 – Test case II, numerical solution for two values of $\boldsymbol{\mu}$ obtained with the MSRB preconditioning technique.

the MSRB preconditioning strategy or the standard RB method to deal with problem (3.71). Then, the solution of the Stokes problem (3.6) is computed employing the MSRB preconditioner. Here we report in particular the results obtained with the **aLSRB-P$_{\mathbf{X_h^0}}$** case and the fixed dimension approach only, however a detailed analysis similar to the one carried out to Test case I can also be done. For the aim of RB spaces construction, we use $n_s = 350$ snapshots, which are computed incrementally as explained in Section 3.5.4, with $M = 3$ and $n_s^1 = 100$, $n_s^2 = 100$ and $n_s^3 = 150$. Then, we set $\varepsilon_r = 10^{-7}$, by choosing $N_k^u = N_k^p = 50$ for any $k = 0, \dots, L-1$, leading to L coarse operators with dimension $N_k = 100$ for any $k = 0, \dots, L-1$. We test the resulting preconditioner on 100 online instances of the parameter randomly chosen, by solving the resulting FE problem up to a tolerance 10^{-5}. For the MSRB preconditioner, we employ MDEIM (with tolerance $\delta_{\text{MDEIM}} = 10^{-4}$) to compute an approximated affine decomposition of the matrices $\mathbf{D}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$, allowing us to cheaply assemble online the coarse operators $\mathbf{A}_{N_k}(\boldsymbol{\mu})$, $k = 0, \dots, L-1$.

We compare the results obtained with the MSRB precondtioner with the ones obtained by relying on the standard RB method, where the **aLSRB-P$_{\mathbf{X_h^0}}$** approach detailed in Section 3.3 is used as solver. For this latter, we build the RB basis functions by using POD with a tolerance of 10^{-9} on $n_s = 350$ snapshots; then we construct the RB approximation by affinely approximating the FE Stokes right hand sides and matrices by using DEIM and MDEIM, respectively. Indeed, we remark that, as highlighted in Section 3.3.2, the standard RB method also relies on the affine dependence of the FE right hand side $\mathbf{g}(\boldsymbol{\mu})$.

Table 3.11 – Test case II, DEIM and MDEIM number of affine basis functions computed during the offline phase as function of the tolerances δ_{DEIM} and δ_{MDEIM} (always chosen with the same value). The number of affine components affects the duration of the offline phase of a time t_{affine}, which in the **aLSRB-$\mathbf{P}_{\mathbf{X_h^0}}$** solver case depends quadratically on the number of affine terms.

$\delta_{\text{DEIM}} = \delta_{\text{MDEIM}}$	MDEIM - $\mathbf{D}(\boldsymbol{\mu})$	MDEIM - $\mathbf{B}(\boldsymbol{\mu})$	DEIM - $\mathbf{f}(\boldsymbol{\mu})$	DEIM - $\mathbf{r}(\boldsymbol{\mu})$	t_{affine} (sec)
1e-02	1	3	3	4	75.41
1e-03	1	6	6	13	184.68
1e-04	3	17	15	25	1165.29
1e-05	8	36	29	48	5013.85
1e-06	19	79	63	117	49129.40

Since in the considered test case this assumption is not satisfied, DEIM is performed on the right hand side to compute an affine approximation of the vectors $\mathbf{f}(\boldsymbol{\mu})$ and $\mathbf{r}(\boldsymbol{\mu})$.

Numerical results: comparison with the standard RB method

We show the results obtained by using the **aLSRB-$\mathbf{P}_{\mathbf{X_h^0}}$** method as solver on a set of 100 instances of the parameter and varying the tolerances δ_{MDEIM} and δ_{DEIM} employed for the MDEIM and DEIM algorithms, respectively. In Table 3.11, the number of affine components for the different FE arrays is reported, together with the computational time (part of the offline phase of the standard RB method) t_{affine} to build and store the affine RB matrices $\mathbf{A}_N^{q_1,q_2}$, $q_1, q_2 = 1, \dots, Q_a$ in (3.41) and the RB vectors $\mathbf{g}_N^{q_1,q_2}$, $q_1, \dots, Q_a, q_1, \dots, Q_g$ in (3.42). Notice that the number of affine basis functions largely affects the time t_{affine}, leading overall to a very demanding offline phase.

By setting $\delta_{RB} = 10^{-9}$ to construct the RB space, we obtain $N_u = 327$ and $N_p = 111$ basis functions for velocity and pressure, respectively. In order to evaluate the accuracy of the RB solution, we compute the average relative residual r_{RB} of the FE problem evaluated on the RB solution defined as in (2.49), which we report in Table 3.12. As a matter of fact, in order to obtain an accurate RB solution, it is mandatory to build an accurate approximate affine decomposition of the FE arrays, cf. Table 3.12, since the accuracy of the RB solution is strongly related to the accuracy of the affine approximations. The online time t_{onl} to assemble and solve the RB problem is significantly affected by the values δ_{DEIM} and δ_{MDEIM} and reaches up to 8.66 seconds in the most demanding case. In particular, the time for assembling the RB matrix $\mathbf{A}_N(\boldsymbol{\mu})$ and the time for assembling the RB right hand side $\mathbf{g}_N(\boldsymbol{\mu})$ are the most affected ones by the number of affine components. As regards the computational time t_{off} required by the offline phase, it largely increases according to the number of affine terms, since it takes into account the time t_{affine} reported in Table 3.11.

In Table 3.13, the results obtained with the FGMRES method preconditioned with MSRB preconditioner (with **aLSRB-$\mathbf{P}_{\mathbf{X_h^0}}$** coarse operators) are presented. We employ MDEIM

Table 3.12 – Test case II, results with $\mathbf{aLSRB\text{-}P}_{\mathbf{X_h^0}}$ solver, basis computed with tolerance $\delta_{RB} = 10^{-9}$ leading to $N_u = 327$ and $N_p = 111$ basis functions. The accuracy of the method largely depends on the system approximation carried out with DEIM and MDEIM.

$\delta_{\text{DEIM}} = \delta_{\text{MDEIM}}$	r_{RB}	$t_{\text{RB}}^{\text{onl}}$ (sec)	t_{off} (sec)
1e-02	1.9e-02	5.75	41931.61
1e-03	4.0e-03	5.39	42040.87
1e-04	1.1e-03	5.33	43021.49
1e-05	2.8e-04	5.81	46870.05
1e-06	6.3e-05	8.66	90985.60

with $\delta_{\text{MDEIM}} = 10^{-4}$ to build an approximated affine decomposition of the FE matrices $\mathbf{D}(\boldsymbol{\mu})$ and $\mathbf{B}(\boldsymbol{\mu})$, leading to $Q_d = 3$ and $Q_b = 17$ affine basis functions, respectively. A large MDEIM tolerance δ_{MDEIM} is employed since each RB coarse operator is trained to solve equation (3.54) up to an accuracy greater than $\delta_{\text{MDEIM}} = 10^{-4}$; therefore such value does not affect the local accuracy of any coarse operator, as deeply investigated in the case of the thermal beam in Section 2.4.2. Furthermore, we notice that in this context there is no need to employ DEIM to approximate $\mathbf{f}(\boldsymbol{\mu})$ and $\mathbf{r}(\boldsymbol{\mu})$, as explained in Section 2.2.4.

$L = 4$ RB spaces are computed with a dimension $N_k^u = N_k^p = 50$ for $k = 0, 1, 2, 3$ for both velocity and pressure, respectively; as a matter of fact, the convergence up to a tolerance of 10^{-5} on r_{RB} is reached on average in 5 iterations and about 6.45 seconds. The cheaper computation of the solution is motivated by the milder dependence on the MDEIM tolerance, which allows to obtain a significantly more accurate solution (with a residual r_{RB} lower than 10^{-5}) in a shorter computational time, compared to the one computed with the standard RB method. In addition, the results obtained show that a cheaper offline phase is also achieved. This is motivated by two reasons: since the employed affine decomposition is coarser than the one exploited by the aLSRB solver, a lower number of affine RB components must be precomputed and stored; secondly, the sequential construction of the RB spaces allows to exploit the (still in construction) MSRB preconditioner, thus boosting the computation of the snapshots.

Finally, we compare the iteration count and the computational time employed by MSRB-preconditioned FGMRES iterations with the ones needed to solve the same problem with the FGMRES method preconditioned with the PMM preconditioner, reported in Table 3.13 as well. When this latter technique is employed, the problem is solved in about 80.69 seconds and 87 iterations, on average. Therefore the proposed MSRB technique allows to obtain the solution by reducing by more than 92% the time needed by employing the PMM preconditioner only, yielding a break-even point of 627 evaluations.

Table 3.13 – Test case II, MSRB preconditioner results with FGMRES with a final tolerance $\varepsilon_r = 10^{-5}$. The RB spaces are built with the fixed dimension approach, with $N_k^u = N_k^p = 50$, $\forall k$, and $\mathbf{aLSRB}$-$\mathbf{P}_{\mathbf{X_h^0}}$ coarse operators. The computation is carried out with 360 cores, such that $\sim$ 8890 dofs per core.

N_h	L	$t_{\text{MSRB}}^{\text{onl}}$ (sec)	It^{onl}	t_{PMM} (sec)	It_{PMM}	t_{off} (sec)
3198820	4	6.45	5	80.69	87	46554.90

Conclusions

In this chapter, we have proposed a new algebraic LSRB method for the Stokes equations, we have verified its well-posedness and outlined its construction. Furthermore, we have compared it to the current state-of-the-art ROM techniques for dealing with the parametrized Stokes equations: *i*) our aLSRB approach extends the current LSRB method, since it is applicable when we consider a geometric parametrization for which an analytical map is not known *a priori*; *ii*) we have shown it allows to cut offline and online costs and, at the same time, achieve a better online accuracy if compared with a G-RB solver exploiting a velocity enrichment formulation.

Secondly, we have developed a new framework for the MSRB-precondioned FGMRES iterations for the Stokes equations, by exploiting either a G-RB or aLSRB formulation for building a RB coarse component. We have outlined the advantages and drawbacks of either option, and compared them with the PMM preconditioner to solve large-scale FE linear systems. As a matter of fact, exploiting the MSRB preconditioning technique largely improves the efficiency of the PMM preconditioner when a new instance of the parameter is considered. Also a comparison with a RB solver has been carried out, highlighting that the performances (in terms of computational time during both the offline and online phases) of the MSRB preconditioner benefit from its milder dependence on the affine decomposition of the FE arrays with respect to the standard RB solver. As a matter of fact, the results look promising towards the extension to nonlinear saddle-point problems as the parametrized Navier-Stokes equations.

Finally, even if we focused on the Stokes equations only, both the RB and MSRB preconditioning methods proposed in this chapter are straightforwardly applicable to other parametrized linear saddle-point systems.

4 RB methods and multi space RB preconditioners for parametrized Navier-Stokes equations

In this chapter we consider the unsteady Navier-Stokes (NS) equations in parametrized domains. After recalling their differential and weak formulation and the corresponding FE approximation, we present a new model reduction method obtained by extending the currently employed RB methods for linear PDEs in deformed domains, as in [Manzoni and Negri, 2017], and unsteady NS equations where only physical parameters are considered [Veroy and Patera, 2005, Negri et al., 2015a]. In particular, we treat the NS nonlinearity by adopting a double POD hyper-reduction algorithm which relies on (M)DEIM.

We then extend the MSRB preconditioner to the case of the unsteady NS equations. Starting from the framework developed in Section 2.5 for linear parabolic PDEs, here we adopt a velocity enriching Galerkin RB approach, as done for the parametrized Stokes equations in Section 3.5.1. Additionally, we exploit the double POD hyper-reduction algorithm developed for the RB approximation of parametrized NS equation to enhance the efficiency of the resulting preconditioner. Numerical examples are presented to show the advantages of the proposed methodologies in terms of both efficiency and accuracy.

4.1 Parametrized Navier-Stokes equations

In this section we introduce the Navier-Stokes equations for an incompressible, homogeneous, Newtonian fluid. Given an open bounded and $\boldsymbol{\mu}$-dependent domain $\Omega(\boldsymbol{\mu}) \subset \mathbb{R}^d$, $d = 2, 3$, such that, for any $\boldsymbol{\mu} \in \mathcal{D}$, $\partial\Omega(\boldsymbol{\mu}) = \Gamma_{out}(\boldsymbol{\mu}) \cup \Gamma_{in}(\boldsymbol{\mu}) \cup \Gamma_w(\boldsymbol{\mu})$ 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 $T > 0$, let us consider the following nonlinear differential problem

$$\begin{cases} \dfrac{\partial \vec{u}(\boldsymbol{\mu})}{\partial t} + \vec{u}(\boldsymbol{\mu}) \cdot \nabla \vec{u}(\boldsymbol{\mu}) - \nabla \cdot \boldsymbol{\sigma}(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu})) + \nabla p(\boldsymbol{\mu}) = \vec{0} & \text{in } \Omega(\boldsymbol{\mu}) \times (0,T) \\ \nabla \cdot \vec{u}(\boldsymbol{\mu}) = 0 & \text{in } \Omega(\boldsymbol{\mu}) \times (0,T) \\ \vec{u}(\boldsymbol{\mu}) = \vec{0} & \text{on } \Gamma_w(\boldsymbol{\mu}) \times (0,T) \\ \vec{u}(\boldsymbol{\mu}) = \vec{g}_{NS}(\boldsymbol{\mu}) & \text{on } \Gamma_{in}(\boldsymbol{\mu}) \times (0,T) \\ \boldsymbol{\sigma}(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu}))\vec{n}(\boldsymbol{\mu}) = \vec{0} & \text{on } \Gamma_{out}(\boldsymbol{\mu}) \times (0,T) \\ \vec{u}(0;\boldsymbol{\mu}) = \vec{u}_0 & \text{on } \Omega(\boldsymbol{\mu}), \end{cases} \tag{4.1}$$

where $\vec{u}(\boldsymbol{\mu})$ and $p(\boldsymbol{\mu})$ are the velocity and the pressure of the fluid and $\boldsymbol{\sigma}(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu}))$ is the stress tensor defined as

$$\boldsymbol{\sigma}(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu})) = -p(\boldsymbol{\mu})\mathbf{I} + 2\nu\boldsymbol{\varepsilon}(\vec{u}(\boldsymbol{\mu})). \tag{4.2}$$

Here $\nu = \nu(\boldsymbol{\mu})$ denotes the (possibly parameter-dependent) kinematic viscosity that is, $\nu = \bar{\mu}/\bar{\rho}$, being $\bar{\mu}$ and $\bar{\rho}$ the dynamic viscosity and density, respectively, and

$$\boldsymbol{\varepsilon}(\vec{u}(\boldsymbol{\mu})) = \frac{1}{2}\left(\nabla \vec{u}(\boldsymbol{\mu}) + \nabla \vec{u}(\boldsymbol{\mu})^T\right) \tag{4.3}$$

is the strain tensor. The Dirichlet boundary datum is supposed to be time-dependent, such that the time dependence can be expressed by separating time and $\boldsymbol{\mu}$, that is, we assume

$$\vec{g}_{NS}(\boldsymbol{\mu}) = \vec{g}_{NS}(t;\boldsymbol{\mu}) = w(t)g_D(\boldsymbol{\mu}). \tag{4.4}$$

In this context, we define the well-known Reynolds number Re as the non-dimensional ratio of convection to diffusion

$$Re = \frac{L\bar{U}}{\nu}, \tag{4.5}$$

where L and $\bar{U}$ are the characteristic length of the domain and velocity of the flow. In this work we are interested in laminar flows, featuring a Reynolds number in the range $[1, 10^3]$.

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

$$V = \left\{\vec{v} \in \left[H^1(\Omega(\boldsymbol{\mu}))\right]^d : \vec{v}|_{\Gamma_{in}(\boldsymbol{\mu})\cup\Gamma_w(\boldsymbol{\mu})} = \vec{0}\right\}, \qquad Q = L^2(\Omega(\boldsymbol{\mu})), \tag{4.6}$$

the functional spaces for velocity and pressure, respectively. We highlight that, as in Section 3.1, 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 $(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu})) \in V \times Q$ such that

$$\begin{aligned}\left(\frac{\partial \vec{u}(\boldsymbol{\mu})}{\partial t}, \vec{v}\right) &+ d(\vec{u}(\boldsymbol{\mu}), \vec{v}; \boldsymbol{\mu}) + b(\vec{v}, p(\boldsymbol{\mu}); \boldsymbol{\mu}) + c(\vec{u}(\boldsymbol{\mu}), \vec{u}(\boldsymbol{\mu}), \vec{v}; \boldsymbol{\mu}) \\ &+ b(q, \vec{u}(\boldsymbol{\mu}); \boldsymbol{\mu}) = F_1(t, \vec{v}; \boldsymbol{\mu}) + F_2(t, q; \boldsymbol{\mu}) \quad \forall (\vec{v}, q) \in V \times Q\end{aligned} \tag{4.7}$$

with $\vec{u}(0; \boldsymbol{\mu}) = \vec{u}_0$ as initial condition. The forms in (4.7) are defined, for any $\vec{u}, \vec{v}, \vec{w} \in V$ and $q \in Q$, as

$$d(\vec{u}, \vec{v}; \boldsymbol{\mu}) = \int_{\Omega(\boldsymbol{\mu})} \nu(\boldsymbol{\mu})(\nabla \vec{u} + \nabla \vec{u}^T) : \nabla \vec{v} \, d\Omega(\boldsymbol{\mu}) \tag{4.8}$$

$$b(q, \vec{v}; \boldsymbol{\mu}) = -\int_{\Omega(\boldsymbol{\mu})} q \nabla \cdot \vec{v} \; d\Omega(\boldsymbol{\mu}) \tag{4.9}$$

$$c(\vec{u}, \vec{v}, \vec{w}; \boldsymbol{\mu}) = \int_{\Omega(\boldsymbol{\mu})} (\vec{v} \cdot \nabla) \vec{u} \cdot \vec{w} \; d\Omega(\boldsymbol{\mu}), \tag{4.10}$$

and $F_1(t, \vec{v}; \boldsymbol{\mu})$, $F_2(t, q; \boldsymbol{\mu})$ are linear $(t, \boldsymbol{\mu})$-dependent forms accounting for the contribution of the lifting function.

4.1.1 FE discretization and BDF time integration

Problem (4.7) is first discretized in space by means of the FE method, and in time with the BDF scheme. Given two finite dimensional spaces $V_h \subset V$ and $Q_h \subset Q$ with dimensions N_h^u, N_h^p, such that $N_h^u + N_h^p = N_h$, respectively, the semi-discretized (in space) problem reads: for any $t \in (0,T)$, find $(\vec{u}_h(\boldsymbol{\mu}), p_h(\boldsymbol{\mu})) \in V_h \times Q_h$ such that

$$\begin{aligned}\left(\frac{\partial \vec{u}_h(\boldsymbol{\mu})}{\partial t}, \vec{v}_h\right) &+ d(\vec{u}_h(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu}) + b(\vec{v}_h, p_h(\boldsymbol{\mu}); \boldsymbol{\mu}) + c(\vec{u}_h(\boldsymbol{\mu}), \vec{u}_h(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu}) \\ &+ b(q, \vec{u}_h(\boldsymbol{\mu}); \boldsymbol{\mu}) = F_1(t, \vec{v}_h; \boldsymbol{\mu}) + F_2(t, q_h; \boldsymbol{\mu}) \quad \forall (\vec{v}_h, q_h) \in V_h \times Q_h.\end{aligned} \tag{4.11}$$

A fully-discretized problem is finally obtained from (4.11) by using the BDF scheme of order $\sigma_1 = \{1,\, 2\}$ (see Section 1.1.2). Let us introduce a partition of the interval $[0,T]$ in N_t subintervals of equal size $\Delta t = T/N_t$, such that $t_n = n\Delta t$, 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 $n \geq \sigma_1 - 1$ 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

$$\begin{aligned}\left(\frac{1\vec{u}_h^{n+1}(\boldsymbol{\mu}) - \vec{u}_h^{n,\sigma}(\boldsymbol{\mu})}{\Delta t}, \vec{v}_h\right) &+ d(\vec{u}_h^{n+1}(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu}) + b(\vec{v}_h, p_h^{n+1}(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ &+ c(\vec{u}_h^{n+1}(\boldsymbol{\mu}), \vec{u}_h^{n+1}(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu}) + b(q, \vec{u}_h^{n+1}(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ &= F_1(t, \vec{v}_h; \boldsymbol{\mu}) + F_2(t, q_h; \boldsymbol{\mu}) \quad \forall (\vec{v}_h, q_h) \in V_h \times Q_h,\end{aligned} \tag{4.12}$$

where $(\vec{u}_h^n(\boldsymbol{\mu}), p_h^n(\boldsymbol{\mu}))$ is the FE solution at time n and $\vec{u}_h^{n,\sigma}(\boldsymbol{\mu})$ is defined as in (1.20).

Following this strategy, after spatial and time discretizations, the fully discrete formulation of problem (4.12) consists in a nonlinear problem to be solved at each time-step. An approximation of this nonlinear problem can be obtained, for example, by using the Newton method, this latter requiring at each iteration the assembly of the Jacobian matrix and the solution of a linear system. 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 associated linear system. To reduce the computational cost related with the use of a fully implicit BDF approach, we consider instead a semi-implicit BDF scheme, for which the nonlinear terms in $\vec{u}_h^n(\boldsymbol{\mu})$ are extrapolated by means of the Newton-Gregory backward polynomials, as similarly done in [Gervasio et al., 2006, Forti and Dedè, 2015]. To this aim, we consider the following extrapolations of order $\sigma_1 = 1, 2$ for the velocity at the discrete time t_{n+1}:

$$\vec{u}_h^{n,*}(\boldsymbol{\mu}) = \begin{cases} \vec{u}_h^n(\boldsymbol{\mu}) & \text{if } \sigma_1 = 1 \\ 2\vec{u}_h^n(\boldsymbol{\mu}) - \vec{u}_h^{n-1}(\boldsymbol{\mu}) & \text{if } \sigma_1 = 2, \end{cases}$$

and starting from the fully implicit formulation (4.12), we use the above extrapolations by Newton-Gregory backward polynomials. In this way, the fully discrete linearized semi-implicit formulation of problem (4.12) reads: given $\boldsymbol{\mu} \in \mathcal{D}$, $\vec{u}_h^n(\boldsymbol{\mu}), \dots, \vec{u}_h^{n+1-\sigma}(\boldsymbol{\mu})$, for $n \geq \sigma_1 - 1$ 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

$$\begin{aligned} \left(\frac{\alpha_1 \vec{u}_h^{n+1}(\boldsymbol{\mu}) - \vec{u}_h^{n,\sigma}(\boldsymbol{\mu})}{\Delta t}, \vec{v}_h \right) &+ d(\vec{u}_h^{n+1}(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu}) + b(\vec{v}_h, p_h^{n+1}(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ &+ c(\vec{u}_h^{n+1}(\boldsymbol{\mu}), \vec{u}_h^{n,*}(\boldsymbol{\mu}), \vec{v}_h; \boldsymbol{\mu}) + b(q, \vec{u}_h^{n+1}(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ &\quad = F_1(t, \vec{v}_h; \boldsymbol{\mu}) + F_2(t, q_h; \boldsymbol{\mu}) \quad \forall (\vec{v}_h, q_h) \in V_h \times Q_h, \end{aligned} \tag{4.13}$$

Thanks to this time discretization, the fully discrete semi-implicit formulation (4.13) 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. We consider problem (4.13) as our high-fidelity approximation of the unsteady NS equations.

4.1.2 Algebraic formulation

Problem (4.13) leads to a sequence in time of parametrized linear systems of the form

$$\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) \begin{bmatrix} \mathbf{u}^{n+1}(\boldsymbol{\mu}) \\ \mathbf{p}^{n+1}(\boldsymbol{\mu}) \end{bmatrix} = \mathbf{g}^{n+1}(\boldsymbol{\mu}) \qquad n = 0, \dots, N_t - 1, \tag{4.14}$$

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

$\mathbf{u}^0(\boldsymbol{\mu}) = \mathbf{u}_0 \in \mathbb{R}^{N_h^u}$ as the initial condition. The arrays $\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ and $\mathbf{g}^{n+1}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ are defined as

$$\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) = \begin{bmatrix} \frac{\alpha_1}{\Delta t}\mathbf{M}^u(\boldsymbol{\mu}) + \mathbf{D}(\boldsymbol{\mu}) + \mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) & \mathbf{B}^T(\boldsymbol{\mu}) \\ \mathbf{B}(\boldsymbol{\mu}) & \mathbf{0} \end{bmatrix} \tag{4.15}$$

$$\mathbf{g}^{n+1}(\boldsymbol{\mu}) = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}^u(\boldsymbol{\mu})\mathbf{u}^{n,\sigma}(\boldsymbol{\mu}) + \mathbf{f}_1^{n+1}(\boldsymbol{\mu}) \\ \mathbf{f}_2^{n+1}(\boldsymbol{\mu}) \end{bmatrix}. \tag{4.16}$$

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

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

$\mathbf{D}(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^u \times N_h^u}$ 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 in (3.8) and $\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) \in \mathbb{R}^{N_h^u \times N_h^u}$ is the matrix arising from the linearization of the nonlinear convective term,

$$\left(\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\right)_{ij} = c(\phi_j^u, \vec{u}_h^{n,*}(\boldsymbol{\mu}), \phi_i^u; \boldsymbol{\mu}) \qquad i,j = 1,\dots,N_h^u. \tag{4.17}$$

Remark 4.1.1. *Similarly to what discussed for the Stokes equations in Section 3.1, the velocity and pressure FE spaces V_h and Q_h must yield a divergence matrix $\mathbf{B}(\boldsymbol{\mu})$ that fulfills the inf-sup condition* (3.13) *to guarantee the well-posedness of* (4.14). *A possible choice, which is the one used in the numerical experiments, consists in employing Taylor-Hood FE spaces, that is $\mathcal{P}_2$ and $\mathcal{P}_1$ basis functions for velocity and pressure, respectively.*

The efficient solution of the sequence of linear systems defined in (4.14) calls into play suitable numerical methods. Several techniques have been proposed to deal with systems (4.14), and in addition to the already mentioned multilevel, domain decomposition methods and block preconditioners, which have been designed for both steady and unsteady NS equations, we specifically recall the SIMPLE method [Segal et al., 2010, Vuik et al., 2000], based on Patankar's Semi-Implicit Pressure Linked Equation technique [Patankar, 1980]. We start from the following factorization

$$\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) = \mathcal{L}_{bt}(\boldsymbol{\mu})\mathcal{U}(\boldsymbol{\mu}) \tag{4.18}$$

with

$$\mathcal{L}_{bt}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{F}(\boldsymbol{\mu}) & 0 \\ \mathbf{B}(\boldsymbol{\mu}) & \mathbf{S}(\boldsymbol{\mu}) \end{bmatrix}, \qquad \mathcal{U}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{I}_{N_h^u} & \mathbf{F}^{-1}(\boldsymbol{\mu})\mathbf{B}^T(\boldsymbol{\mu}) \\ 0 & \mathbf{I}_{N_h^p} \end{bmatrix}, \tag{4.19}$$

where $\mathbf{S}(\boldsymbol{\mu}) = -\mathbf{B}(\boldsymbol{\mu})\mathbf{F}^{-1}(\boldsymbol{\mu})\mathbf{B}^T(\boldsymbol{\mu})$ denotes the Schur complement matrix and $\mathbf{F}(\boldsymbol{\mu})$ is the block (1,1) of the matrix $\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$ defined in (4.15). The application of the

Algorithm 12 Computation of $\mathbf{z} = \mathbf{P}_{\text{SIMPLE}}^{-1}\mathbf{v}$ ($\boldsymbol{\mu}$ is omitted)

```
1: approximately solve with inner iterations $\mathbf{F}\mathbf{z}_u = \mathbf{v}_u$
2: approximately solve with inner iterations $\widetilde{\mathbf{S}}\mathbf{z}_p = \mathbf{v}_p - \mathbf{B}\mathbf{z}_u$
3: update $\mathbf{z}_u = \mathbf{z}_u - \mathbf{D}_F^{-1}\mathbf{B}^T\mathbf{z}_p$
```

SIMPLE preconditioner, which we denote by $\mathbf{P}_{\text{SIMPLE}}(\boldsymbol{\mu})$, to the Krylov basis consists in employing the procedure is reported in Algorithm 12 and requires to approximately solve two FE linear systems up to a provided tolerance. The matrix $\mathbf{D}_F \in \mathbb{R}^{N_h^u \times N_h^u}$ is the diagonal of $\mathbf{F}$ and $\widetilde{\mathbf{S}} \in \mathbb{R}^{N_h^p \times N_h^p}$ is an approximation to the Schur complement $\mathbf{S}$, where $\mathbf{F}$ is substituted by $\mathbf{D}_F$. In our implementation, steps 1-2 are carried out by inner GMRES iterations. Suitable modifications of SIMPLE preconditioner have also been proposed, in particular we recall SIMPLER preconditioner, which provides Reynolds-independent convergence rates, and MSIMPLER, which improves the latter by substituting $\mathbf{F}(\boldsymbol{\mu})$ with the velocity mass matrix in the Schur complement and in the update step; we refer to [Van Doormaal and Raithby, 1984, Wesseling, 2009, Vuik et al., 2009, Segal et al., 2010] for further details about SIMPLE and its variants.

In our numerical experiments, we will use a FE approximation and BDF2 time-approximation scheme; the linearized linear system (4.14) will be solved with the SIMPLE-preconditioned FGMRES method. On the other hand, in the numerical experiments for the MSRB preconditioner for the unsteady NS equations, the SIMPLE preconditioner will play the role of fine grid component.

4.2 A ROM framework for parametrized unsteady NS equations in moving domains

In this section we present a ROM technique to reduce the cost needed to solve the FE system (4.14), 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. The RB approximation of velocity and pressure fields at time t_n is expressed as a linear combination of the RB basis functions,

$$\mathbf{u}^n(\boldsymbol{\mu}) \approx \mathbf{V}_{N_u}\mathbf{u}_N^n(\boldsymbol{\mu}), \qquad \mathbf{p}^n(\boldsymbol{\mu}) \approx \mathbf{V}_{N_p}\mathbf{p}_N^n(\boldsymbol{\mu}) \tag{4.20}$$

where $\mathbf{V}_{N_u} \in \mathbb{R}^{N_h^u \times N_u}$ and $\mathbf{V}_{N_p} \in \mathbb{R}^{N_h^p \times N_p}$ 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.

Substituting (4.20) into (4.14) 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

$$\mathbf{N}_N(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) \begin{bmatrix} \mathbf{u}_N^{n+1}(\boldsymbol{\mu}) \\ \mathbf{p}_N^{n+1}(\boldsymbol{\mu}) \end{bmatrix} = \mathbf{g}_N^{n+1}(\boldsymbol{\mu}). \tag{4.21}$$

The RB arrays $\mathbf{N}_N(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) \in \mathbb{R}^{N \times N}$ and $\mathbf{g}_N^{n+1}(\boldsymbol{\mu}) \in \mathbb{R}^N$ are obtained by projecting onto the RB spaces $\mathbf{V}_{N_u}$ and $\mathbf{V}_{N_p}$ the corresponding blocks defined in (4.15); in other words, they can be obtained as

$$\mathbf{N}_N(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) = \begin{bmatrix} \dfrac{\alpha_1}{\Delta t}\mathbf{M}_N^u(\boldsymbol{\mu}) + \mathbf{D}_N(\boldsymbol{\mu}) + \mathbf{C}_N(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) & \mathbf{B}_N^T(\boldsymbol{\mu}) \\ \mathbf{B}_N(\boldsymbol{\mu}) & 0 \end{bmatrix}, \tag{4.22}$$

$$\mathbf{g}_N^{n+1}(\boldsymbol{\mu}) = \begin{bmatrix} \dfrac{1}{\Delta t}\mathbf{M}_N^u(\boldsymbol{\mu})\mathbf{u}_N^{n,\sigma}(\boldsymbol{\mu}) + \mathbf{f}_{N1}^{n+1}(\boldsymbol{\mu}) \\ \mathbf{f}_{N2}^{n+1}(\boldsymbol{\mu}) \end{bmatrix}, \tag{4.23}$$

where

$$\begin{aligned} &\mathbf{D}_N(\boldsymbol{\mu}) = \mathbf{V}_{N_u}^T\mathbf{D}(\boldsymbol{\mu})\mathbf{V}_{N_u}, \qquad \mathbf{M}_N^u(\boldsymbol{\mu}) = \mathbf{V}_{N_u}^T\mathbf{M}^u(\boldsymbol{\mu})\mathbf{V}_{N_u}, \\ &\mathbf{B}_N(\boldsymbol{\mu}) = \mathbf{V}_{N_p}^T\mathbf{B}(\boldsymbol{\mu})\mathbf{V}_{N_u} \end{aligned} \tag{4.24}$$

and

$$\mathbf{f}_{N1}^{n+1}(\boldsymbol{\mu}) = \mathbf{V}_{N_u}^T\mathbf{f}_1^{n+1}(\boldsymbol{\mu}), \qquad \mathbf{f}_{N2}^{n+1}(\boldsymbol{\mu}) = \mathbf{V}_{N_p}^T\mathbf{f}_2^{n+1}(\boldsymbol{\mu}), \qquad \mathbf{u}_{N,0} = \mathbf{V}_{N_u}^T\mathbf{u}_0.$$

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

$$\mathbf{C}_N(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) = \mathbf{V}_{N_u}^T\mathbf{C}(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})\mathbf{V}_{N_u}. \tag{4.25}$$

Remark 4.2.1. *An alternative to the Galerkin-RB formulation is the Petrov Galerkin method. Such option is particularly convenient when dealing with turbulent flows, since it allows to obtain a properly well-posed reduced problem in terms of long-term stability for highly nonlinear dynamical systems, see e.g. [Carlberg et al., 2013]; however, for the regimes we are interested in, there is not such an issue, and a Galerkin approach represents a reliable option.*

Remark 4.2.2. *Throughout this chapter, we will use the matrices* $\mathbf{X}_u \in \mathbb{R}^{N_h^u \times N_h^u}$ *and* $\mathbf{X}_p \in \mathbb{R}^{N_h^p \times N_h^p}$*, as introduced in (3.12), which algebraically encode the scalar products* $(\cdot,\cdot)_V$ *and* $(\cdot,\cdot)_Q$ *over the velocity and pressure space, respectively.*

4.2.1 Basis construction: double POD strategy

To construct the reduced basis matrices $\mathbf{V}_{N_u}$ and $\mathbf{V}_{N_p}$ we use POD. This requires to collect snapshots of the FE 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 (4.14); then POD is performed separately on velocity and pressure snapshots. This procedure would in principle lead to either the SVD of very large snapshot matrices (of size $n_s N_t \times N_h^u$ and $n_s N_t \times N_h^p$ for velocity and pressure, respectively), or to an eigenproblem for two correlation matrices of size $n_s N_t \times n_s N_t$. As a matter of fact, the computation of its eigenvalues entails a very demanding amount of work.

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 a POD with respect to the time trajectory (for a fixed $\boldsymbol{\mu}$) and secondly collecting together this information to perform a POD with respect to the parametric dependence. This procedure is done in the following steps:

1. for each $\boldsymbol{\mu}_i$, $i = 1, \dots, n_s$, we compute the solution of (4.14) for $n = 0, \dots, N_t - 1$ and then 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 $\mathbf{S}_{\vec{u}}^i$ (resp. $\mathbf{S}_p^i$). Then, we perform a SVD on its time trajectory, that is, for any $i = 1, \dots, n_s$, we compute

$$\mathbf{S}_{\vec{u}}^i = [\mathbf{u}^1(\boldsymbol{\mu}_i), \dots, \mathbf{u}^{N_t}(\boldsymbol{\mu}_i)] \qquad \mathbf{V}_{N_u}^i = \mathrm{POD}\Big(\mathbf{S}_{\vec{u}}^i, \mathbf{X}_u, \varepsilon_t\Big), \tag{4.26}$$

$$\mathbf{S}_p^i = [\mathbf{p}^1(\boldsymbol{\mu}_i), \dots, \mathbf{p}^{N_t}(\boldsymbol{\mu}_i)] \qquad \mathbf{V}_{N_p}^i = \mathrm{POD}\Big(\mathbf{S}_p^i, \mathbf{X}_p, \varepsilon_t\Big); \tag{4.27}$$

   hereon, we refer to this step as *POD in time*.

2. we collect all the basis functions produced by the n_s PODs in time and perform a final POD (*POD in parameter*) of the matrix whose columns are the retained basis functions, that is,

$$\mathbf{S}_{\vec{u}} = [\mathbf{V}_{N_u}^1, \dots, \mathbf{V}_{N_u}^{n_s}] \qquad \mathbf{V}_{N_u} = \mathrm{POD}\Big(\mathbf{S}_{\vec{u}}, \mathbf{X}_u, \varepsilon_\mu\Big) \tag{4.28}$$

$$\mathbf{S}_p = [\mathbf{V}_{N_p}^1, \dots, \mathbf{V}_{N_p}^{n_s}] \qquad \mathbf{V}_{N_p} = \mathrm{POD}\Big(\mathbf{S}_p, \mathbf{X}_p, \varepsilon_\mu\Big) \tag{4.29}$$

The tolerances $\varepsilon_t > 0$ and $\varepsilon_\mu > 0$ are used as stopping criteria (based on the discarded singular values) for the POD in time and in parameter, respectively, and are chosen such that $\varepsilon_\mu \geq \varepsilon_t$, in order to guarantee that the POD in parameter is based on a proper sampling in time.

4.2.2 ROM Stability

In (4.21) we considered a Galerkin projection onto the POD basis to obtain a well-posed RB approximation. However, similarly to the Stokes case of Section 3.2.1, this 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) of the RB problem, thus yielding a potentially singular matrix $\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$. This issue can be overcome by enriching the velocity space by so-called supremizer functions, according to the strategy for the Galerkin RB approximation of the Stokes problem outlined in Section 3.2.1. We recall that the pressure supremizing operator, already introduced in (3.21), is such that, for any given $q_h \in Q_h$, $T_p(q_h; \boldsymbol{\mu})$ returns the solution of the following variational problem

$$(T_p(q_h; \boldsymbol{\mu}), \vec{v}_h)_{V(\boldsymbol{\mu})} = b(\vec{v}_h, q_h; \boldsymbol{\mu}) \qquad \forall \vec{v}_h \in V_h(\boldsymbol{\mu}). \tag{4.30}$$

In particular, an *approximate supremizer* option is pursued, in order the velocity space not to be $\boldsymbol{\mu}$-dependent. Moreover, for the case at hand, the supremizers must take into account also time dependence. We essentially extend the procedure explored in [Ballarin et al., 2015] 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 $n = 1, \dots, N_t$ we compute the supremizers, by solving

$$\mathbf{X}_u(\boldsymbol{\mu}_i)\mathbf{t}_p^n(\boldsymbol{\mu}_i) = \mathbf{B}^T(\boldsymbol{\mu}_i)\mathbf{p}^n(\boldsymbol{\mu}_i), \tag{4.31}$$

   we collect them in $\mathbf{S}^i_{\vec{t}} \in \mathbb{R}^{N_h^u \times n_s}$ and compress them by performing POD in time

$$\mathbf{S}^i_{\vec{t}} = [\mathbf{t}_p^1(\boldsymbol{\mu}_i), \dots, \mathbf{t}_p^{N_t}(\boldsymbol{\mu}_i)] \qquad \mathbf{V}^i_{N_s} = \mathrm{POD}\Big(\mathbf{S}^i_{\vec{t}}, \mathbf{X}_u, \varepsilon_t\Big); \tag{4.32}$$

2. we generate a global snapshot matrix and perform a POD in parameter to obtain an enriching basis $\mathbf{V}_{N_s} \in \mathbb{R}^{N_h^u \times N_s}$

$$\mathbf{S}_{\vec{t}} = [\mathbf{V}^1_{N_s}, \dots, \mathbf{V}^{n_s}_{N_s}] \qquad \mathbf{V}_{N_s} = \mathrm{POD}\Big(\mathbf{S}_{\vec{t}}, \mathbf{X}_u, \varepsilon_\mu\Big); \tag{4.33}$$

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

$$\mathbf{V}_{N_u} = \text{G-S}([\mathbf{V}_{N_u}, \mathbf{V}_{N_s}], \mathbf{X}_u). \tag{4.34}$$

Remark 4.2.3. *An enriching strategy to recover the well-posedness of the RB problem is necessary because our starting high-fidelity model* (4.14) *is not strongly coercive; if, on the other hand, we had employed a SUPG stabilization formulation for the FE problem, we would not need any velocity enrichment, see e.g. [Negri, 2015], where this option is investigated.*

4.2.3 Enhancing efficiency by hyper-reduction

Because of the $\boldsymbol{\mu}$-dependence induced by the geometry deformation, all the matrices and vectors appearing in (4.14) depend nonaffinely on the parameter $\boldsymbol{\mu}$; moreover, a critical issue is represented by the RB linearized term $\mathbf{C}_N(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$. In order to construct it we should at first build $\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu})$, which is used to assemble the FE matrix $\mathbf{C}(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$; then this latter must be projected as in (4.25). As a matter of fact, these operations prevent an efficient decoupling between an offline and online stage, since they entail a huge assembly cost for the ROM. We highlight that such a difficulty arises because of the nonaffine geometric dependence, indeed, if an affine parametrization only is considered, the quadratically nonlinear term $\mathbf{C}_N(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$ could be expressed as an sum of N_u affine components, where N_u is the dimension of the RB velocity space.

With the goal of cheaply assembling the ROM problem, we employ here, for the first time in the case of nonlinear unsteady Navier-Stokes equations in parametrized geometries, the Matrix version of 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 procedure to define a set of well-chosen interpolation points.

To start with, MDEIM can be readily employed to compute an approximated affine decomposition of $\mathbf{M}^u(\boldsymbol{\mu})$, $\mathbf{D}(\boldsymbol{\mu})$ and $\mathbf{B}(\boldsymbol{\mu})$, which is used to build the RB affine approximations $\{\mathbf{D}_N^q\}_{q=1}^{Q_d}$, $\{\mathbf{B}_N^q\}_{q=1}^{Q_b}$, $\{\mathbf{M}_N^q\}_{q=1}^{Q_m}$ for the RB matrices, yielding

$$\mathbf{D}_N(\boldsymbol{\mu}) \approx \widetilde{\mathbf{D}}_N(\boldsymbol{\mu}), \qquad \mathbf{B}_N(\boldsymbol{\mu}) \approx \widetilde{\mathbf{B}}_N(\boldsymbol{\mu}), \qquad \mathbf{M}_N^u(\boldsymbol{\mu}) \approx \widetilde{\mathbf{M}}_N^{\vec{u}}(\boldsymbol{\mu}), \tag{4.35}$$

as already done for the Stokes equations (in the case of the matrices $\mathbf{D}_N(\boldsymbol{\mu})$ and $\mathbf{B}_N(\boldsymbol{\mu})$) and the linear parabolic problems (regarding $\mathbf{M}_N^u(\boldsymbol{\mu})$).

The assumption on the inlet condition outlined in (4.4) allows, on the other hand, to uncouple the time and space-parameter contributions in the inlet Dirichlet condition. This is then reflected in the corresponding contribution at the right hand side of (4.14): this latter can indeed be expressed as

$$\mathbf{f}_1^{n+1}(\boldsymbol{\mu}) = w(t_{n+1})\mathbf{f}_1(\boldsymbol{\mu}) \qquad \mathbf{f}_2^{n+1}(\boldsymbol{\mu}) = w(t_{n+1})\mathbf{f}_2(\boldsymbol{\mu}), \tag{4.36}$$

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

$$\begin{aligned}\mathbf{f}_{N1}^{n+1}(\boldsymbol{\mu}) &= w(t_{n+1})\mathbf{V}_{N_u}^T\mathbf{f}_1(\boldsymbol{\mu}) = w(t_{n+1})\mathbf{f}_{N1}(\boldsymbol{\mu}),\\ \mathbf{f}_{N2}^{n+1}(\boldsymbol{\mu}) &= w(t_{n+1})\mathbf{V}_{N_u}^T\mathbf{f}_2(\boldsymbol{\mu}) = w(t_{n+1})\mathbf{f}_{N2}(\boldsymbol{\mu}).\end{aligned}$$

The uncoupling in (4.36) allows to use DEIM to build an affine approximation $\{\mathbf{f}_1^q\}_{q=1}^{Q_f^1}$, $\{\mathbf{f}_2^q\}_{q=1}^{Q_f^2}$ 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 $\{\mathbf{f}_{N1}^q\}_{q=1}^{Q_f^1}$, $\{\mathbf{f}_{N2}^q\}_{q=1}^{Q_f^2}$ for $\mathbf{f}_{N1}(\boldsymbol{\mu})$ and $\mathbf{f}_{N2}(\boldsymbol{\mu})$, respectively, such that

$$\mathbf{f}_{N1}(\boldsymbol{\mu}) \approx \widetilde{\mathbf{f}}_{N1}(\boldsymbol{\mu}), \qquad \mathbf{f}_{N2}(\boldsymbol{\mu}) \approx \widetilde{\mathbf{f}}_{N2}(\boldsymbol{\mu}). \tag{4.37}$$

The linearized term $\mathbf{C}(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$ nonaffinely depends on the parameter $\boldsymbol{\mu}$, however an MDEIM-approximated affine decomposition is not readily computable, due to its dependence on $\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu})$; 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 (4.14) is solved for the parameter instances $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$, the following steps are executed:

1. 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

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

   we call this stage *MDEIM in time*;

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

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

   where the MDEIM algorithm (1.56) is employed (*MDEIM in parameter*);

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

   $$\begin{aligned}\mathbf{C}_N(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) &\approx \widetilde{\mathbf{C}}_N(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\\ &= \sum_{q=1}^{Q_c} \tilde{\Theta}_c^q(\boldsymbol{\mu})\mathbf{V}_{N_u}^T\mathbf{C}^q\mathbf{V}_{N_u} = \sum_{q=1}^{Q_c} \tilde{\Theta}_c^q(\boldsymbol{\mu})\mathbf{C}_N^q,\end{aligned} \tag{4.38}$$

   where the matrices $\mathbf{C}_N^q \in \mathbb{R}^{N_h^u \times N_h^u}$, $q = 1, \dots, Q_c$, 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 matrices $\mathbf{C}_N^q \in \mathbb{R}^{N\times N}$ are parameter independent and can be precomputed and stored in the offline phase.

In the procedure above, ε_C^{loc} and ε_C are the tolerances used to stop the modes selection for the POD in time (for each $k = 1, \dots, n_s$) and the one in parameter. The final algorithm

Algorithm 13 Offline construction NS-HROM

```
procedure NS-RB-OFFLINE($\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}, \varepsilon_t, \varepsilon_\mu, \varepsilon_C^{loc}, \varepsilon_C$, $\delta_{\mathrm{MDEIM}}$, $\delta_{\mathrm{DEIM}}$)
    Use MDEIM to compute an affine decomposition of $\mathbf{D}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$, $\mathbf{M}^u(\boldsymbol{\mu})$
    Use DEIM to compute an affine decomposition of $\mathbf{f}_1(\boldsymbol{\mu})$, $\mathbf{f}_2(\boldsymbol{\mu})$
    for i = 1 : $n_s$ do
        Compute $\{\mathbf{u}^n(\boldsymbol{\mu}_i)\}_{n=1}^{N_t}$, $\{\mathbf{p}^n(\boldsymbol{\mu}_i)\}_{n=1}^{N_t}$, $\{\mathbf{t}_p^n(\boldsymbol{\mu}_i)\}_{n=1}^{N_t}$
        Set $\mathbf{S}_{\vec{u}}^i = [\mathbf{u}^1(\boldsymbol{\mu}_i), \dots, \mathbf{u}^{N_t}(\boldsymbol{\mu}_i)]$ and $\mathbf{V}_{N_u}^i = \mathrm{POD}(\mathbf{S}_{\vec{u}}^i, \mathbf{X}_u, \varepsilon_t)$
        Set $\mathbf{S}_p^i = [\mathbf{p}^1(\boldsymbol{\mu}_i), \dots, \mathbf{p}^{N_t}(\boldsymbol{\mu}_i)]$ and $\mathbf{V}_{N_p}^i = \mathrm{POD}(\mathbf{S}_p^i, \mathbf{X}_p, \varepsilon_t)$
        Set $\mathbf{S}_{\vec{t}}^i = [\mathbf{t}_p^1(\boldsymbol{\mu}_i), \dots, \mathbf{t}_p^{N_t}(\boldsymbol{\mu}_i)]$ and $\mathbf{V}_{N_s}^i = \mathrm{POD}(\mathbf{S}_{\vec{t}}^i, \mathbf{X}_u, \varepsilon_t)$
        Set $\mathbf{S}_C^i = [\mathrm{vec}(\mathbf{C}(\mathbf{u}^{0,*}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i)), \dots, \mathrm{vec}(\mathbf{C}(\mathbf{u}^{N_t-1,*}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i))]$
            and $\mathbf{V}_C^i = \mathrm{POD}(\mathbf{S}_C^i, \mathbf{I}_{(N_h^u)^2}, \varepsilon_C^{loc})$
    end for
    Set $\mathbf{S}_{\vec{u}} = [\mathbf{V}_{N_u}^1, \dots, \mathbf{V}_{N_u}^{n_s}]$ and $\mathbf{V}_{N_u} = \mathrm{POD}(\mathbf{S}_{\vec{u}}, \mathbf{X}_u, \varepsilon_\mu)$
    Set $\mathbf{S}_p = [\mathbf{V}_{N_p}^1, \dots, \mathbf{V}_{N_p}^{n_s}]$ and $\mathbf{V}_{N_u} = \mathrm{POD}(\mathbf{S}_p, \mathbf{X}_p, \varepsilon_\mu)$
    Set $\mathbf{S}_{\vec{t}} = [\mathbf{V}_{N_s}^1, \dots, \mathbf{V}_{N_s}^{n_s}]$ and $\mathbf{V}_{N_s} = \mathrm{POD}(\mathbf{S}_{\vec{t}}, \mathbf{X}_u, \varepsilon_\mu)$
    Set $\mathbf{S}_C = [\mathbf{V}_C^1, \dots, \mathbf{V}_C^{n_s}]$ and $\mathbf{V}_C = \mathrm{MDEIM}(\mathbf{S}_C, \varepsilon_C)$
    Orthonormalize: $\mathbf{V}_{N_u} = \text{G-S}(\mathbf{V}_{N_u}, \mathbf{V}_{N_s}, \mathbf{X}_u)$
    Precompute and store the (approximated) RB affine decompositions:
    $\{\mathbf{D}_N^q\}_{q=1}^{Q_d}$, $\{\mathbf{B}_N^q\}_{q=1}^{Q_b}$, $\{\mathbf{M}_N^q\}_{q=1}^{Q_m}$, $\{\mathbf{C}_N^q\}_{q=1}^{Q_c}$, $\{\mathbf{f}_{N1}^q\}_{q=1}^{Q_f^1}$, $\{\mathbf{f}_{N2}^q\}_{q=1}^{Q_f^2}$.
end procedure
```

involving the construction of the NS-HROM for the parametrized sequence of algebraic system (4.14) is outlined in Algorithm 13.

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

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

where $\widetilde{\mathbf{N}}_N(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$ features the same saddle-point structure as the matrix in (4.22), but involves the approximated affine matrices $\widetilde{\mathbf{D}}_N(\boldsymbol{\mu})$, $\mathbf{B}_N(\boldsymbol{\mu})$, $\mathbf{M}_N^u(\boldsymbol{\mu})$ and $\widetilde{\mathbf{C}}_N(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$; similarly the DEIM-approximated RB vectors (4.37) are employed for the cheap assembly of the right hand side $\widetilde{\mathbf{g}}_N^{n+1}(\boldsymbol{\mu})$.

4.3 Sequential ROMs for deformation and fluid flows

When dealing with fluid dynamics in deformed domains arising from engineering applications, domain displacement is seldom expressed by an analytical function; indeed, often the deformation results from an additional FE problem either describing the behavior of the structure with respect to given inputs or an harmonic extension of boundary data. In the context of parametrized problems, these inputs typically depend on the underlying parametrization and thus vary when considering different instances of the parameter. Hence, when aiming at solving the NS system (4.14) for any new instance of the geometric parameters, the FE problem describing the deformation must first be solved. More specifically, we set

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

where $\vec{d}(\boldsymbol{\mu})$ is the solution of a variational problem: find $\vec{d}(\boldsymbol{\mu}) \in V_d$, such that

$$a_d(\vec{d}(\boldsymbol{\mu}), \vec{w}; \boldsymbol{\mu}) = f_d(\vec{w}, \boldsymbol{\mu}), \qquad \forall \vec{w} \in V_d \tag{4.39}$$

where V_d is a suitable Hilbert space. Problem (4.39) arises, for instance, when an harmonic or solid extension is considered to extend a boundary data to the whole fluid domain, see e.g. [Staten et al., 2011, Baker, 2002, Stein et al., 2004]. When using the FE method, as described in Section 1.1.1, problem (4.39) yields the linear system

$$\mathbf{A}^d(\boldsymbol{\mu})\mathbf{d}(\boldsymbol{\mu}) = \mathbf{f}^d(\boldsymbol{\mu}), \tag{4.40}$$

where $\mathbf{A}^d(\boldsymbol{\mu}) \in \mathbb{R}^{N_h^d \times N_h^d}$ is the FE matrix obtained 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 $f_d(\cdot, \boldsymbol{\mu})$.

Additionally, one should take into account the effort to obtain a lifting function, which in nontrivial geometries entails a third FE linear system to be solved. Indeed, the following problem is considered to compute the lifting function needed to build the right hand side of (4.13): find $\vec{l}(\boldsymbol{\mu}) \in V_l$, such that

$$a_l(\vec{l}(\boldsymbol{\mu}), \vec{w}; \boldsymbol{\mu}) = f_l(\vec{w}, \boldsymbol{\mu}), \qquad \forall \vec{w} \in V_l, \tag{4.41}$$

where $V_l = V_l(\boldsymbol{\mu})$ is a proper Hilbert space and $\vec{l}(\boldsymbol{\mu})$ is the lifting function. In the same way as for (4.39), problem (4.41) is discretized with the FE element method yielding the linear system

$$\mathbf{A}^l(\boldsymbol{\mu})\mathbf{l}(\boldsymbol{\mu}) = \mathbf{f}^l(\boldsymbol{\mu}), \tag{4.42}$$

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 a large computational grid is considered, the solution of (4.40) and (4.42) entails the use of a proper FE solver when each instance of the parameter $\boldsymbol{\mu}$ is considered for the NS equations (4.14). On the resulting domain, then one can solve the fluid flow problem.

When performing the offline phase of the ROM for the fluid problem (Algorithm 13), problems (4.40) and (4.42) must be solved for each $\boldsymbol{\mu}_i$, $i = 1, \dots, n_s$. The same occurs during the online phase when the NS-ROM is employed to solve the problem for new instances of the parameter and, even though the fluid dynamics problem is typically the most demanding one, the computational costs entailed by problems (4.40) and (4.42) may hamper the overall efficiency of the method.

In the following, we devise a sequential hyper-reduced order model (HROM) construction where the RB method is employed to cheaply compute an approximation of the displacement $\vec{d}(\boldsymbol{\mu})$ and the lifting function $\vec{l}(\boldsymbol{\mu})$, which are then used to feed the construction of the NS-HROM during the offline phase and its use during the online phase.

Toward this goal, we follow the procedure outlined in Section 1.3 to construct two HROMs to compute the RB approximation of $\mathbf{d}(\boldsymbol{\mu})$ and $\mathbf{l}(\boldsymbol{\mu})$. Given a set of offline parameters $\{\boldsymbol{\mu}_i^d\}_{i=1}^{n_s^d}$, we build by POD a N_d-dimensional RB projection matrix $\mathbf{V}^d \in \mathbb{R}^{N_h^d \times N_d}$ for approximating the deformation and MDEIM and DEIM to construct the affine decomposition of $\mathbf{A}^d(\boldsymbol{\mu})$, $\mathbf{f}^d(\boldsymbol{\mu})$, respectively. Then $\mathbf{V}^d \mathbf{d}_N(\boldsymbol{\mu}) \approx \mathbf{d}(\boldsymbol{\mu})$ is obtained by solving the RB problem

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

for $\mathbf{d}_N(\boldsymbol{\mu}) \in \mathbb{R}^{N_d}$ instead of (4.40); $\widetilde{\mathbf{A}}_N^d(\boldsymbol{\mu}) \in \mathbb{R}^{N_d \times N_d}$ and $\widetilde{\mathbf{f}^d}_N(\boldsymbol{\mu}) \in \mathbb{R}^{N_d}$ are obtained by Galerkin projection onto the subspace spanned by the columns of $\mathbf{V}^d$.

Similarly, given a set of offline parameters $\{\boldsymbol{\mu}_i^l\}_{i=1}^{n_s^l}$ (possibly different from the ones considered for the deformation problem), we build by POD a N_l-dimensional RB projection matrix $\mathbf{V}^l$ for approximating the lifting function and MDEIM and DEIM to construct the affine decomposition of $\mathbf{A}^l(\boldsymbol{\mu})$, $\mathbf{f}^l(\boldsymbol{\mu})$, respectively. The RB problem (D-HROM) is used in view of the computation of the lifting functions $\{\mathbf{l}(\boldsymbol{\mu}_i^l)\}_{i=1}^{n_s^l}$, which are used to build the projection matrix $\mathbf{V}^l$ by POD, and of the arrays $\mathbf{A}^l(\boldsymbol{\mu}_i^l)$, $\mathbf{f}^l(\boldsymbol{\mu}_i^l)$, which are employed to compute a (M)DEIM affine decompositions. Finally, problem (4.42) is substituted by

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

where $\mathbf{l}_N(\boldsymbol{\mu}) \in \mathbb{R}^{N_l}$ is the RB solution, such that $\mathbf{l}(\boldsymbol{\mu}) \approx \mathbf{V}^l \mathbf{l}_N(\boldsymbol{\mu})$ and $\widetilde{\mathbf{A}}_N^l(\boldsymbol{\mu}) \in \mathbb{R}^{N_l \times N_l}$ and $\widetilde{\mathbf{f}^l}_N(\boldsymbol{\mu}) \in \mathbb{R}^{N_l}$ are obtained by Galerkin projection onto $\mathbf{V}^l$.

Problems (D-HROM) and (L-HROM) are then used in the construction of NS-HROM in Algorithm 13. The complete procedure is outlined in Figure 4.1.

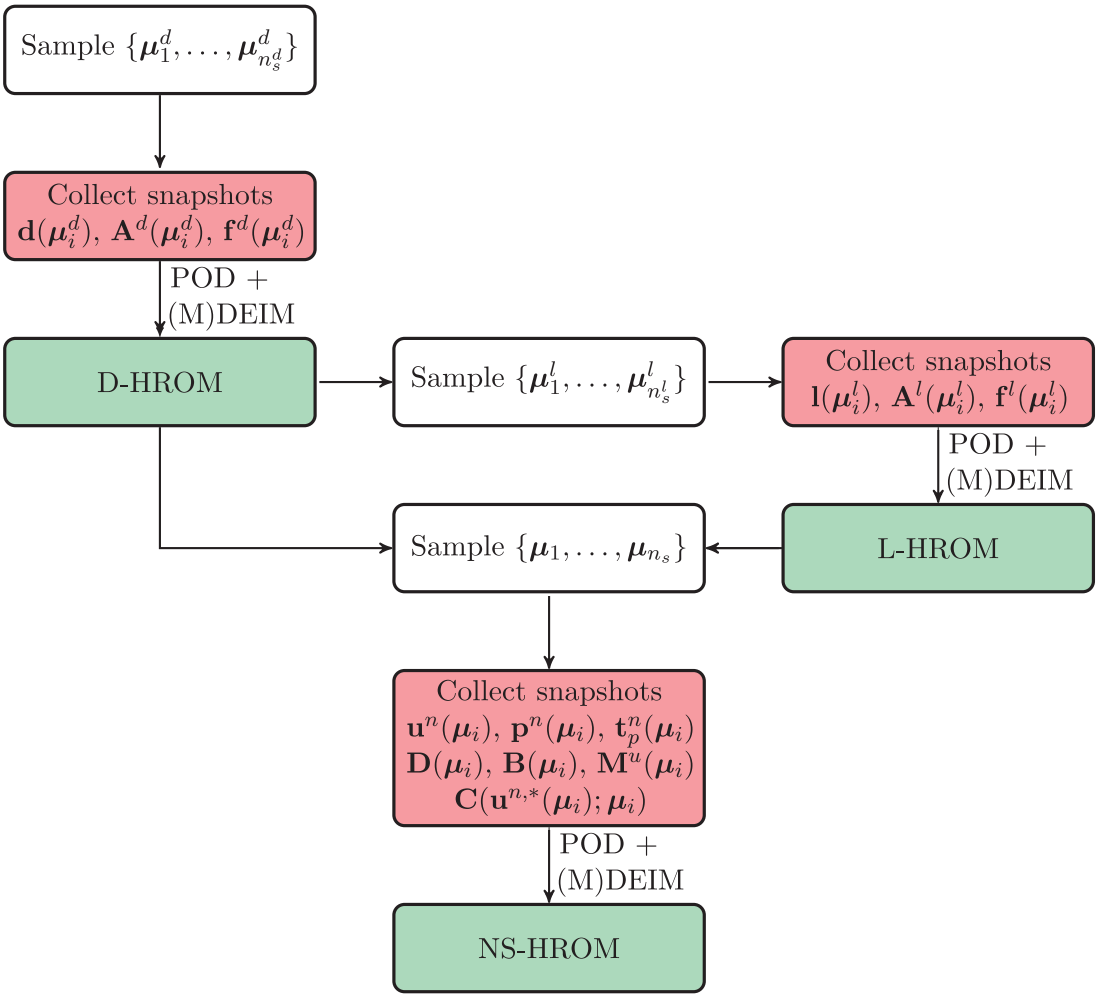

Sample $\{\boldsymbol{\mu}_1^d, \dots, \boldsymbol{\mu}_{n_s^d}^d\}$
Collect snapshots
$\mathbf{d}(\boldsymbol{\mu}_i^d)$, $\mathbf{A}^d(\boldsymbol{\mu}_i^d)$, $\mathbf{f}^d(\boldsymbol{\mu}_i^d)$
POD +
(M)DEIM
D-HROM
Sample $\{\boldsymbol{\mu}_1^l, \dots, \boldsymbol{\mu}_{n_s^l}^l\}$
Collect snapshots
$\mathbf{l}(\boldsymbol{\mu}_i^l)$, $\mathbf{A}^l(\boldsymbol{\mu}_i^l)$, $\mathbf{f}^l(\boldsymbol{\mu}_i^l)$
POD +
(M)DEIM
L-HROM
Sample $\{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{n_s}\}$
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 4.1 – Offline strategy to build NS-HROM. Red blocks specify when the collection of solution, matrix and right hand side snapshots is performed, green blocks specify when a HROM is constructed and then used to boost the offline phase.

4.4 Numerical results for NS-HROM

We consider in this section the solution of the NS problem (4.1), and set $\nu = 0.01$, the time interval $(0, 0.5)$ and use the geometrical configuration already employed in the numerical experiments in Section 3.4.2.

Test case setting

We consider the cylindrical reference domain

$$\Omega^0 = \{\vec{x} \in \mathbb{R}^3 : \; x_1^2 + x_1^2 < 0.25, \, x_3 \in (0, 5)\}$$

and generate the displacement $\vec{d}(\boldsymbol{\mu})$ as the harmonic extension of a boundary deformation; this yields the $\boldsymbol{\mu}$-dependent domain

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

Here $\vec{d}(\boldsymbol{\mu})$ solves the (vector) Laplace equation (3.51), where we define

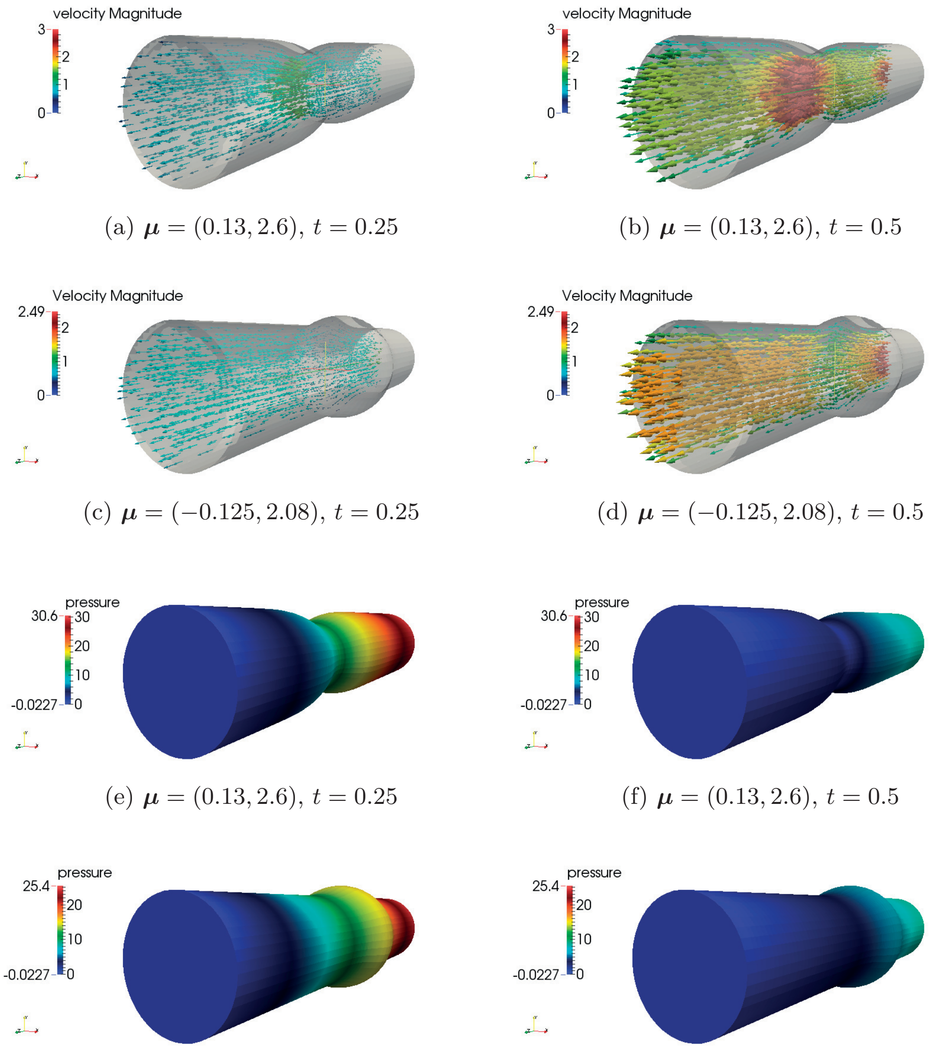
$$\vec{h}(\boldsymbol{\mu}) = \begin{bmatrix} -x_1\mu_1 \exp\{-5(x_3-\mu_2)^2\} \\ -x_2\mu_1 \exp\{-5(x_3-\mu_2)^2\} \\ 0 \end{bmatrix},$$

thus entailing a deformation of the cylinder which consists in the narrowing or enlargement (according to the sign of μ_1) of its section, in different positions, depending on the value of μ_1. Examples of the solution of (4.14) for different values of the parameter and times are reported in Figure 4.2.

Since the solution $\vec{d}(\boldsymbol{\mu})$ of (3.51) is not known a-priori, we compute its numerical approximation $\vec{d}_h(\boldsymbol{\mu})$ by employing the FE method (with $\mathcal{P}_2$ FE basis functions), yielding problem (4.40). Notice that by considering the special case (3.51), the FE matrix $\mathbf{A}^d(\boldsymbol{\mu})$ does not depend on the parameter.

Once the computational domain has been deformed, the lifting function is computed by solving problem (3.52), where $\vec{g}_D(\boldsymbol{\mu})$ is the one defined in (4.4) and such that the velocity vanishes on the wall and the flow rate at the inlet is equal to 1; regarding the time dependent contribution, we take $w(t) = \sin(2\pi t)$. Problem (3.52) as well is discretized with the FE method with second order polynomials ($\mathcal{P}_2$) basis functions, leading to the parametrized linear system (4.42). We take as parameters the coefficients μ_1 and μ_2, that is $\boldsymbol{\mu} = (\mu_1, \mu_2) \in [-0.3, 0.3] \times [2, 3]$, which yield a parametrization affecting the deformation problem, the computation of the lifting and ultimately the fluid flow.

We use a computational domain with 13’603 vertices, and discretize the NS equations using Taylor-Hood FE spaces, that is with $\mathcal{P}_2 - \mathcal{P}_1$ FE basis functions leading to $N_h^u = 306'735$ and $N_h^p = 13'603$ degrees of freedom for velocity and pressure, respectively, leading to a total dimension of the FE problem of $N_h = 320'338$. We employ the BDF2 method with $\Delta t = 0.01$ for the time discretization. The FE problem (4.14) is solved with FGMRES preconditioned with a SIMPLE preconditioner, where the solves (steps 1 and 2 in Algorithm 12) are carried out by inner iterations up to a tolerance of 10^{-5} using an Additive Schwarz preconditioner from the `Ifpack` package of `Trilinos`. The FE solver takes on average 0.55 and 0.41 seconds for the deformation and lifting problem respectively solved with an AMG-preconditioned GMRES and with a stopping criterion of 10^{-9} on the FE residual rescaled with the Euclidean norm of the right hand side.



(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 4.2 – 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).

Offline phase

The offline phase is divided in three stages for the subsequent construction of the HROM for the domain deformation, the lifting function and the fluid flow. We report in the following the results for the first two steps.

1. We first construct an HROM for the deformation (D-HROM); the results are reported in Table 4.1. 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 parameter independent); the singular value decompositions (SVDs) corresponding to the

Table 4.1 – 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

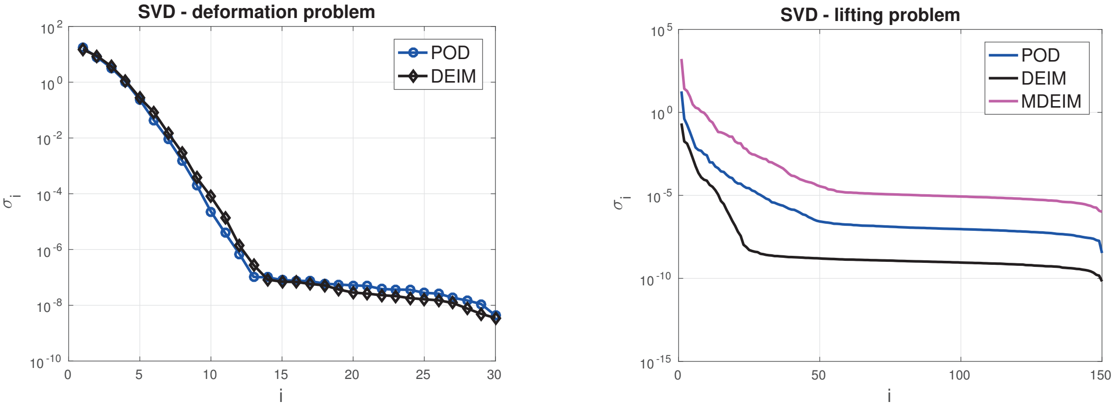
Table 4.2 – L-HROM: POD for state reduction and (M)DEIM have been run with $\varepsilon_{\text{POD}} = \delta_{\text{DEIM}} = \delta_{\text{MDEIM}} = 10^{-7}$. Computational times are expressed in seconds.

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

construction of the RB matrix $\mathbf{V}^d$ and the DEIM affine approximation are reported in Figure 4.3a. By plugging in 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; in Figure 4.3a the singular value 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 (3.51), as one should expect for such linear elliptic problem. The offline phase is $t_{\text{off}} = 51.17$ seconds long; by testing online the HROM for the deformation on 50 instances of the parameter, we obtain an average FE residual $r_{\text{RB}} = 5.5 \cdot 10^{-7}$, with a solution computed in 0.026 seconds on average. 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 (cf. Figure 4.1).

2. We then construct an HROM for the lifting function (L-HROM), which is fed with the approximated deformation computed at step 1; the corresponding results are reported in Table 4.2. 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 4.3b. By plugging in 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 is $t_{\text{off}} = 89.13$ seconds long. By testing online the HROM for the lifting on 50 instances of the parameter, we obtain an average FE residual $r_{\text{RB}} = 7.3 \cdot 10^{-6}$, with a solution computed in 0.03 seconds on average. Similarly to the previous case, the resulting ROM also leads to a speed up of about 14 with respect to the solution of the FE linear system.

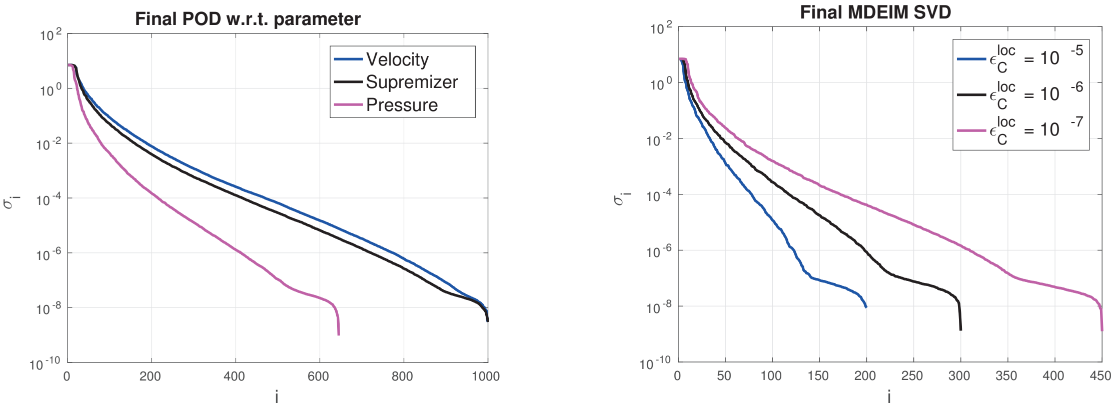
The third step consists in building the HROM for the NS equations on top of the previous two, which represents the most demanding stage of the offline phase. To this aim, we employ $n_s = 50$ snapshots for the state reduction, keeping fixed the tolerance



(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 4.3 – SVDs for building D-HROM and L-HROM problems.



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

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

Figure 4.4 – SVDs for building final RB spaces (left) and final MDEIM of the linearized term (right). Notice that in the latter, according to ε_C^{loc} of the MDEIMs in time, the number of snapshots and the decay of the resulting SVD for the final MDEIM change.

Table 4.3 – Chosen settings for numerical experiments of NS-HROM.

Setting	ε_t	ε_μ	$\delta_{\mathrm{DEIM}} = \delta_{\mathrm{MDEIM}} = \varepsilon_C^{loc}$	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}

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 $\mathbf{V}^i_{N_s}$, $i = 1, \dots, n_s$ and the final $\mathbf{V}_{N_s}$, 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. Having 50 offline parameters, the final RB matrices $\mathbf{V}_{N_u}, \mathbf{V}_{N_p}, \mathbf{V}_{N_s}$ are built with POD starting from 1000, 1000 and 645 snapshots, respectively, and their dimensions depend on the value of ε_μ.

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. In Table 4.3 a summary of the considered combinations of these tolerances are reported; 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 $\mathbf{C}_N(\mathbf{V}_{N_u}\mathbf{u}_N^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$.

In Table 4.4 the results of the offline phase obtained with the settings reported in Table 4.3 are reported. In particular, N_u, N_s, N_p ia the number of RB functions retained by the PODs for velocity, enriching velocities and pressure, respectively; in Figure 4.4a the corresponding SVDs are shown instead: as a matter of fact, 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, the final dimension of $\mathbf{V}_{N_s}$ is larger than the one of $\mathbf{V}_{N_u}$. These functions are then merged with a Gram-Schmidt 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 from the MDEIM in time Q_c^t and from 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. By decreasing the employed tolerances, the offline time increases accordingly, since it entails the computation of the RB affine basis for the matrices and right hand sides (cf. step 17-18 of Algorithm 13), however we remark that the most demanding effort is that of the NS FE problem for building the fluid flow snapshots.

Table 4.4 – Results of the offline phase for settings defined in Table 4.3. 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

Table 4.5 – Results averaged on 50 instances of the parameter considered online. Times are reported in seconds.

Setting	$\bar{e}^u_{\mathrm{RB}}$	$\bar{e}^p_{\mathrm{RB}}$	$t^{\mathrm{onl}}_{\mathrm{RB}}$	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

Online phase

During the online phase we use the four ROMs defined in Table 4.3 on 50 online parameters. In order to assess the quality of the RB approximation, we define

$$e^u_{\mathrm{RB}}(\boldsymbol{\mu}) = \sqrt{\frac{\sum_{n=1}^{N_t} \|\mathbf{u}^n(\boldsymbol{\mu}) - \mathbf{V}_{N_u}\mathbf{u}^n_N\|^2_{\mathbf{X}_u}}{\sum_{n=1}^{N_t} \|\mathbf{u}^n(\boldsymbol{\mu})\|^2_{\mathbf{X}_u}}}, \tag{4.43}$$

as the velocity relative error on the time interval $[0, T]$ and similarly we introduce

$$e^p_{\mathrm{RB}}(\boldsymbol{\mu}) = \sqrt{\frac{\sum_{n=1}^{N_t} \|\mathbf{p}^n(\boldsymbol{\mu}) - \mathbf{V}_{N_p}\mathbf{p}^n_N\|^2_{\mathbf{X}_p}}{\sum_{n=1}^{N_t} \|\mathbf{p}^n(\boldsymbol{\mu})\|^2_{\mathbf{X}_p}}}, \tag{4.44}$$

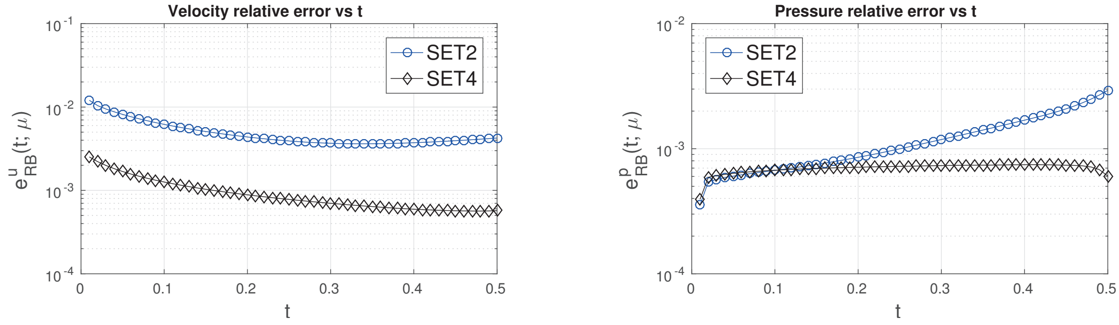
as equivalent for the pressure.

We denote by $\bar{e}^u_{\mathrm{RB}}, \bar{e}^p_{\mathrm{RB}}$ the errors averaged on the number of parameters considered online, which are reported in Table 4.5, and indicate the accuracy of the RB solution with respect to the FE one. The time $t^{\mathrm{onl}}_{\mathrm{RB}}$ to compute online the RB solution in the online phase and the speedup obtained with respect to the FE simulation are also shown. Let us consider SET 1 and SET 2, where only the tolerance ε^{loc}_C varies; by decreasing it from 10^{-5} to 10^{-7}, there is a significant improvement in the quality of the RB approximation, since the relative errors (4.43)-(4.44) 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 approximation. This fact can also be observed by the singular values of the final MDEIM with respect to the parameter variability where the bases in time have been computed using $\varepsilon^{loc}_C = 10^{-5}, 10^{-6}$ and 10^{-7}; the decay of the singular values significantly changes by considering a smaller ε^{loc}_C. As a matter of fact, the number of affine terms produced is 36 for SET 1 and 69 for SET 2, even though the same tolerance ε_C has been used (cf. Table 4.4). In the next experiments we use $\varepsilon^{loc}_C = 10^{-7}$.

Let us now consider SET 3 and SET 4, where either ε_C or ε_μ is decreased to 10^{-5}; the first option has not a beneficial impact on the solution, since the error keeps constant: the convergence of the error is hampered by the too coarse state reduction. On the other hand, in the latter case, the quality of the RB approximations improves, especially for the velocity. This fact is confirmed by considering the velocity and pressure relative errors

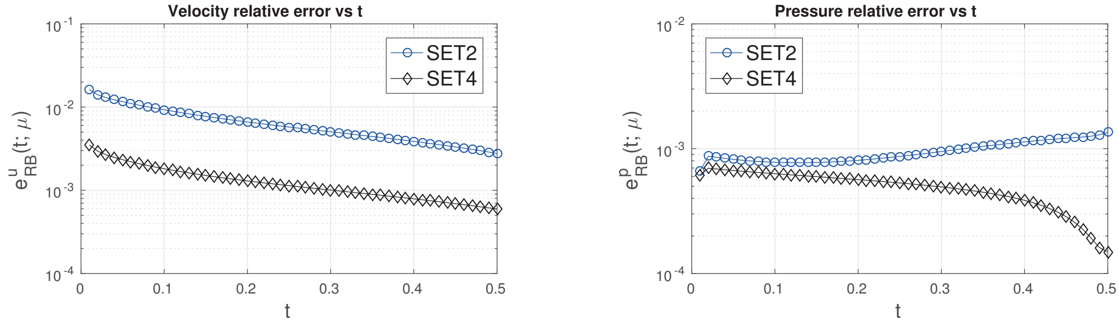
$$e^u_{\mathrm{RB}}(t_n;\boldsymbol{\mu}) = \frac{\|\mathbf{u}^n(\boldsymbol{\mu}) - \mathbf{V}_{N_u}\mathbf{u}^n_N\|^2_{\mathbf{X}_u}}{\|\mathbf{u}^{t_n}(\boldsymbol{\mu})\|^2_{\mathbf{X}_u}} \qquad e^p_{\mathrm{RB}}(t_n;\boldsymbol{\mu}) = \frac{\|\mathbf{p}^n(\boldsymbol{\mu}) - \mathbf{V}_{N_p}\mathbf{p}^n_N\|^2_{\mathbf{X}_u}}{\|\mathbf{p}^n(\boldsymbol{\mu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

as function of the time step t_n, which are reported for two values of the parameters in Figure 4.5 for SET 2 and SET 4. As a matter of fact, the velocity is approximated at every time step with the same discrepancy from the two methods, while the pressure error tends to be similar at the beginning but then follows a different path according to the employed setting. As the time to compute the RB approximation concerns, it ranges from about 4.5 to 32 seconds, with a speed up which varies from 10 to 71 times faster, depending on the accuracy entailed, than the reference FE 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 demonstrated to be a reliable and efficient option to efficiently assembly the RB system, and it will be used in the next sections for developing an efficient MSRB preconditioner for the NS problem (4.14), where we will also consider more involved test cases.



(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 4.5 – Velocity and pressure errors entailed by the NS-HROM as function of time for two values of the parameter.

4.5 MSRB preconditioners for the NS equations

In this section we will tailor the construction of the MSRB preconditioners to the case of unsteady parametrized NS equations, which will exploit the low-rank NS-HROM developed in the previous sections as RB coarse operator. Solving the sequence of linear systems (4.14) raises some critical issues we need to deal with: *i*) the time-dependent nature of the solution, *ii*) the efficient assembling of the linearized nonlinear term appearing in block (1,1) of (4.15), *iii*) the saddle-point nature of the system. To tackle them, we make use of different ingredients introduced above and in the previous chapters. In particular:

1. time dependence is treated as in Section 2.5 for linear parabolic problems, that is by constructing the RB coarse components by accounting for the time variability. We start by considering a MSRB preconditioner version with a single time slab, then we extend it to the multi time slab construction;

2. similarly to the case of nonaffine problems, the use of hyper-reduction techniques to handle linearized terms is not necessary when employing a MSRB preconditioner. However, it can significantly speed up the construction of the RB coarse operators during the online phase; for this reason, we use MDEIM to treat the linearized nonlinear term. Our strategy is inspired by the one developed for the ROM of the NS equations in Section 4.2.3, where a MDEIM basis (with a double POD approach) was constructed for the linearized convective term;

3. as seen for the Stokes equations, the construction of a well-posed RB coarse operator cannot rely on a simple Galerkin RB approximation on the velocity and pressure RB spaces. Hence, we employ an enriched-velocity Galerkin RB formulation for defining the RB coarse operators, similarly to what we have done for the Stokes equations in Section 3.5.1. An enriched G-RB approximation has been preferred to a Petrov Galerkin RB one (such as the one provided by a LSRB for the NS equations) because, as we have seen from the numerical experiments in Section 4.4, the number of affine terms Q_c computed when approximating the linearized term with MDEIM can be very large, and the RB coarse operators obtained with a PG-RB formulation would depend quadratically on Q_c, whereas a G-RB approach only entails a linear dependence.

As usual, we focus our attention on the FGMRES method, however we remark that the framework developed is easily applicable to other iterative methods as well.

4.5.1 Preconditioner construction

The definition of the MSRB preconditioner for problem (4.14) follows (2.10) and (2.53), that is, we can set

$$\mathbf{Q}_{\text{MSRB},k}(\boldsymbol{\mu}) = \mathbf{P}^{-1}(\boldsymbol{\mu}) + \mathbf{Q}_{N_k}(\boldsymbol{\mu})\Big(\mathbf{I}_{N_h} - \mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big), \tag{4.45}$$

where this time $\mathbf{P}(\boldsymbol{\mu})$ is a fine grid nonsingular preconditioner for the NS operator $\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$, and the RB coarse operator $\mathbf{Q}_{N_k}(\boldsymbol{\mu})$ is trained to approximate the solution of

$$\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\mathbf{y}_k^n(\boldsymbol{\mu}) = \Big(\mathbf{I}_{N_h} - \mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\mathbf{P}^{-1}(\boldsymbol{\mu})\Big)\,\mathbf{v}_k^n(\boldsymbol{\mu}), \tag{4.46}$$

which is a FE saddle-point linear system featuring the NS matrix at the left hand side. We introduce the projection matrix $\mathbf{V}_k \in \mathbb{R}^{N_h \times N_k}$, which algebraically represents a N_k-dimensional RB space V_{N_k} spanned by solutions of (4.46) and computed for properly chosen time steps and parameter values. Then, for $k = 1, 2, \dots$, we define the RB matrices

$$\mathbf{N}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) = \mathbf{V}_k^T\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\mathbf{V}_k \tag{4.47}$$

obtained by performing a Galerkin projection of the NS matrix $\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$ onto the subspace spanned by the columns of $\mathbf{V}_k$. Then, we set the RB coarse operator as the RB low-rank solver for (4.46) with respect to $\mathbf{V}_k$, that is,

$$\mathbf{Q}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k\mathbf{N}_{N_k}^{-1}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\mathbf{V}_k^T. \tag{4.48}$$

Since $\mathbf{V}_k$ contains solutions with respect to both time and parameter variability, $\mathbf{Q}_{N_k}(\boldsymbol{\mu})$ as in (4.48) is used as coarse correction of iteration k for all time steps, and yields a RB approximation $\mathbf{y}_{N_k}^n(\boldsymbol{\mu}) \in \mathbb{R}^{N_k}$ of $\mathbf{y}_k^n(\boldsymbol{\mu})$ by solving

$$\mathbf{N}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\mathbf{y}_{N_k}^n(\boldsymbol{\mu}) = \mathbf{V}_k^T\mathbf{v}_{k+\frac{1}{2}}^n(\boldsymbol{\mu}), \tag{4.49}$$

such that $\mathbf{V}_k\mathbf{y}_{N_k}^n(\boldsymbol{\mu}) \approx \mathbf{y}_k^n(\boldsymbol{\mu})$.

As already remarked, problem (4.46) is a NS problem of the same form as (4.14), with a modified right hand side; its solution is made of a velocity and a pressure component, that is, $\mathbf{y}_k^n(\boldsymbol{\mu}) = [\mathbf{y}_{u,k}^n(\boldsymbol{\mu}), \, \mathbf{y}_{p,k}^n(\boldsymbol{\mu})]^T$, which we aim at approximating by relying on a RB coarse operator. With this goal, we introduce two RB spaces, for velocity and pressure approximation, respectively, algebraically represented by two matrices $\mathbf{V}_{uk} \in \mathbb{R}^{N_h^u \times N_k^u}$, $\mathbf{V}_{pk} \in \mathbb{R}^{N_h^p \times N_k^p}$, and constructed by (separately) using POD with solutions of (4.46) for selected values of $\boldsymbol{\mu}$ and time steps. In other words, given a set of parameter values $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$, we set, for $k = 1, 2, \ldots$,

$$\mathbf{V}_{uk} = \mathrm{POD}\Big(\mathbf{S}_{\vec{u}}^{(k)}, \mathbf{X}_u, \delta_{RB,k}\Big), \qquad \mathbf{V}_{pk} = \mathrm{POD}\Big(\mathbf{S}_{p}^{(k)}, \mathbf{X}_p, \delta_{RB,k}\Big), \tag{4.50}$$

where

$$\mathbf{S}_{\vec{u}}^{(k)} = [\mathbf{y}_{u,k}^{1}(\boldsymbol{\mu}_1), \ldots, \mathbf{y}_{u,k}^{N_t}(\boldsymbol{\mu}_1), \ldots, \mathbf{y}_{u,k}^{1}(\boldsymbol{\mu}_{n_s}), \ldots, \mathbf{y}_{u,k}^{N_t}(\boldsymbol{\mu}_{n_s})] \tag{4.51}$$

$$\mathbf{S}_{p}^{(k)} = [\mathbf{y}_{p,k}^{1}(\boldsymbol{\mu}_1), \ldots, \mathbf{y}_{p,k}^{N_t}(\boldsymbol{\mu}_1), \ldots, \mathbf{y}_{p,k}^{1}(\boldsymbol{\mu}_{n_s}), \ldots, \mathbf{y}_{p,k}^{N_t}(\boldsymbol{\mu}_{n_s})] \tag{4.52}$$

and $\delta_{RB,k} > 0$ is the prescribed tolerance. The matrix $\mathbf{V}_{uk}$ (resp. $\mathbf{V}_{pk}$) is then used to approximate $\mathbf{y}_{u,k}^{n}(\boldsymbol{\mu})$ (resp. $\mathbf{y}_{u,k}^{n}(\boldsymbol{\mu})$); however, their simple combination in a Galerkin RB formulation as the one employed in (4.47) does not guarantee the well-posedness of the RB coarse component. As a matter of fact, $\mathbf{V}_{uk}$ and $\mathbf{V}_{pk}$ possibly yield a singular RB matrix $\mathbf{N}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$, since they do not verify an equivalent *inf-sup* condition on the reduced problem, similarly to what explained in Section 4.2.2 for the NS-HROM and in Section 3.5.1 for the Stokes equations. To overcome this issue, we rely on the pressure supremizing operator defined in (4.30) to enrich the velocity spaces $\mathbf{V}_{uk}$, $k = 1, 2, \ldots$ with a proper set of functions collected in the matrix $\mathbf{V}_{sk} \in \mathbb{R}^{N_h^u \times N_k^s}$. More precisely, we consider the pressure error snapshots $\{\mathbf{y}_{p,k}^{n}(\boldsymbol{\mu}_i)\}_{i=1,n=1}^{n_s,N_t}$: for each $n = 1, \ldots, N_t$ and parameter $\boldsymbol{\mu}_i$, $i = 1, \ldots, n_s$, we solve

$$\mathbf{X}_u(\boldsymbol{\mu}_i)\mathbf{y}_{t,k}^{n}(\boldsymbol{\mu}_i) = \mathbf{B}^T(\boldsymbol{\mu}_i)\mathbf{y}_{p,k}^{n}(\boldsymbol{\mu}_i); \tag{4.53}$$

then, we set

$$\begin{aligned}
\mathbf{S}_{\vec{t}}^{(k)} &= [\mathbf{y}_{t,k}^{1}(\boldsymbol{\mu}_1), \ldots, \mathbf{y}_{t,k}^{N_t}(\boldsymbol{\mu}_1), \ldots \mathbf{y}_{t,k}^{1}(\boldsymbol{\mu}_{n_s}), \ldots, \mathbf{y}_{t,k}^{N_t}(\boldsymbol{\mu}_{n_s})], \\
\mathbf{V}_{sk} &= \mathrm{POD}\Big(\mathbf{S}_{\vec{t}}^{(k)}, \mathbf{X}_u, \delta_{RB,k}^{s}\Big).
\end{aligned}$$

The basis $\mathbf{V}_{sk}$ is used to augment the velocity space, with a proper Gram-Schmidt procedure

$$\mathbf{V}_{uk} = \text{G-S}([\mathbf{V}_{uk}, \mathbf{V}_{sk}], \mathbf{X}_u),$$

with respect to the scalar product induced by $\mathbf{X}_u$. Finally, by setting in (4.47)-(4.48)

$$\mathbf{V}_k = \begin{bmatrix} \mathbf{V}_{uk} & 0 \\ 0 & \mathbf{V}_{pk} \end{bmatrix}, \tag{4.54}$$

we obtain a well-posed RB coarse operator, yielding the following RB matrix

$$\mathbf{N}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) = \begin{bmatrix} \dfrac{\alpha_1}{\Delta t}\mathbf{M}_{N_k}^{u}(\boldsymbol{\mu}) + \mathbf{D}_{N_k}(\boldsymbol{\mu}) + \mathbf{C}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu}) & \mathbf{B}_{N_k}^{T}(\boldsymbol{\mu}) \\ \mathbf{B}_{N_k}(\boldsymbol{\mu}) & 0 \end{bmatrix}, \tag{4.55}$$

where the RB matrices $\mathbf{M}_{N_k}^{u}(\boldsymbol{\mu})$, $\mathbf{D}_{N_k}(\boldsymbol{\mu})$, $\mathbf{C}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$ and $\mathbf{B}_{N_k}(\boldsymbol{\mu})$ are defined as in (4.24) and (4.25), provided we replace $\mathbf{V}_{N_u}$ and $\mathbf{V}_{N_p}$ with $\mathbf{V}_{uk}$ and $\mathbf{V}_{pk}$, respectively.

Remark 4.5.1. *Since we are using a Galerkin projection approach, the nonsingularity of the preconditioner operator defined in* (4.45) *is ensured by Theorem 2.2.1.*

In order to speed up the assembly of $\mathbf{N}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$, we can use the (approximated) affine decomposition of the RB matrices in (4.55). The efficient assembly of $\mathbf{M}_{N_k}(\boldsymbol{\mu})$ is treated as in the case of unsteady problems (cf. Section 2.5.1), whereas for $\mathbf{D}_{N_k}(\boldsymbol{\mu})$ and $\mathbf{B}_{N_k}(\boldsymbol{\mu})$ we adopt the strategy outlined for the Stokes problem (cf. Section 3.5.4). On the other hand, we employ the double POD approach devised in in Section 4.2.3 to construct an approximated MDEIM basis for the efficient construction of $\mathbf{C}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$. As a matter of fact, we end up with the affinely approximated saddle-point matrix

$$\mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) \approx \widetilde{\mathbf{N}}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$$

which is used to build the corresponding RB matrix $\widetilde{\mathbf{N}}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$. §This is pursued with a Galerkin projection of $\widetilde{\mathbf{N}}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$ onto $\mathbf{V}_k$, that is

$$\begin{aligned}\widetilde{\mathbf{N}}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) &= \mathbf{V}_k^T \widetilde{\mathbf{N}}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\mathbf{V}_k \\ &= \begin{bmatrix} \dfrac{\alpha_1}{\Delta t}\widetilde{\mathbf{M}}^{\vec{u}}_{N_k}(\boldsymbol{\mu}) + \widetilde{\mathbf{D}}_{N_k}(\boldsymbol{\mu}) + \widetilde{\mathbf{C}}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) & \widetilde{\mathbf{B}}^T_{N_k}(\boldsymbol{\mu}) \\ \widetilde{\mathbf{B}}_{N_k}(\boldsymbol{\mu}) & 0 \end{bmatrix}\end{aligned}$$

where the symbol ~ stands for substitution of the RB matrices with their corresponding MDEIM approximations. Finally, the RB coarse operator is constructed by setting

$$\mathbf{Q}_{N_k}(\boldsymbol{\mu}) = \mathbf{V}_k \left(\widetilde{\mathbf{N}}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\right)^{-1} \mathbf{V}_k^T. \tag{4.56}$$

Similarly to what done for linear parabolic problems in Section 2.5.1, a convenient way to tackle relatively long times is to divide the interval $[0,T]$ in time slabs, such that the memory and computational load entailed by the snapshots and the PODs is not excessive. This is pursued by dividing the interval $[0,T]$ in $\mathcal{S}$ time slabs and by constructing for each of them a sequence of RB spaces and the corresponding coarse operators, as the one defined in (4.56). As a matter of fact, for each time slab s, we end up with L_s RB projection matrices $\mathbf{V}_{k,s}$, $k = 0,\dots,L_s-1$, $s = 0,\dots,\mathcal{S}-1$ of the form

$$\mathbf{Q}_{N_k,s}(\boldsymbol{\mu}) = \mathbf{V}_{k,s} \left(\widetilde{\mathbf{N}}_{N_k,s}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\right)^{-1} \mathbf{V}_{k,s}^T, \tag{4.57}$$

each one tailored for the corresponding time slab, where $\widetilde{\mathbf{N}}_{N_k,s}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})$ is obtained by Galerkin projection onto $\mathbf{V}_{k,s}$, that is

$$\widetilde{\mathbf{N}}_{N_k,s}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu}) = \mathbf{V}_{k,s}^T \mathbf{N}(\mathbf{u}^{n,*}(\boldsymbol{\mu});\boldsymbol{\mu})\mathbf{V}_{k,s}.$$

The local-in-time construction we have carried out by means of a partition in time slabs yields some relevant benefits:

- with a fixed dimension of the RB spaces, it allows to reach a better accuracy, since only a specific part of the time trajectory is employed for building the RB coarse operators;
- a local-in-time MDEIM approximation of the linearized matrix $\widetilde{\mathbf{C}}_{N_k}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$ can be exploited, yielding a more precise affine approximation and possibly a reduction of the number of affine terms computed by MDEIM. This will be specifically addressed in the next section;
- in principle, problems featuring different solution behaviors over the time interval could benefit by this local-in-time construction, carried out by partitioning the time interval in accordance with the different physical behaviors (this case however will not be addressed in this work).

4.5.2 Algorithmic procedures

We now focus on the construction of the MSRB preconditioner during the offline phase; the corresponding procedure is outlined in Algorithm 13. We start by building an approximated MDEIM affine decomposition of $\mathbf{D}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$, $\mathbf{M}^u(\boldsymbol{\mu})$ and dividing the time interval $[0, T]$ in $\mathcal{S}$ time slabs of equal size $\Delta\tau = T/\mathcal{S}$, such that in each time slab $\mathcal{N}_\tau$ time steps are contained. Then, given a set of parameters $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$ and a time slab s, we compute the corresponding FE solutions $\{\mathbf{u}^n(\boldsymbol{\mu}_i)\}_{n=\mathcal{N}_\tau s+1}^{\mathcal{N}_\tau(s+1)}$, $\{\mathbf{p}^n(\boldsymbol{\mu}_i)\}_{n=\mathcal{N}_\tau s+1}^{\mathcal{N}_\tau(s+1)}$, the supremizers snapshots $\{\mathbf{t}_p^i\}_{n=\mathcal{N}_\tau s+1}^{\mathcal{N}_\tau(s+1)}$ as in (4.31) and a set of local (in time) MDEIM basis $\mathbf{V}_C^{s,i}$ for the linearized term $\{\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i)\}_{n=\mathcal{N}_\tau s+1}^{\mathcal{N}_\tau(s+1)}$ for $i = 1, \dots, n_s$; then a MDEIM global basis $\mathbf{V}_C^s$ (with respect to the parameters) is extracted for the time slab s. In the next phase, we build the basis for the RB coarse operators: at each iteration k, a basis from the velocity, pressure and supremizers snapshots is computed and used to build the corresponding (approximated) RB affine decompositions

$$\begin{aligned}
\mathbf{M}_{N_k}^{\vec{u},q,s} &= \mathbf{V}_{uk,s}^T \mathbf{M}^q \mathbf{V}_{uk,s}, && q = 1, \dots, Q_m,\\
\mathbf{D}_{N_k}^{q,s} &= \mathbf{V}_{uk,s}^T \mathbf{D}^q \mathbf{V}_{uk,s}, && q = 1, \dots, Q_d,\\
\mathbf{B}_{N_k}^{q,s} &= \mathbf{V}_{pk,s}^T \mathbf{B}^q \mathbf{V}_{uk,s}, && q = 1, \dots, Q_b,\\
\mathbf{C}_{N_k}^{q,s} &= \mathbf{V}_{uk,s}^T \mathbf{C}^q \mathbf{V}_{uk,s}, && q = 1, \dots, Q_c,
\end{aligned}$$

which are then used at time slab s. In the online phase, given t_n and $\boldsymbol{\mu}$, the MSRB preconditioner built for the time slab s is built and applied to $\mathbf{v}_k^n(\boldsymbol{\mu})$.

Algorithm 14 NAVIER-STOKES-MSRB-OFFLINE

```
procedure NS-MSRB-OFFLINE($\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}, \varepsilon_r, \{N_k\}_k$, $\delta_{\mathrm{MDEIM}}, \varepsilon_C^{loc}, \varepsilon_C, \mathcal{S}$)
    Use MDEIM to compute an affine decomposition of $\mathbf{D}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$, $\mathbf{M}^u(\boldsymbol{\mu})$
    for $s = 0, \dots, \mathcal{S} - 1$ do
        for i = 1 : $n_s$ do
            Compute $\{\mathbf{u}^n(\boldsymbol{\mu}_i)\}_{n=\mathcal{N}_\tau s+1}^{\mathcal{N}_\tau(s+1)}$, $\{\mathbf{p}^n(\boldsymbol{\mu}_i)\}_{n=\mathcal{N}_\tau s+1}^{\mathcal{N}_\tau(s+1)}$, $\{\mathbf{t}_p^n(\boldsymbol{\mu}_i)\}_{n=\mathcal{N}_\tau s+1}^{\mathcal{N}_\tau(s+1)}$
            Set $\mathbf{S}_C^i = [\mathrm{vec}(\mathbf{C}(\mathbf{u}^{\mathcal{N}_\tau s,*}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i)), \dots, \mathrm{vec}(\mathbf{C}(\mathbf{u}^{\mathcal{N}_\tau(s+1)-1,*}(\boldsymbol{\mu}_i); \boldsymbol{\mu}_i))]$
            and $\mathbf{V}_C^{s,i} = \mathrm{POD}(\mathbf{S}_C^i, \mathbf{I}_{(N_h^u)^2}, \varepsilon_C^{loc})$
        end for

        Set $\mathbf{S}_{\vec{u}}^{(0)} = [\mathbf{u}^{\mathcal{N}_\tau s+1}(\boldsymbol{\mu}_1), \dots, \mathbf{u}^{\mathcal{N}_\tau(s+1)}(\boldsymbol{\mu}_1), \dots, \mathbf{u}^{\mathcal{N}_\tau s+1}(\boldsymbol{\mu}_{n_s}), \dots, \mathbf{u}^{\mathcal{N}_\tau(s+1)}(\boldsymbol{\mu}_{n_s})]$
        Set $\mathbf{S}_p^{(0)} = [\mathbf{p}^{\mathcal{N}_\tau s+1}(\boldsymbol{\mu}_1), \dots, \mathbf{p}^{\mathcal{N}_\tau(s+1)}(\boldsymbol{\mu}_1), \dots, \mathbf{p}^{\mathcal{N}_\tau s+1}(\boldsymbol{\mu}_{n_s}), \dots, \mathbf{p}^{\mathcal{N}_\tau(s+1)}(\boldsymbol{\mu}_{n_s})]$
        Set $\mathbf{S}_{\vec{t}}^{(0)} = [\mathbf{t}_p^{\mathcal{N}_\tau s+1}(\boldsymbol{\mu}_1), \dots, \mathbf{t}_p^{\mathcal{N}_\tau(s+1)}(\boldsymbol{\mu}_1), \dots, \mathbf{t}_p^{\mathcal{N}_\tau s+1}(\boldsymbol{\mu}_{n_s}), \dots, \mathbf{t}_p^{\mathcal{N}_\tau(s+1)}(\boldsymbol{\mu}_{n_s})]$
        Set $\mathbf{S}_C = [\mathbf{V}_C^{s,1}, \dots, \mathbf{V}_C^{s,n_s}]$
        Compute MDEIM basis $\mathbf{V}_C^s = \mathrm{MDEIM}(\mathbf{S}_C, \varepsilon_C)$
        Set $[\mathbf{C}_{N_k}^{s,1}, \dots, \mathbf{C}_{N_k}^{s,Q_c}] = \mathrm{unvec}(\mathbf{V}_C^s)$

        while $\prod_k \delta_{RB,k} > \varepsilon_r$ do
            $[\mathbf{V}_{uk,s}, \delta_{RB,k}] = \mathrm{POD}(\mathbf{S}_{\vec{u}}^{(k)}, \mathbf{X}_u, N_k)$
            $[\mathbf{V}_{pk,s}, \delta_{RB,k}] = \mathrm{POD}(\mathbf{S}_p^{(k)}, \mathbf{X}_p, N_k)$
            $[\mathbf{V}_{sk,s}, \delta_{RB,k}] = \mathrm{POD}(\mathbf{S}_{\vec{t}}^{(k)}, \mathbf{X}_u, N_k)$
            $\mathbf{V}_{uk,s} = \text{G-S}(\mathbf{V}_{uk,s}, \mathbf{V}_{sk,s}, \mathbf{X}_u)$
            Build RB affine matrices $\{\mathbf{M}_{N_k}^{\vec{u},q,s}\}_{q=1}^{Q_m}$, $\{\mathbf{D}_{N_k}^{q,s}\}_{q=1}^{Q_d}$, $\{\mathbf{B}_{N_k}^{q,s}\}_{q=1}^{Q_b}$, $\{\mathbf{C}_{N_k}^{q,s}\}_{q=1}^{Q_c}$
            Compute new snapshots $\{\mathbf{y}_{u,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ and $\{\mathbf{y}_{p,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ with (2.42)
            Compute new supremizer snapshots $\{\mathbf{y}_{t,k}(\boldsymbol{\mu}_i)\}_{i=1}^{n_s}$ with (4.53)
            $\mathbf{S}_{\vec{u}}^{(k+1)} = [\mathbf{y}_{u,k+1}(\boldsymbol{\mu}_1), \dots, \mathbf{y}_{u,k+1}(\boldsymbol{\mu}_{n_s})]$,
            $\mathbf{S}_p^{(k+1)} = [\mathbf{y}_{p,k+1}(\boldsymbol{\mu}_1), \dots, \mathbf{y}_{p,k+1}(\boldsymbol{\mu}_{n_s})]$
            $\mathbf{S}_{\vec{t}}^{(k+1)} = [\mathbf{y}_{t,k+1}(\boldsymbol{\mu}_1), \dots, \mathbf{y}_{t,k+1}(\boldsymbol{\mu}_{n_s})]$
            $k = k + 1$
        end while
    end for
end procedure
```

4.6 Numerical results: flow past a cylinder

We consider the domain Ω^0 shown in Figure (4.6) (already taken into account when dealing with the heat equation) and the flow described by the NS equations. We set the inlet condition on Γ_d as

$$\vec{g}_{NS}(\boldsymbol{\mu}) = 8yz(H-y)(H-z)/H^4(1-\cos(2\pi t))\vec{e}_1 \qquad t \in (0,T), \tag{4.58}$$

where $\vec{e}_1 \in \mathbb{R}^3$ is the unit vector in direction x and $H = 0.41m$. The test case is inspired from the classical test for laminar flows proposed in [Schäfer et al., 1996]. We show the results for two computational grids: the first one featuring 100'256 elements and 20'245 vertices (Mesh #1), the second one with 802'048 tetrahedra and 147'558 vertices (Mesh #2). By employing Taylor-Hood FE spaces we come up with $N_h^u = 442'674$ and $N_h^p = 20'245$ degrees of freedom in the first case and with $N_h^u = 3'373'240$ and $N_h^p = 147'558$ in the second one, for velocity and pressure, respectively. The time discretization employs the BDF2 scheme with $\Delta t = 0.01$. The velocity and pressure fields computed with the finest grid are reported in Figure 4.7 for time $t = 2.5$ seconds and 4.8 seconds.

In all experiments, the offline phase is carried out by employing the FGMRES method with a final tolerance set to 10^{-8} and the SIMPLE preconditioner described in Algorithm 12, where the steps 1 and 2 use GMRES inner iterations (up to a tolerance of 10^{-5}) preconditioned with an Additive Schwarz preconditioner (with 2-level overlap) from the `Ifpack` package of Trilinos. In the online phase, we combine the RB coarse components with the same SIMPLE preconditioner, that is $\mathbf{P}(\boldsymbol{\mu}) = \mathbf{P}_{\text{SIMPLE}}(\boldsymbol{\mu})$ in (4.45). The problem is solved with 128 and 1024 computing cores for the coarse and fine grid, respectively.

At first, we consider the case where the problem is parametrized with respect to time only; subsequently, we call into play geometrical and physical parameters. In both tests,

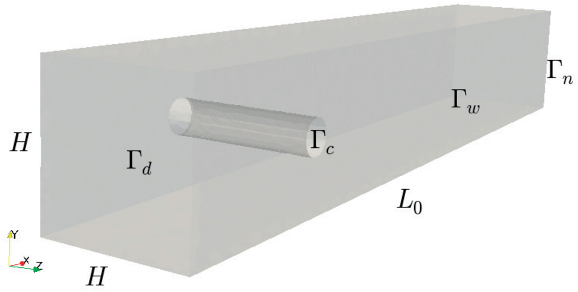


Figure 4.6 – Computational domain employed for the flow past a cylinder, with $H = 0.41$m, $L_0 = 2.5$m. The cylinder has radius $r = 0.05$m, its center is distant 0.5m from Γ_d and 0.2m from the bottom face.

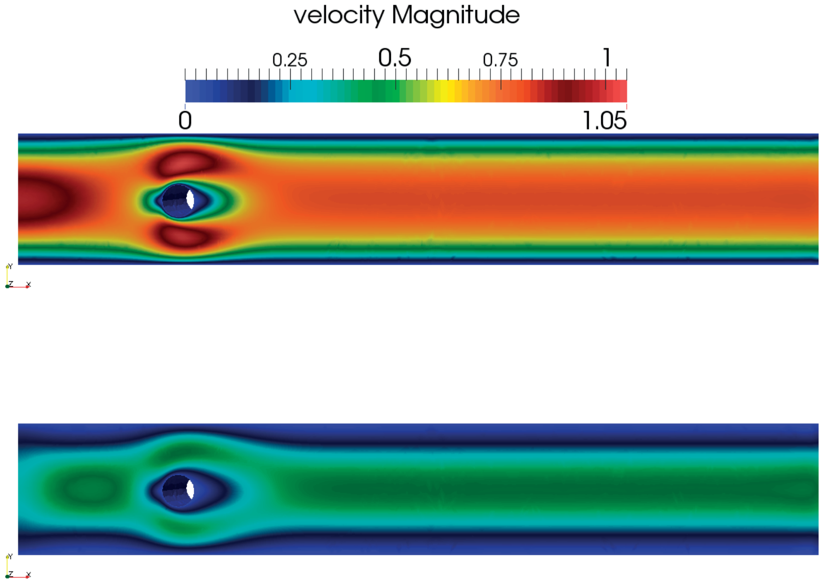


Figure 4.7 – Clip of velocity field at the central section at time $t = 2.5$ seconds (top) and $t = 4.8$ seconds (bottom) with $\nu = 0.01$.

we are interested in computing the drag coefficient C_D and the lifting coefficient C_L, which are defined as

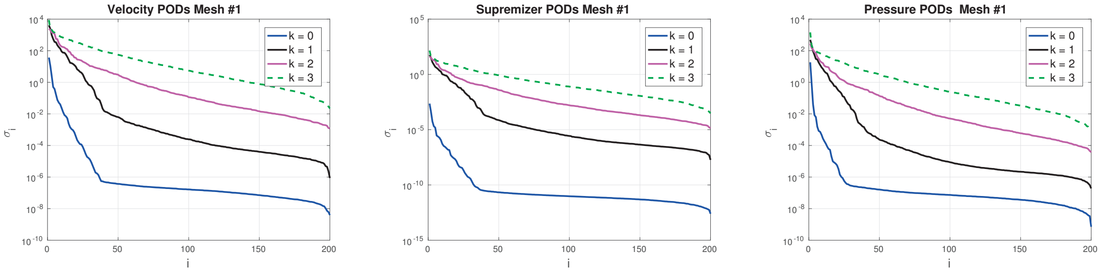
$$C_D = C_D(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu})) = -\frac{1}{q_\infty S}\int_{\Gamma_C(\boldsymbol{\mu})} (\boldsymbol{\sigma}(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu}))\vec{n}) \cdot \vec{v}_\infty d\Gamma_C(\boldsymbol{\mu}) \tag{4.59}$$

$$C_L = C_L(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu})) = \frac{1}{q_\infty S}\int_{\Gamma_C(\boldsymbol{\mu})} (\boldsymbol{\sigma}(\vec{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu}))\vec{n}) \cdot \vec{n}_\infty d\Gamma_C(\boldsymbol{\mu}), \tag{4.60}$$

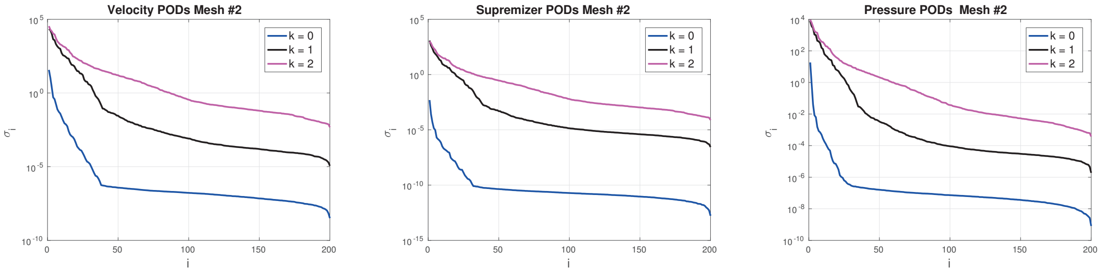
where S is the surface area of the cylinder, $\vec{v}_\infty$ is the unit vector directed as the incoming flow, $q_\infty = 0.5\tilde{\rho}V_\infty^2$, with $\tilde{\rho}$ the fluid density, is the dynamic pressure, while $\vec{n}_\infty$ is a unit vector orthogonal to $\vec{v}_\infty$.

4.6.1 Periodic regime case

In this first test case, we consider $\nu = 0.01$, leading to a Reynolds number $Re \in (0, 4.45)$ and the time interval [0, 10]. Furthermore, we employ the MSRB preconditioner in a predictive setting: during the offline phase, we train it on the interval $[0, 2]$ (on $\mathcal{S} = 1$ time slab); then we use it during the online phase for the interval $(2, 10]$. We set $n_s = 1$, $\varepsilon_r = 10^{-9}$ and $N_k = 20$ velocity, supremizer and pressure RB functions, leading to coarse operators of dimension $3N_k = 60$. The PODs corresponding to velocity snapshots, supremizer snapshots and pressure snapshots are reported in Figure 4.8a, 4.8b and 4.8c for Mesh #1 and in Figure 4.8d, 4.8e and 4.8f for Mesh # 2, respectively, and a similar behavior can be observed with the two meshes for the decay of the singular values. As



(a) Velocity - Mesh #1. (b) Supremizer - Mesh #1. (c) Pressure - Mesh #1.



(d) Velocity - Mesh #2. (e) Supremizer - Mesh #2. (f) Pressure - Mesh #2.

Figure 4.8 – POD corresponding to the construction of the RB spaces for Mesh #1 (top row) and Mesh #2 (bottom row). We distinguish between the creation of the RB spaces for velocity (left), supremizer (centre) and pressure (right).

analyzed in Section 2.4.1 for the steady case, the decay of the singular values is slower as the iteration k grows; by using a fixed dimension approach, this leads to the construction of less accurate RB coarse components. The number of RB coarse corrections produced by Algorithm 14 is $L = 4$ in the first case (Mesh #1) and $L = 3$ in the second one (Mesh #2).

Concerning the term $\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$, the singular values corresponding to the local-in-time MDEIM are reported in Figure 4.9 for the two computational grids. By setting a tolerance equal to $\varepsilon_C^{loc} = 10^{-6}$, MDEIM leads to 4 matrix bases (Mesh #1) and 3 matrix bases (Mesh #2); as a matter of fact, only few terms are needed to provide an accurate approximate affine decomposition of the time trajectory of $\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$, this is explained by the extremely rapid decay of the singular values, cf. Figure 4.9. Since we do not have a parameter dependence, the final MDEIM w.r.t. to the parameter variability (step 13 in Algorithm 14) is not performed.

The problem is then solved on the interval $(2, 10]$ and the corresponding results are reported in Table 4.6. Compared to using only the SIMPLE preconditioner, about one third of the iterations are needed to reach the same convergence tolerance of 10^{-5} in the FGMRES method, and about half of the time with Mesh #1 and one third with Mesh #2. The evolution of the drag and lift coefficients as function of the time is reported in Figure 4.10. Mesh #1 leads to an error of about 0.3% on C_D and 4.8% for C_L. In

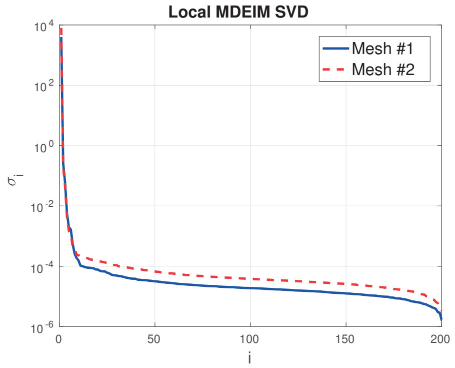


Figure 4.9 – SVD for local-in-time MDEIM Mesh #1 and Mesh #2.

Table 4.6 – Summary results for periodic regime. The problem is solved online on the interval $(2, 10]$. Computational times are expressed in seconds and refer to the average time needed for the solution of one time step.

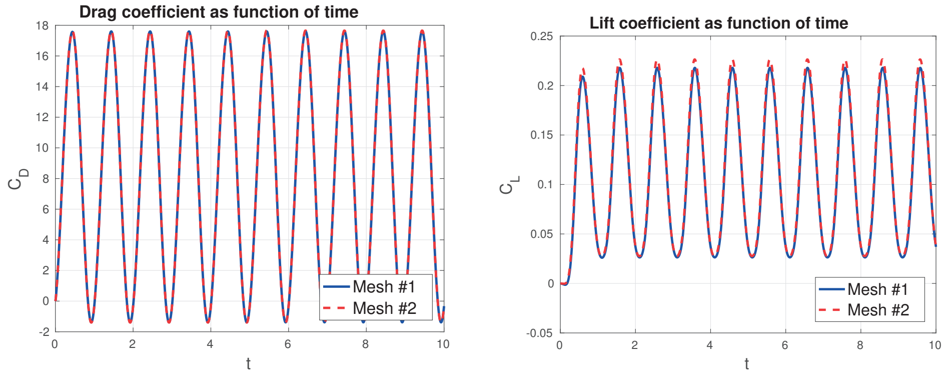
	$t^{\text{onl}}_{\text{MSRB}}$	It^{onl}	t_{SIMPLE}	It_{SIMPLE}	t_{off}
Mesh #1	5.15	4	9.42	11	5430.6
Mesh #2	13.28	8	44.07	27	12762.1

Figure 4.11 the phase diagram of C_D-C_L is reported and the convergence towards the periodic regime can be clearly observed.

4.6.2 Parametrized case

In the second part, we consider the NS equations for the time interval $[0, 1]$ in the parametrized domain

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



(a) Drag coefficient C_D vs t.

(b) coefficient C_L vs t.

Figure 4.10 – Time evolution of C_D and C_L for two different computational grids.

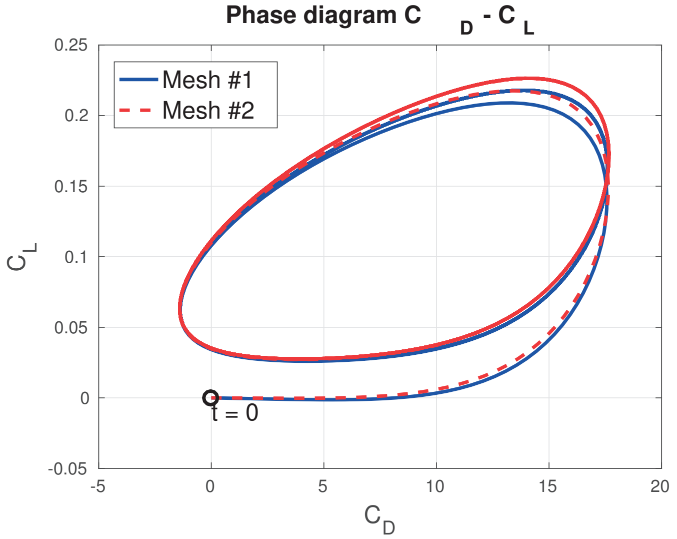


Figure 4.11 – Phase diagram of the drag and lift coefficients evolution for $t \in [0, 10]$.

where $\vec{d}(\boldsymbol{\mu})$ solves the following (vector) Laplace equation

$$\begin{cases} -\Delta\vec{d}(\boldsymbol{\mu}) = \vec{0} & \text{in } \Omega^0 \\ \vec{d}(\boldsymbol{\mu}) = \vec{h}(\boldsymbol{\mu}) & \text{on } \Gamma_C \\ \vec{d}(\boldsymbol{\mu}) = \vec{0} & \text{on } \partial\Omega^0 \backslash \Gamma_C, \end{cases} \tag{4.61}$$

where $\vec{h}(\boldsymbol{\mu}) = [0,\, h_2(\boldsymbol{\mu}),\, 0]^T$ and

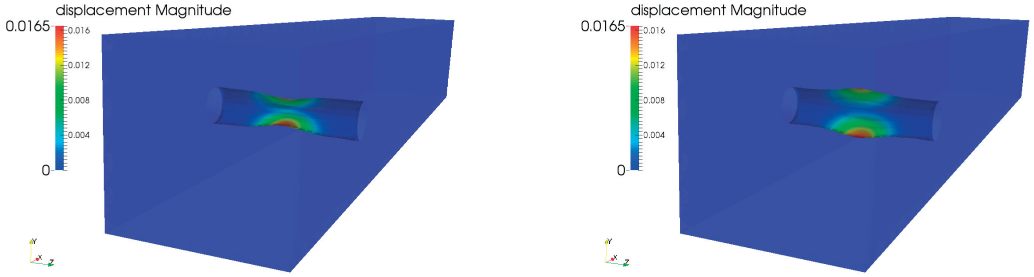
$$h_2(\boldsymbol{\mu}) = -(x_2 - 0.205) r_y \exp\{-200(x_3 - 0.205)^2\}, \tag{4.62}$$

entailing a displacement on the section cylinder along the y coordinate; a few examples of deformed domains are reported in Figure 4.12. We introduce the parameter vector

$$\boldsymbol{\mu} = (\nu, r_y) \in \mathcal{D} = (0.001, 0.01) \times (-0.3, 0.3)$$

where ν is the kinematic viscosity appearing in the definition of the stress tensor (4.2), and r_y is the parameter in (4.62); the considered values of ν lead to a Reynolds number in the range of [0, 44.5]. In the following, we present results for the two meshes Mesh #1 and Mesh #2 already considered in the previous section.

We solve the FE problem in the offline phase (with the same settings as in the previous section) on 16 offline parameters, which are chosen according to a 4×4 tensor grid with equidistant points. The time interval is divided into $\mathcal{S} = 2$ time slabs (of size $\Delta\tau = 0.5$) and on each of the MSRB preconditioner is built with a fixed dimension approach, with $N_k = 80$ RB basis functions for velocity, supremizer and pressure. This leads to a dimension for the RB coarse operators equal to $3N_k = 240$. The POD corresponding



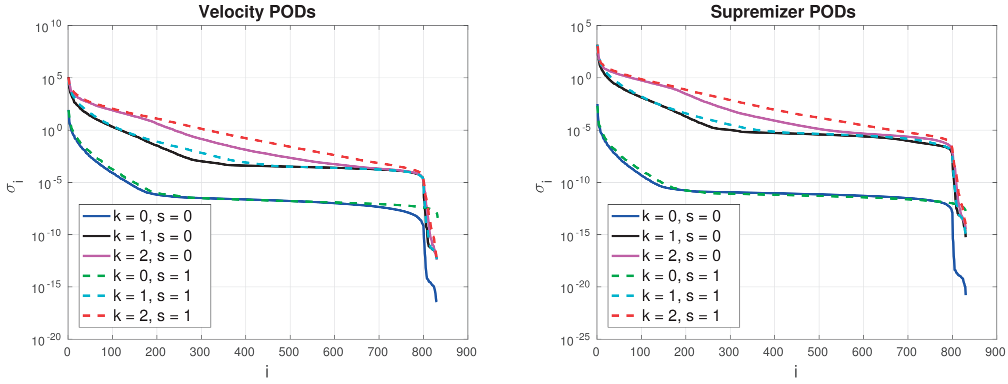
(a) $r_y = 0.3$.

(b) $r_y = -0.3$.

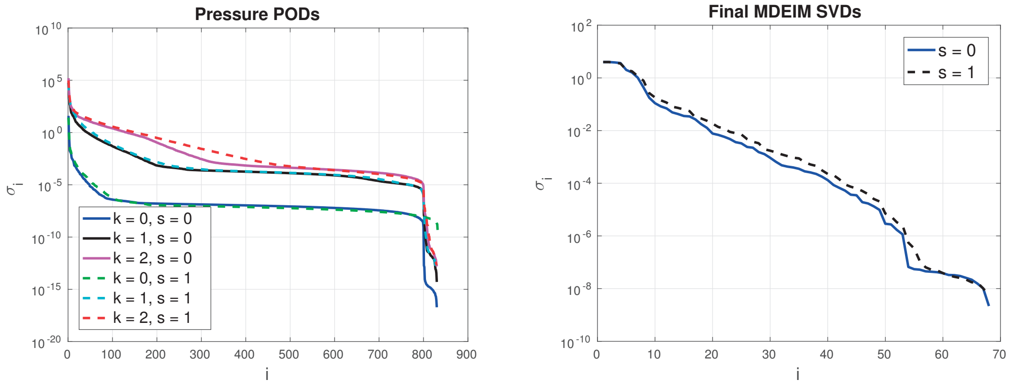
Figure 4.12 – Examples of deformation according to the value of r_y. If $r_y > 0$ the section of the cylinder in the y direction is narrowed, whereas a negative value enlarges it.

to the construction of the RB spaces is reported in Figure 4.13a, 4.13b and 4.13c for Mesh #2, a similar behavior can be observed, however, for Mesh #1 as well. Algorithm 14 builds $L = 3$ RB coarse components on each time slab and for the two meshes; as a matter of fact, the same number of RB spaces is built since the singular values on the two time slabs feature a similar decay. If we compare the results with the ones obtained for the periodic case analyzed in the previous section, the decay of the singular values is much slower (cf. Figure 4.8). This is an expected result due to the parametric dependence which enters into play in the test considered in this section. However, notice that about the same number of RB spaces is created (3 or 4) for both tests and meshes, thanks to suitably setting N_k as the desired dimension; in this case the dimensions of the RB spaces are indeed four times larger than in the previous test (240 instead of 60).

Concerning the affine approximation of the nonlinear term, for each parameter considered in the offline phase a decay of the eigenvalues similar to the one in Figure 4.9 can be observed, leading to collect between 4 and 5 MDEIM basis matrices for each instance of the parameter and for each time slab when a tolerance $\varepsilon_C^{loc} = 10^{-7}$ is employed, leading to a total of 68 matrix snapshots for the final MDEIMs for both time slabs. These matrix snapshots are then used to build the final MDEIM basis to affinely approximate the matrix $\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$, and the SVD corresponding to the final MDEIMs for time slabs $s = 0, 1$ is reported in Figure 4.13d for Mesh #2: by using a final tolerance $\varepsilon_C = 10^{-6}$, 45 high modes are retained for the first time slab, whereas 47 matrix basis are constructed in the second one. On the other hand for Mesh #1 58 affine terms are retained for both time slabs in the final MDEIM. A summary of the number of the affine decomposition can be found in Table 4.7, where Q_a sums the number of affine terms for $\mathbf{D}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$ and $\mathbf{M}^u(\boldsymbol{\mu})$ (and it does not change from one time slab to another), Q_c^t is the average number of affine terms for the MDEIMs in time and Q_c the one for the final MDEIM. As expected, there is not a significant difference for the two meshes and the two time slabs, since Q_c mainly depends on the time trajectory and the $\boldsymbol{\mu}$-dependence.

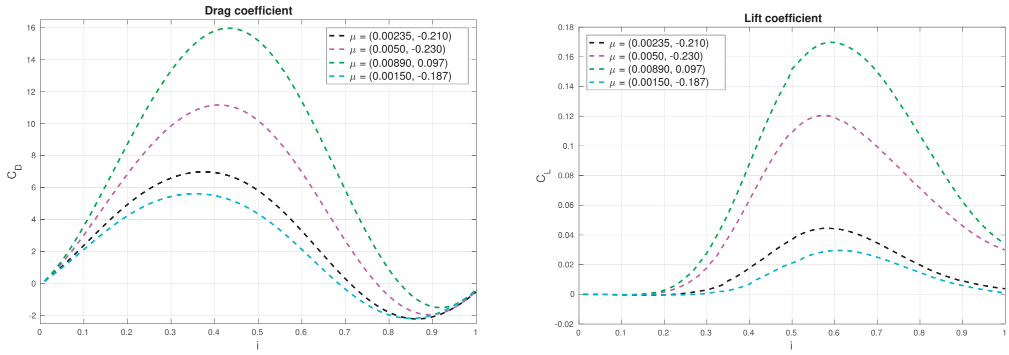


(a) Velocity PODs. (b) Supremizer PODs.



(c) Pressure PODs. (d) Final MDEIM.

Figure 4.13 – SVD for Mesh # 2. We report the PODs employed to build the RB spaces for velocity (top-left), supremizer (top-right), pressure (bottom-left); solid lines correspond to the first time slab $s = 0$, dash lines to the second time slab $s = 1$. Singular values (SV) for the final MDEIM for the two time slabs $s = 0, 1$ are in the bottom-right corner. A similar decay of the SV is observed for the two time slabs in all the cases.



(a) Drag coefficient. (b) Lift coefficient.

Figure 4.14 – Evolution of drag and lift coefficients for different values of the parameters computed during the online phase.

Table 4.7 – Number of affine terms (MDEIM decompositions) for the two meshes and the two time slabs. The value of Q_a sums the number of affine components of $\mathbf{D}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$ and $\mathbf{M}^u(\boldsymbol{\mu})$, Q_c^t is the average number of affine terms for the local-in-time MDEIMs and Q_c the number of affine terms for the final MDEIM.

	slab s	Q_a	Q_c^t	Q_c
Mesh #1	0	9	5.25	58
Mesh #1	0	9	5	58
Mesh #2	1	8	4	45
Mesh #2	1	8	4	47

Table 4.8 – Summary results for parametrized case. The problem is solved online on the interval $(0, 1]$ on instances of the parameter different from the training set. Computational times are expressed in seconds and refer to the average time needed for the solution of one time step.

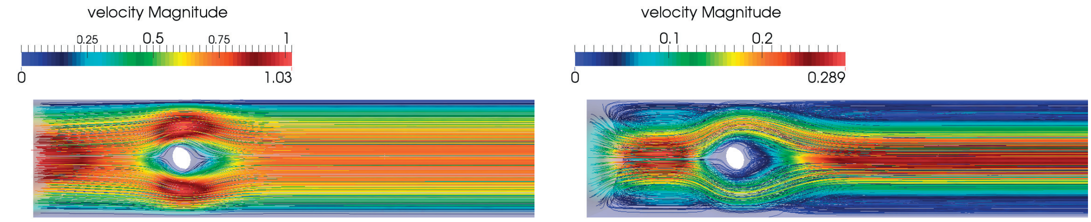
	$t_{\text{MSRB}}^{\text{onl}}$	It^{onl}	t_{SIMPLE}	It_{SIMPLE}	t_{off}	SPEEDUP	BEP
Mesh #1	2.94	2	7.40	11	24275.9	2.5	55
Mesh #2	12.42	7	44.07	27	96004.4	3.5	31

The problem is solved online up to a tolerance of 10^{-5} on the relative residual; the corresponding results and a comparison with the ones obtained by using the SIMPLE preconditioner alone is given in Table 4.8. The problem is solved in about 3 and 12.5 seconds per time step for Mesh #1 and #2, respectively. If compared to the SIMPLE preconditioner, the number of iterations significantly decreases of about 80% in the first case and 75% in the second one, leading to a significant speedup of 2.5 and 3.5 for the two computational grids. Moreover, we define the break even point (BEP), as the number of parameter evaluations to repay the offline phase, in other words

$$\text{BEP} = \frac{t_{\text{off}}}{N_t(t_{\text{SIMPLE}} - t_{\text{MSRB}}^{\text{onl}})}, \tag{4.63}$$

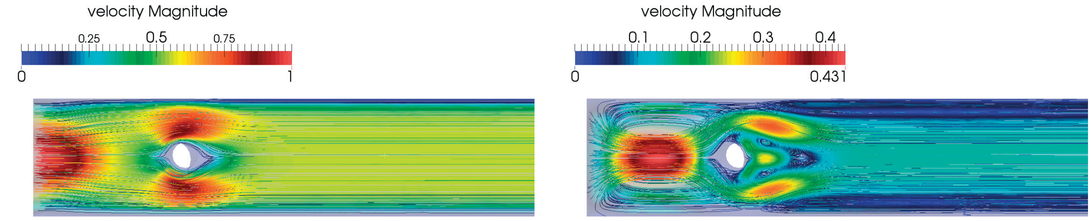
where N_t is the number of time steps; its computation is as well reported and is equal to 55 and 31 online evaluations, depending on the employed grid. As remarked also in other tests carried out in this thesis, it is evident how the MSRB preconditioning strategy is more convenient when the dimension of the FE problem becomes larger. This is also confirmed in this case, where with the larger grid a smaller BEP and a higher speedup are achieved if compared with the smaller mesh.

Some examples of solutions computed using the MSRB preconditioner for few instances of the parameter (with Mesh #2) can be found in Figure 4.15. For the same parameters the time evolution of the drag and lift coefficients has been computed and reported in Figure 4.14. The range of C_D and C_L, as expected, largely changes when physical and geometrical parameters vary.



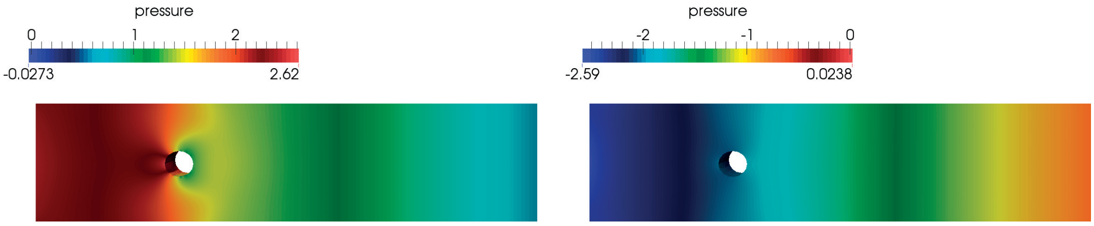
(a) $\boldsymbol{\mu} = (0.00889, 0.09), t = 0.5$.

(b) $\boldsymbol{\mu} = (0.00889, 0.09), t = 0.9$.



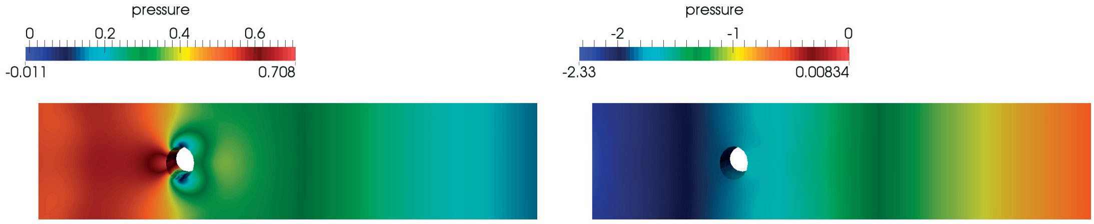
(c) $\boldsymbol{\mu} = (0.00145, -0.19), t = 0.5$.

(d) $\boldsymbol{\mu} = (0.00145, -0.19), t = 0.9$.



(e) $\boldsymbol{\mu} = (0.00889, 0.09), t = 0.5$.

(f) $\boldsymbol{\mu} = (0.00889, 0.09), t = 0.9$.



(g) $\boldsymbol{\mu} = (0.00145, -0.19), t = 0.5$.

(h) $\boldsymbol{\mu} = (0.00145, -0.19), t = 0.9$.

Figure 4.15 – Clip of the domain at the central section of the channel: zoom next to the cylinder of velocity streamlines and pressure field for different values of parameters and times.

5 Applications to the cardiovascular system

In this chapter we present different test cases related to parametrized cardiovascular simulations, where MSRB preconditioners are used to accelerate the solution of the parametrized system. After a briefly recall of the role of numerical simulations in cardiovascular pathologies, we consider three test cases of interest: *i*) the dynamics of a solute in blood flow, *ii*) the blood dynamics in carotid bifurcations and, finally, *iii*) an abdominal aortic bifurcation affected by an aneurysm where we simulate the behavior of the arterial tissue.

5.1 The role of mathematical simulation in cardiovascular applications

The mathematical and numerical modeling of the cardiovascular system acquired a conspicuous attention in recent years, representing a subject undergoing intense study from the numerical analysis community [Quarteroni et al., 2017, Formaggia et al., 2010, Taylor and Figueroa, 2009, Quarteroni et al., 2016b]. This trend is motivated by the well-known social importance (and impact) of cardiovascular pathologies, which represent nowadays the main cause of death in the western world, and by the remarkable contribution that computational mathematics can provide clinicians. As a matter of fact, its noninvasive nature allows to obtain quantitative information often not available from imaging and/or and measurements, but which are, at the same time, of great interest in the decision-making process of patients' treatment; examples of these quantities are, e.g., wall shear stresses and vorticity in vascular districts.

When dealing with cardiovascular applications, two important aspects need to be taken into account: the correct modeling of all the components entering into play and the use of proper numerical methods to accurately approximate the dynamics. These two factors, together with the recent improvement of hardware computational capabilities, make nowadays feasible the simulation of real-case scenarios, see e.g. [Malossi and Bonnemain, 2013, Forti, 2016] for the numerical study of blood flows in the arterial tree. These achievements, which rely on a high-fidelity approximation as the one computed through the FE method, represent a remarkable step in the correct simulation of the functioning of the cardiovascular system. On the other hand, such a complex dynamics is strongly related to the correct choice of important coefficients of the model, whose values are often experimentally acquired (thus uncertain) and, most of all, vary from one patient to another. Last but not least, the underlying geometry plays one of the most relevant roles in this perspective.

As a matter of fact, for each set of new parameter coefficients and geometry, the simulation must be carried out once again from scratch, thus making unfeasible the description of the dynamics for a large set of parameter scenarios. ROM techniques can give a significant contribution from this viewpoint, by building a cheap low-rank solver which can be used to compute outputs of interest in a wide range of predetermined cases, see e.g. [Manzoni et al., 2012a, Negri, 2015, Colciago et al., 2014, Pagani, 2017]. Similarly, in this chapter we aim at using the MSRB preconditioner developed so far to accelerate the simulation of problems arising in cardiovascular applications in a parametrized setting when the FE method, and thus the high fidelity linear system, must be solved. Towards this goal, three test cases are presented: *i*) the dynamics of a solute, which may represent oxygen or drugs, in the carotid bifurcation, *ii*) the blood dynamics in carotid bifurcations parametrized with respect to the geometry and, finally, *iii*) the simulation of the arterial tissue located in the abdominal aortic bifurcation and affected by an aneurysm. The computational grids employed in these test cases have been generated from patients' specific geometries and have been obtained by using the Vascular Modeling Toolkit (`vmtk`), [Antiga et al., 2008], for centerlines extraction and `gmsh` for the 3D mesh generation [Geuzaine and Remacle, 2009].

5.2 Solute dynamics in carotid bifurcation

The first test case of this chapter concerns the dynamics of a solute by focusing on the solution of a fluid-wall mass-transport model which describes the exchange of substances between blood in the lumen and arterial wall. In this context, the solute, which in our model can represent, e.g., oxygen, macromolecules or drugs, is regarded as a passive scalar transported along the artery by the blood, which is modeled as a Newtonian fluid and governs the exchange of the solute through the stress produced on the arterial wall. We take into account the so-called steady wall-free model for the absorption of the solute, [Quarteroni et al., 2002], which couples the steady Navier-Stokes (NS) equations with an advection-diffusion equation governing the concentration of the solute. This model is parametrized with respect to the permeability of the arterial wall and the diffusion coefficient of the solute in the blood, whereas the concentration of the solute in the wall is assumed to be constant. This problem has been largely addressed in the literature, see e.g. [Caputo et al., 2013, Quarteroni et al., 2002] and the references therein.

5.2.1 The physical model and its FE discretization

We consider an open bounded domain $\Omega_f \in \mathbb{R}^3$, such that $\partial\Omega_f = \Gamma_w \cup \Gamma_{out} \cup \Gamma_{in}$. Here, Γ_w, Γ_{out} and Γ_{in} denote the artery wall, the outlet and the inlet, respectively, see Figure 5.1a. The physical domain Ω_f describes the carotid bifurcation with an average section radius $r = 0.3\,cm$. We define $C_f \in [0,1]$ as the normalized concentration of the solute, whose dynamics is governed by the following advection diffusion equation:

$$\begin{cases} -\nabla \cdot (\nu_f \nabla C_f) + \tilde{\mathbf{u}} \cdot \nabla C_f = 0, & \mathbf{x} \in \Omega_f \\ \mathbf{n} \cdot (\nu_f \nabla C_f) + \xi C_f = \xi k_w & \text{on } \Gamma_w \\ C_f = 1 & \text{on } \Gamma_{in} \\ \mathbf{n} \cdot (\nu_f \nabla C_f) = 0 & \text{on } \Gamma_{out}, \end{cases} \tag{5.1}$$

where ν_f is the diffusivity coefficient of the solute, ξ and k_w are the permeability and the concentration in the arterial wall, respectively. We model the permeability of the wall as $\xi = \xi(\tilde{\mathbf{u}}) = \beta(1 + \tau_w(\tilde{\mathbf{u}}))$, being $\tau_w(\tilde{\mathbf{u}})$ the wall shear stress (WSS) distribution on Γ_w, and we choose as vector of parameters $\boldsymbol{\mu} = (\nu_f, \beta) \in [5 \cdot 10^{-5}, 5 \cdot 10^{-2}] \times [10^{-4}, 10^{-3}]$. On the other hand, we fix the value of $k_w = 0.5$ for all the simulations. The advection field $\tilde{\mathbf{u}} = \tilde{\mathbf{u}}(\mathbf{x})$ describes the velocity of the blood flow, and it is obtained as the solution of the NS equations corresponding during the diastolic phase. As boundary conditions for the NS equations we set a no-slip condition on Γ_w, homogeneous Neumann conditions on Γ_{out} and a parabolic inlet velocity, with a peak 22.5 cm s^{-1}, on Γ_{in}. Finally we consider a constant kinematic viscosity of the blood $\nu = 0.035$ $\mathrm{cm}^2\mathrm{s}^{-1}$. We remark that in the model considered in this section the NS equations are not parametrized, their solution only representing a datum for problem (5.1). Here we consider the solution of problem (5.1) for very small values of ν_f which yield huge Péclet numbers $Pe = \frac{|\tilde{\mathbf{u}}|r}{2\nu_f}$. Since the standard FE method may lead to oscillations for such convective dominant problems, we employ a stabilized FE formulation. In the following, we detail the weak formulation of problem (5.1) and its corresponding high-fidelity FE discretization involving a streamline-upwind/Petrov-Galerkin (SUPG) stabilization.

The variational formulation of problem (5.1) reads: find $C_f \in V = V(\Omega_f) = \{v \in H^1(\Omega_f) : v|_{\Gamma_{in}} = 1\}$ such that

$$\int_{\Omega_f} (\nu_f \nabla C_f \cdot \nabla w + \tilde{\mathbf{u}} \cdot \nabla C_f w) + \int_{\Gamma_w} \xi C_f w = \int_{\Gamma_w} \xi k_w w, \qquad \forall w \in H^1_{\Gamma_{in}}(\Omega_f), \tag{5.2}$$

where $H^1_{\Gamma_{in}}(\Omega_f) = \{v \in H^1(\Omega_f) : v|_{\Gamma_{in}} = 0\}$. To this aim, we introduce a conforming partition $\mathcal{T}_h$ of Ω_f and the FE space

$$X_h^r = \Big\{ w_h \in C^0(\bar{\Omega}_f) : w_h|_K \in \mathcal{P}_r(K) \, \forall K \in \mathcal{T}_h \Big\}, \tag{5.3}$$

where $\mathcal{P}_r(K)$ denotes the space of polynomials with degree lower than or equal to r on the element K. Then, the SUPG-FE formulation reads: find $C_{f,h} \in V_h = X_h^r \cap V$ such that

$$\begin{aligned}\int_{\Omega_f} (\nu_f \nabla C_{f,h} \nabla w_h + \tilde{\mathbf{u}} \cdot \nabla C_{f,h} w_h) + \int_{\Gamma_w} \xi C_{f,h} w_h \\ + \sum_{K \in \mathcal{T}_h} \Big(\tilde{\mathbf{u}} \cdot \nabla C_{f,h} - \nabla \cdot (\nu_f \nabla C_f) \tau_K \tilde{\mathbf{u}} \cdot \nabla w_h \Big)_K \\ = \int_{\Gamma_w} \xi k_w w_h, \qquad \forall w_h \in W_h = X_h^r \cap H^1_{\Gamma_{in}}(\Omega_f);\end{aligned} \tag{5.4}$$

here $(\cdot,\cdot)_K$ denotes the $L^2(K)$ scalar product on $K \in \mathcal{T}_h$, whereas

$$\tau_K = \delta_S \frac{h_K}{|\tilde{\mathbf{u}}|}, \tag{5.5}$$

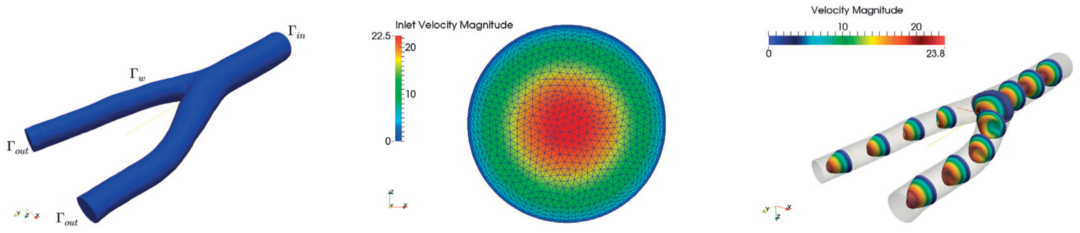
being δ_S a positive constant, which in the numerical experiments is set to 1, and h_K the diameter of the element $K \in \mathcal{T}_h$. Similarly to the previous cases, problem (5.4) can be written in algebraic form as (2.1).

A quantity of interest to be evaluated for different values of the parameters is the Sherwood number, whose distributions measures the non-dimensional mass flux through the vessel wall, see e.g. [Coppola and Caro, 2008], and is defined as

$$Sh = \frac{-2r(\nabla C_f \cdot \mathbf{n})}{C_{f,in} - k_w},$$

where $r = 0.3\,cm$ is the reference radius of the artery and $C_{f,in} = 1$ is the inlet concentration.

Concerning the numerical setting, we employ a mesh with boundary layer, and a $\mathcal{P}_2 - \mathcal{P}_1$ FE discretization for the Navier Stokes equations, whose resulting velocity field is reported in Figure 5.1c. Concerning the discretization of equation (5.4), we analyze the performance of the MSRB preconditioner $\mathbf{P}_{\text{MSRB},k}(\boldsymbol{\mu})$ with respect to the employment of $\mathcal{P}_1$ and $\mathcal{P}_2$ finite elements basis functions, resulting in computational linear systems with 429’892 and 3’467’673 unknowns, respectively. We are particularly interested in the case of quadratic ($\mathcal{P}_2$) elements because the evaluation of quantities involving the gradient of the concentration, as the Sherwood number, need a very accurate computation of the derivatives of the unknown. In Figure 5.2 we report the Sherwood number obtained for different instances of the parameter: we notice that employing quadratic FE polynomials can yield a smoother field. For the solution of the FE linear system with the FGMRES method, we use a stopping criterion based on the Euclidean norm of the FE residual rescaled with respect to the right hand side with a tolerance equal to $\varepsilon_r = 10^{-7}$.

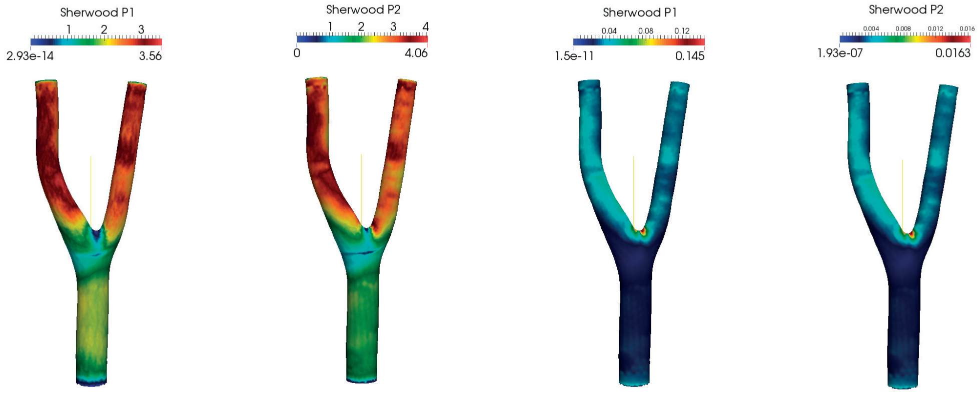


(a) Physical domain Ω_f with boundary conditions.

(b) Velocity inlet and grid.

(c) Velocity field.

Figure 5.1 – Inlet velocity profile with mesh and velocity field for the dynamics of a solute.



(a) $\boldsymbol{\mu} = (5 \cdot 10^{-5}, 10^{-4})$

(b) $\boldsymbol{\mu} = (5 \cdot 10^{-2}, 10^{-3})$

Figure 5.2 – Sherwood number distribution for values of the parameter vector.

5.2.2 Numerical results using the MSRB preconditioner

We now assess the computational performance of the MSRB preconditioner on this problem, which is constructed by combining the RB coarse operators for elliptic problems, devised in Chapter 2, with a block Jacobi preconditioner $\mathbf{P}_{\mathrm{BJ}}(\boldsymbol{\mu})$ as fine grid component. The results are compared with the ones obtained by employing the ML preconditioner $\mathbf{P}_{\mathrm{ML}}(\boldsymbol{\mu})$ from the `ML` package of `Trilinos`.

We first remark that very similar outcomes are obtained either with the fixed accuracy or the fixed dimension approach. In Table 5.1 and 5.2 we show detailed results for the fixed accuracy (with $\delta_{RB,k} = 0.001$, $k = 0, 1, 2$) and fixed dimension (with $N_k = 20$, $k = 0, 1, \ldots$) approach employing a number of cores $N_{\mathrm{core}} = 96$, 192, 384. The FGMRES method with the MSRB preconditioner converges in 3 iterations (at most), both for $\mathcal{P}_1$ and $\mathcal{P}_2$ finite elements: employing different FE degrees does not impact on the dimension of the reduced spaces, and consequently on the time needed for the solution online of

Table 5.1 – results for FGMRES obtained with MSRB preconditioner built a fixed accuracy approach, $\delta_{RB,k} = 0.001$, $k =, 1, 2$, $n_s = 300$. Time are expressed in seconds.

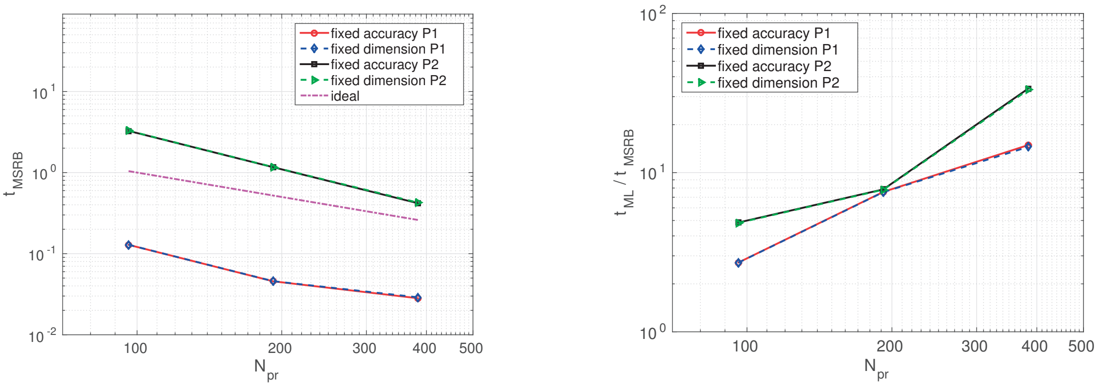
	N_{core}	L	N_k	$t^{\text{onl}}_{\text{MSRB}}$ (It)	t_{GML} (It)	t_{off}	t_{n_s}	t_{POD}	BEP
$\mathcal{P}_1$	96	3	8 22 46	0.12852 (2)	0.35 (57)	260.85	258.37	2.48	1178
$\mathcal{P}_1$	192	3	8 22 45	0.0456 (2)	0.34 (62)	185.31	183.16	2.15	619
$\mathcal{P}_1$	384	3	8 22 44	0.0282 (2)	0.42 (67)	188.47	186.34	2.13	482
$\mathcal{P}_2$	96	3	9 24 50	3.2651 (2)	15.86 (177)	9172.65	9158.77	13.88	729
$\mathcal{P}_2$	192	3	9 23 49	1.1635 (2)	9.13 (195)	4935.78	4927.46	8.32	620
$\mathcal{P}_2$	384	3	9 23 49	0.4188 (2)	14.12 (401)	5877.46	5872.17	5.29	429

Table 5.2 – results for FGMRES obtained with MSRB preconditioner built a fixed dimension approach, $\delta_{RB,k} = 0.001$, $k =, 1, 2$, $n_s = 300$. Time are expressed in seconds.

	N_{core}	L	N_k	$t^{\text{onl}}_{\text{MSRB}}$ (It)	t_{GML} (It)	t_{off}	t_{n_s}	t_{POD}	BEP
$\mathcal{P}_1$	96	4	20	0.12868 (2)	0.35 (57)	314.01	307.74	6.27	1419
$\mathcal{P}_1$	192	4	20	0.04576 (2)	0.34 (62)	208.85	205.67	3.18	698
$\mathcal{P}_1$	384	4	20	0.02896 (2)	0.42 (67)	201.1	198.05	3.05	515
$\mathcal{P}_2$	96	4	20	3.2818 (3)	15.86 (177)	10391.4	10370.12	21.28	827
$\mathcal{P}_2$	192	4	20	1.1689 (3)	9.13 (195)	5363.1	5350.8	12.3	674
$\mathcal{P}_2$	384	4	20	0.4264 (3)	14.12 (401)	6095.57	6087.78	7.79	446

the reduced problems. On the other hand, employing $\mathcal{P}_2$ FE has a huge impact on the performances of the $\mathbf{P}_{\text{ML}}(\boldsymbol{\mu})$ preconditioner: the iteration count is three times higher and the overall computational times largely increase.

The small sizes of the RB coarse operators plays a relevant role in the overall efficiency, since the computational times obtained with $\mathbf{P}_{\text{MSRB},k}(\boldsymbol{\mu})$ in the online phase are essentially dominated by the construction of the fine preconditioner $\mathbf{P}_{\text{BJ}}(\boldsymbol{\mu})$, which is embarrassingly parallel, thus yielding a very good scalability, see Figure 5.3a for both fixed dimension and fixed accuracy approach. Such result is motivated by the fact that the computational time is mainly governed by the LU factorizations of the local matrices in $\mathbf{P}_{\text{BJ}}(\boldsymbol{\mu})$. On the other hand, solving the linear system with $\mathbf{P}_{\text{ML}}(\boldsymbol{\mu})$ (and consequently the offline phase as it mainly involves snapshots computation) results in a larger time when using 384 cores due to the communication costs of the ML preconditioner. In Figure 5.3b we report the speedup in computational time obtained by employing $\mathbf{P}_{\text{ML}}(\boldsymbol{\mu})$ and $\mathbf{P}_{\text{MSRB},k}(\boldsymbol{\mu})$: by increasing the number of cores we solve the problem online up to 14 times faster than ML in the case of $\mathcal{P}_1$ elements and 35 in the case of $\mathcal{P}_2$ elements. As a result, the break-even point (BEP) of online evaluations decreases with the number of cores up to about 450 (resp. about 500) for $\mathcal{P}_2$ (resp. $\mathcal{P}_1$) elements.



(a) Scalability.

(b) Speedup.

Figure 5.3 – Scalability and speedup as function of the number of cores N_{core} for $\mathcal{P}_1$ and $\mathcal{P}_2$ FE basis functions and fixed accuracy and fixed dimension approach.

5.3 Blood flows in carotid bifurcations

The carotid bifurcation is located along the sides of the neck and furnishes the blood supply to the face and the brain [Wootton and Ku, 1999]. 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 5.4. 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 [Slager et al., 2005], in this perspective, numerical simulations can play a relevant role in providing quantitative results able to support clinicians. As the carotid bifurcation concerns, several studies have been conducted [Lancellotti et al., 2017, Guerciotti et al., 2016]. In the following we consider two problems related to the carotid bifurcation: at first, we analyze the dynamics of a solute in the carotid bifurcation; secondly, we investigate the behavior of the blood flow when different physical and geometrical configurations described in terms of parameters are considered.

We consider now the unsteady NS equations (4.1) in the carotid bifurcation with parametrized domain configurations and inlet boundary conditions.

5.3.1 Test case setting

To start with, as done in Section 3.6.2 for the Stokes flow, we consider a deformation of the reference domain, shown in Figure 5.4, obtained as the harmonic extension of a Neumann boundary datum. Hence, let us consider the following (vectorial) Laplace

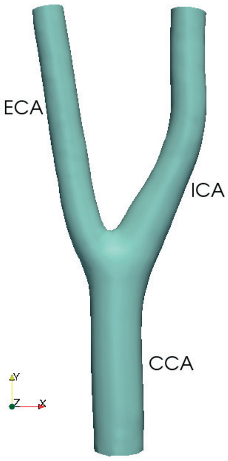


Figure 5.4 – Reference domain Ω_f; common carotid artery (CCA), internal carotid artery (ICA) and external carotid artery (ECA).

problem

$$\begin{cases} -\Delta \vec{d}(\boldsymbol{\mu}) = \vec{0} & \text{in } \Omega_f \\ \vec{d}(\boldsymbol{\mu}) = \vec{0} & \text{on } \Gamma_{in} \cup \Gamma_{out} \\ \dfrac{\partial \vec{d}(\boldsymbol{\mu})}{\partial \vec{n}} = \vec{h}(\boldsymbol{\mu}) & \text{on } \Gamma_w, \end{cases} \tag{5.6}$$

where the parametrized datum $\vec{h}(\boldsymbol{\mu})$ represents a stress load entailing a deformation which narrows the two branches of the bifurcation (notice that in the Stokes example only one of the branches was significantly deformed, instead). We introduce the region

$$A = \{\vec{x} \in \mathbb{R}^3 : \; r^2 \leq R^2\} \cap \partial\Omega_f, \qquad r^2 = r^2(\vec{x}) = x_1^2 + (x_2 - 2.5)^2 + x_3^2, \tag{5.7}$$

which identifies the portion of volume where $\vec{h}(\boldsymbol{\mu})$ is loaded as follows

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

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 5.5a, where the region affected by the load is reported in red.

By following the setup employed in [Lancellotti et al., 2017], 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 5.5b for a single heartbeat; the resulting prescribed inlet velocity $\vec{g}_{NS}(\boldsymbol{\mu})$ is the unique parabolic function in the normal direction 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 parabolic profile at the inlet represents a proper choice when dealing with carotid bifurcations, see e.g. [Campbell et al., 2012].

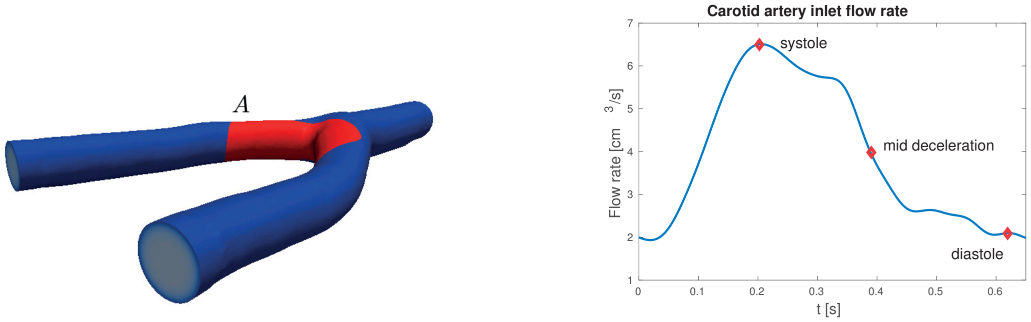
The parameter parameter vector is $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathcal{D} = [0.2,\, 0.4] \times [0.85, 1.0] \subset \mathbb{R}^2$; the value of μ_1 entails a narrowing of the bifurcation, thus simulating the effect of a stenosis obstructing the vessel; μ_2 determines instead the magnitude 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 ms^{-1}, when $\mu_2 = 1$, during the systolic phase.

Two examples of deformation with respect to the reference domain are reported in Figure 5.6 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.

Taylor-Hood ($\mathcal{P}_2 - \mathcal{P}_1$) 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. The deformation problem (5.6) is discretized by means of the FE method and solved with the AMG-preconditioned CG up to a tolerance of 10^{-7}. Notice that a coarser grid, with respect to the one used for the solute dynamics case, has been used due to the extensive computational effort entailed by the offline phase and the fact that the computational resources limit the job time to 24 hours. The simulations have been carried out by employing 32 cores.

5.3.2 Numerical results

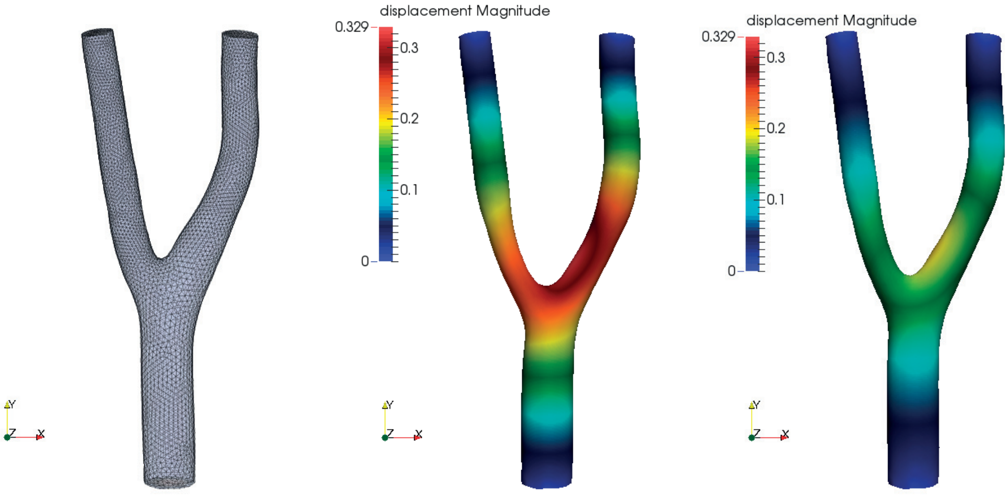
The numerical results are summarized in Table 5.3. In the construction of the MSRB preconditioner, we use the SIMPLE preconditioner $\mathbf{P}_{\mathrm{SIMPLE}}(\boldsymbol{\mu})$ as fine grid component, this latter employs GMRES inner iterations, with a final tolerance on the relative residual of 10^{-5}, where an Additive Schwarz preconditioner, with two layers of elements of overlap, is used for the two solve steps (cf. Algorithm 12). We set $\mathcal{S} = 1$ time slab, and 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 $Q_a = 12$ affine terms for their approximation in total. The same offline parameters are then employed to construct the solution snapshots



(a) Region A (in red) as defined in (5.7) .

(b) Inlet flow rate $Q_{\text{CCA}}(t)$ [cm^{3}s^{-1}] with highlighted systole, mid deceleration and diastole phases.

Figure 5.5 – Region A where the stress $\vec{h}(\boldsymbol{\mu})$ is applied (left) and reference flow rate $Q^0_{\text{CCA}}(t)$ (right).



(a) Starting mesh for Ω_f.

(b) $\boldsymbol{\mu}^1 = (0.375, 0.975)$.

(c) $\boldsymbol{\mu}^2 = (0.225, 0.875)$.

Figure 5.6 – Starting mesh for Ω_f (left) and deformation entailed by parameter instances $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$ (middle and right): the higher the value of μ_1, the larger the displacement entailed by $\vec{h}(\boldsymbol{\mu})$.

Table 5.3 – Summary results for blood flow in bifurcation. Computational times are expressed in seconds and $t_{\text{MSRB}}^{\text{onl}}$ and t_{SIMPLE} refer to the average time needed for the solution of one time step.

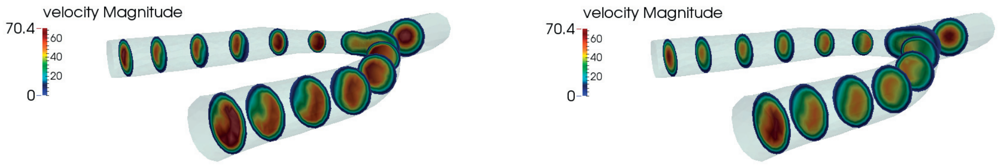
Q_a	Q_c^t	Q_c	$t_{\text{MSRB}}^{\text{onl}}$	It^{onl}	t_{SIMPLE}	It_{SIMPLE}	t_{off}	SPEEDUP	BEP
12	32	463	6.33	3	55.06	91	98074.1	8.68	65

for the MSRB coarse components, where a fixed dimension approach, with $N_k = 220$ RB functions for velocity, supremizer and pressure and $\varepsilon_r = 10^{-9}$ is set. Such a choice leads to the construction of 5 RB coarse operators $\mathbf{Q}_{N_k}(\boldsymbol{\mu})$, $k = 0, \dots, 4$, each with dimension $3N_k = 660$.

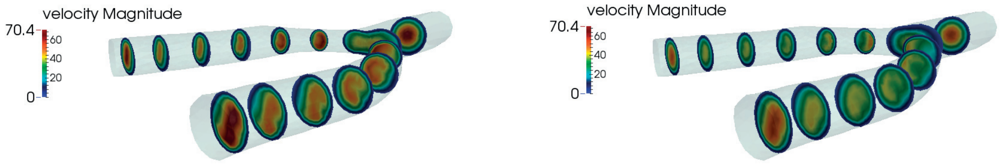
For the convective term $\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$, we compute the affine approximation following the double POD strategy as outlined in Algorithm 14, by first constructing for each parameter $\boldsymbol{\mu}_i$, $i = 1, \dots, n_s$ a MDEIM basis in time with a tolerance $\varepsilon_C^{loc} = 10^{-7}$, which in this case retains 32 matrix bases on average. Notice that compared to the cylinder case examined in Section 4.6, a much larger number of bases are retained to properly approximate the time trajectory of $\mathbf{C}(\mathbf{u}^{n,*}(\boldsymbol{\mu}); \boldsymbol{\mu})$, due to the larger Reynolds number of the case under examination. The final MDEIM retains instead 463 affine terms by employing a tolerance $\varepsilon_C = 5 \cdot 10^{-5}$.

The system is solved online for 20 new instances of the parameter, different from the training set $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s}$, up to a tolerance of 10^{-5} on the rescaled residual in 3 iterations and with a computational cost of 6.33 seconds per time step on average; this is a remarkable gain compared to the time employed by the SIMPLE preconditioner only, which requires 91 iterations and 55.06 seconds per time step on average. Hence, the MSRB preconditioning strategy leads to a significant speedup of 8.7 with respect to the use of the pure SIMPLE case.

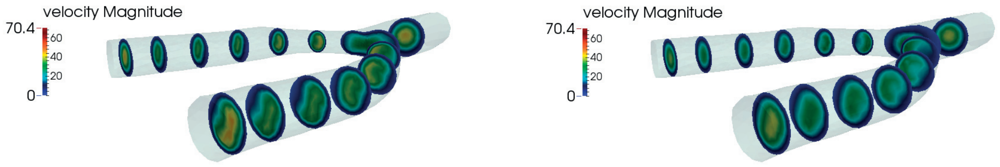
Examples of solutions for different values of the parameters, computed with MSRB preconditioner, are reported in Figure 5.9b. The velocity pattern greatly changes by varying the parameter configuration, also influencing the flow rate at the outlet boundaries: in Figure 5.9a we report the outflow rate $Q_{\text{ICA}}(t; \boldsymbol{\mu})$ at the ICA boundary as function of the time, while in Figure 5.9b the ratio $\frac{Q_{\text{ICA}}(t;\boldsymbol{\mu})}{Q_{\text{CCA}}(t;\boldsymbol{\mu})}$, that is the percentage of flow rate exiting from the ICA branch, is reported. As a matter of fact, the physical parameter μ_2, mainly affects the absolute value of $Q_{\text{ICA}}(t; \boldsymbol{\mu})$, whereas the geometrical parameter μ_1 plays a more relevant role in how the fluid is distributed between the two branches: the higher μ_1, the larger the portion of blood directed in the ICA with respect to the one entering the ECA.



(a) $\boldsymbol{\mu}^1 = (0.375, 0.975)$, $t = 0.2$. (b) $\boldsymbol{\mu}^2 = (0.225, 0.875)$, $t = 0.2$.



(c) $\boldsymbol{\mu}^1 = (0.375, 0.975)$, $t = 0.3$. (d) $\boldsymbol{\mu}^2 = (0.225, 0.875)$, $t = 0.3$.



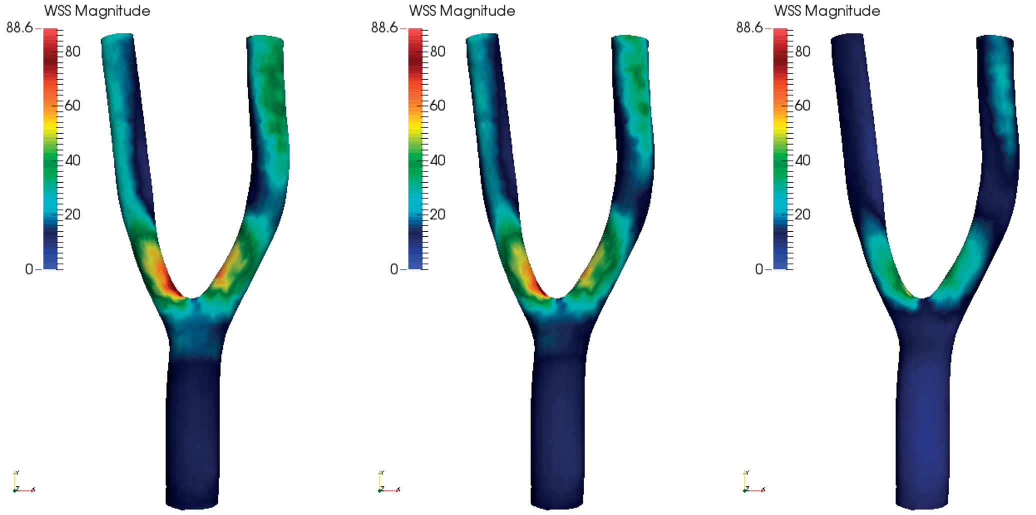
(e) $\boldsymbol{\mu}^1 = (0.375, 0.975)$, $t = 0.4$. (f) $\boldsymbol{\mu}^2 = (0.225, 0.875)$, $t = 0.4$.

Figure 5.7 – Slices of velocity magnitude for $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$ at different times.

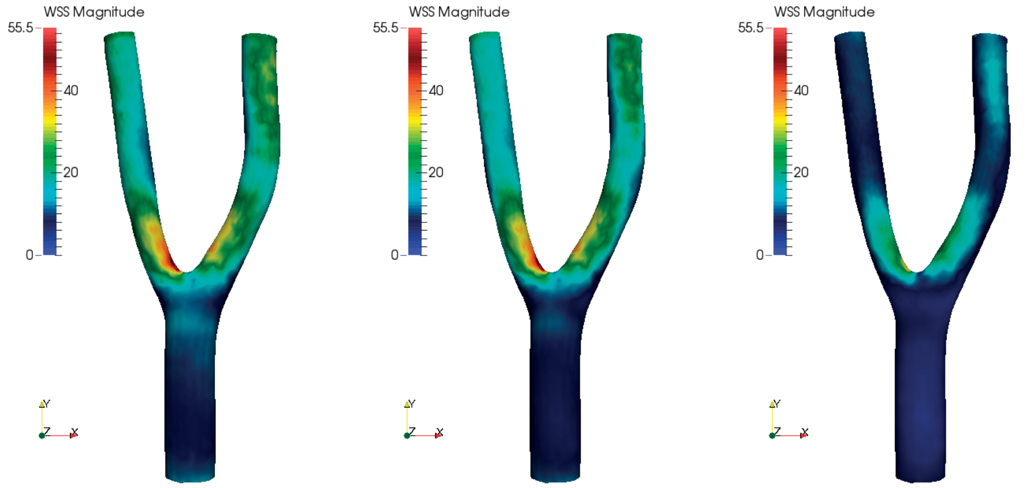
As already remarked, a relevant quantity of interest when dealing with cardiovascular simulations is the wall shear stress (WSS) distribution $\vec{\tau}_w$ on Γ_w, which is defined as

$$\vec{\tau}_w = \left(2\bar{\mu}\boldsymbol{\varepsilon}(\vec{u})\vec{n}\right) \cdot \vec{t} = 2\bar{\mu}\left(\boldsymbol{\varepsilon}(\vec{u})\vec{n} - (\boldsymbol{\varepsilon}(\vec{u})\vec{n} \cdot \vec{n})\vec{n}\right), \tag{5.8}$$

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 at different times; as expected, the WSS magnitude is is higher during the systolic peak and concentrated close to the separation of the branches. To further investigate the phenomenon, we place three probes in different location of Γ_w, shown in Figure 5.10a: P2 is located close to the outflow of the ICA, whereas P1 and P3 close to the bifurcation, at the entrance of the ICA and the ECA, respectively. The time dependence of the WSS magnitude in the points identifies by P1, P2 and P3 is reported for 5 values of the parameter in Figure 5.10b–5.10f. As a matter of fact both the geometrical and physical parameters give a large contribution to the time variability of $\vec{\tau}_w$, especially next to bifurcation at P1 and P3. In particular, in the latter location the WSS magnitude reaches the highest values, due to the smaller diameter of the ECA.

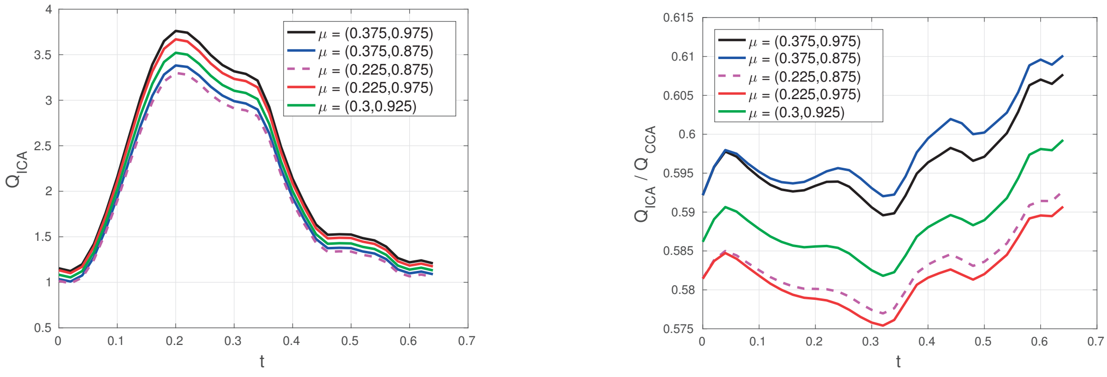


(a) $\boldsymbol{\mu}^1 = (0.375, 0.975)$, $t = 0.2$. (b) $\boldsymbol{\mu}^2 = (0.375, 0.975)$, $t = 0.3$. (c) $\boldsymbol{\mu}^1 = (0.375, 0.975)$, $t = 0.4$.



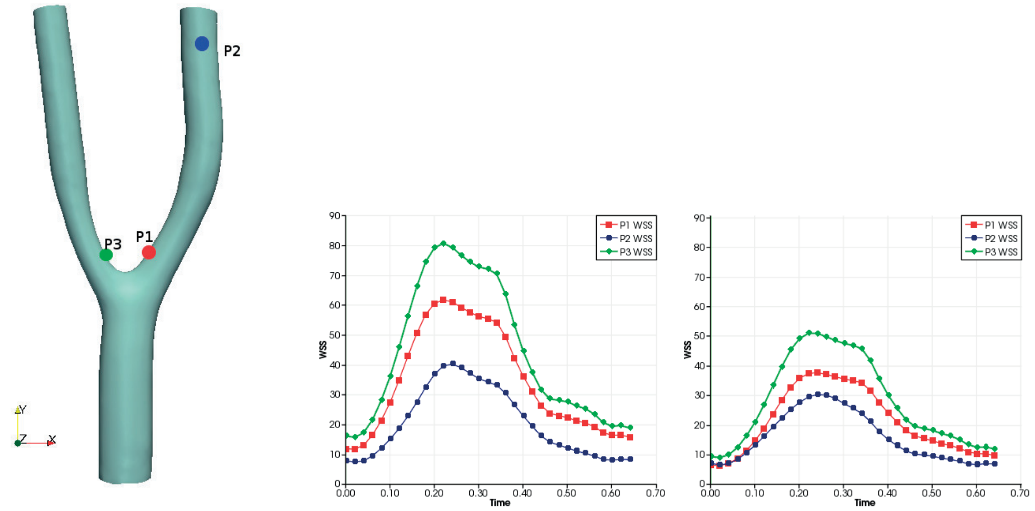
(d) $\boldsymbol{\mu}^2 = (0.225, 0.875)$, $t = 0.2$. (e) $\boldsymbol{\mu}^2 = (0.225, 0.875))$, $t = 0.3$. (f) $\boldsymbol{\mu}^2 = (0.225, 0.875)$, $t = 0.4$.

Figure 5.8 – WSS [dyn$\cdot$cm^{-2}] magnitude distribution at different times for $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$.

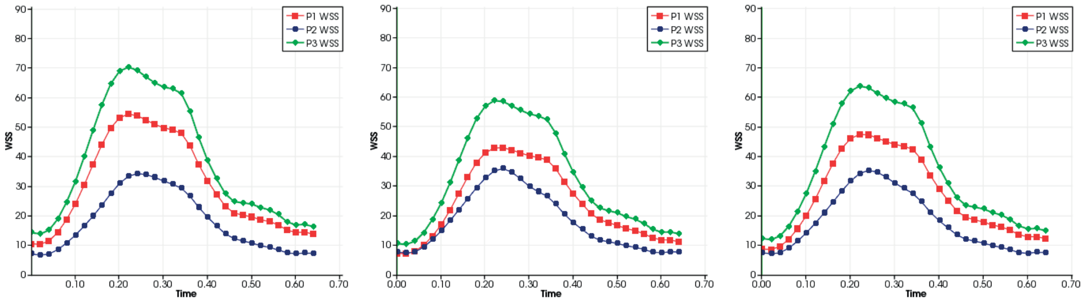


(a) $Q_{\mathrm{ICA}}(t;\boldsymbol{\mu})$. (b) $\frac{Q_{\mathrm{ICA}}(t;\boldsymbol{\mu})}{Q_{\mathrm{CCA}}(t;\boldsymbol{\mu})}$.

Figure 5.9 – Flow rate $Q_{\mathrm{ICA}}(t;\boldsymbol{\mu})$ at the ICA (left) and ratio $\frac{Q_{\mathrm{ICA}}(t;\boldsymbol{\mu})}{Q_{\mathrm{CCA}}(t;\boldsymbol{\mu})}$ (right) for different values of $\boldsymbol{\mu}$.



(a) Location of probes. (b) $\boldsymbol{\mu}^1 = (0.375, 0.975)$. (c) $\boldsymbol{\mu}^2 = (0.225, 0.875)$.



(d) $\boldsymbol{\mu}^3 = (0.375, 0.875)$. (e) $\boldsymbol{\mu}^4 = (0.225, 0.975)$. (f) $\boldsymbol{\mu}^5 = (0.3, 0.925)$.

Figure 5.10 – WSS [dyn·cm^{-2}] magnitude distribution at different times for $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$.

Figure 5.11 – Location of probes (top left) and time evolution of the distribution of the magnitude of the WSS [dyn·cm^{-2}] for different parameter values.

5.4 Abdominal aorta aneurysm in patient specific geometry

Abdominal aortic aneurysms (AAAs) represent a relevant disease in population above the age of 65, being the cause of death from 75 % to 90% of cases, see [Fleming et al., 2005, Reid et al., 1990]. The diameter, that is, the maximum transverse dimension of the AAA, is considered as the main indicator for the rupture risk in the current clinical treatment. However, more recently, the analysis of the wall stresses demonstrated to provide more reliable measures to predict AAA's rupture, which occurs when the wall stresses overcome the wall strength [Fillinger et al., 2002]. Due to its relevant social issues, studies have been conducted to evaluate the rupture of an AAA, see e.g. [Raghavan and Vorp, 2000, Gasser et al., 2010] and references therein; FE analysis thus represents a convenient and noninvasive tool in this respect [Maier et al., 2010, Colciago, 2014].

In this section, we consider a parametrized model to describe, as a function of the Young modulus and the arterial pressure, the displacement of the arterial wall. Such a phenomenon is modeled with a linear elastodynamics equation, which is discretized by means of the FE method in space and the BDF scheme in time. By doing so, for each parameter we obtain a sequence (in time) of parametrized linear systems, and we use the MSRB preconditioning strategy, as the one employed for parabolic problems and devised in Section 2.5, to accelerate their solution. We highlight that the considered model is rather simple to capture the physiological dynamics, which is normally simulated through the use of complex nonlinear material models; nevertheless it already represents a first approximation to describe the considered problem, providing us with the possibility to test the effectivity of the proposed MSRB preconditioning in an involved context.

5.4.1 Test case setting

We consider the time interval $(0, T]$, $T > 0$ and the open domain Ω shown in Figure 5.12a, such that $\partial\Omega = \Gamma_D \cup \Gamma_{in} \cup \Gamma_{ex}$, and $\Gamma_D = \Gamma_D^1 \cup \Gamma_D^2 \cup \Gamma_D^3$. The following parametrized unsteady linear elasticity problem describes the displacement $\vec{d}(\boldsymbol{\mu}) = \vec{d}(t;\boldsymbol{\mu})$ of the arterial tissue with respect to a reference configuration

$$\begin{cases} \rho \dfrac{\partial^2 \vec{d}(\boldsymbol{\mu})}{\partial t^2} - \nabla \cdot \mathbf{\Pi}(\vec{d}(\boldsymbol{\mu}); \boldsymbol{\mu}) = \vec{0} & \text{in } \Omega \times (0, T] \\ \vec{d}(\boldsymbol{\mu}) = \vec{0} & \text{on } \Gamma_D \times (0, T] \\ \mathbf{\Pi}(\vec{d}(\boldsymbol{\mu}); \boldsymbol{\mu})\vec{n} = -p(t)\vec{n} & \text{on } \Gamma_{in} \times (0, T] \\ \mathbf{\Pi}(\vec{d}(\boldsymbol{\mu}); \boldsymbol{\mu})\vec{n} + k_s \vec{d}(\boldsymbol{\mu}) + c_s \dfrac{\partial \vec{d}(\boldsymbol{\mu})}{\partial t} = 0 & \text{on } \Gamma_{ex} \times (0, T], \end{cases} \tag{5.9}$$

with $\vec{d}(\boldsymbol{\mu})|_{t=0} = \frac{\partial \vec{d}(\boldsymbol{\mu})}{\partial t}|_{t=0} = \vec{0}$ as initial conditions. Here $\rho = 1.2\text{gcm}^{-3}$ is the material density, $\vec{n}$ is the outer normal vector,

$$\mathbf{\Pi}(\vec{d}(\boldsymbol{\mu}); \boldsymbol{\mu}) = 2\tilde{\mu}\boldsymbol{\varepsilon}(\vec{d}(\boldsymbol{\mu})) + \tilde{\lambda}\nabla \cdot \vec{d}(\boldsymbol{\mu})\mathbf{I}, \tag{5.10}$$

is the stress tensor and

$$\boldsymbol{\varepsilon}(\vec{d}(\boldsymbol{\mu})) = \frac{1}{2}(\nabla\vec{d}(\boldsymbol{\mu}) + \nabla\vec{d}(\boldsymbol{\mu})^T) \tag{5.11}$$

is the strain tensor. The coefficients $\tilde{\mu}$ and $\tilde{\lambda}$ are the Lamé coefficients, which can be expressed in terms of the Young's modulus E and the Poisson's ration ν as

$$\tilde{\lambda} = \frac{E\nu}{(1+\nu)(1-2\nu)}, \qquad \tilde{\mu} = \frac{E}{2(1+\nu)}.$$

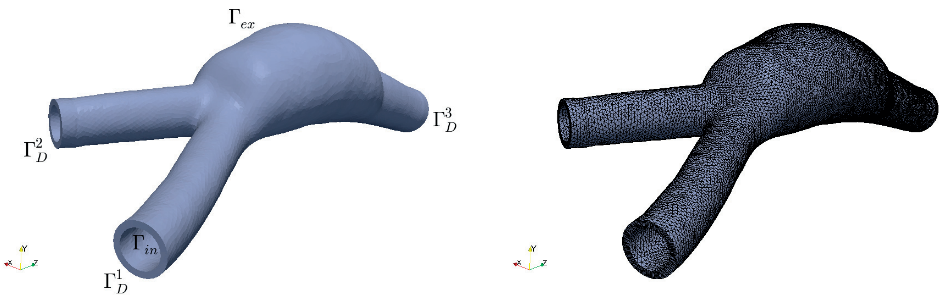
To start with, we divide the domain Ω in two regions: Ω_A, representing the dome of the aneurysm sac, and $\Omega_H = \Omega\backslash\Omega_A$; they are shown in Figure 5.12c. The different behaviors of the tissue in the two regions is modeled by a piecewise Young's modulus coefficient, that is

$$E = E_A\mathcal{X}_A + E_H\mathcal{X}_H \tag{5.12}$$

where E_A, E_H are positive real numbers and $\mathcal{X}_A$ and $\mathcal{X}_H$ are the indicator functions on Ω_A, Ω_H, respectively. Regarding the boundary conditions, the artery bifurcation is clamped at the inlet and outlet and a stress $p(t;\boldsymbol{\mu}) = \mu_p p_0(t)$ is prescribed on the inner surface Γ_{in} to simulate the arterial pressure, where μ_p is a scaling parameter and $p_0(t)$ is a reference pressure reported in Figure 5.12d, which has been computed through a fluid structure interaction model in [Malossi and Bonnemain, 2013] and has been taken in this work as reference arterial pressure at the abdominal aorta. An important aspect to consider when dealing with the structural dynamics of arterial tissue is the boundary condition prescribed to the external boundary Γ_{ex}. Indeed, one should ideally model the contact problems between the artery and the surrounding organs; however, the computation of the interaction with the environment may lead to unbearable computational costs. In the literature, it has been proposed to simulate the presence of the surrounding environment by employing a Robin boundary condition for viscoelastic materials, see (5.9), which is set on Γ_{ex}. The values of the coefficients k_s and c_s are not trivial. It has been shown in [Malossi and Bonnemain, 2013] that a practical option consists in choosing $c_s = k_s/10$, even if in general the value of k_s itself is not easy to be chosen.

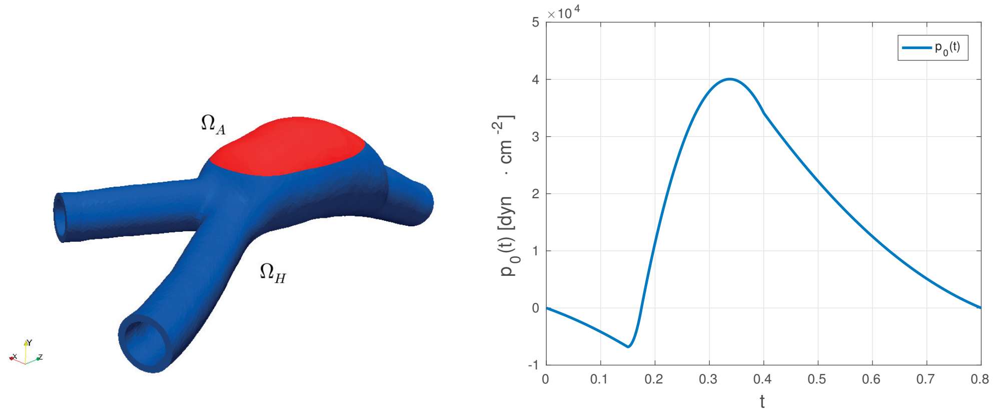
We parametrize the model with respect to k_s, the Young's modulus in Ω_A to simulate different aneurysm behaviors and the factor μ_p, ending up with the parameter vector

$$\boldsymbol{\mu} = (E_A, \mu_p, k_s) \in \mathcal{D} = [10^6, 3 \cdot 10^6] \times [0.875, 1.25] \times [6 \cdot 10^4, 10^5], \tag{5.13}$$



(a) Computational domain Ω with boundaries.

(b) Computational mesh.



(c) Location Ω_A of the dome of the aneurysm sac (red).

(d) $p_0(t)$ as function of time.

Figure 5.12 – Domain Ω, boundary flags and computational domain.

where the value of E_A is expressed in [dyn cm^{-2}] and the ones for k_s in [dyn cm^{-3}]. Instead E_H is set to $3 \cdot 10^6$dyn cm^{-2} and $\nu = 0.48$ as in [Malossi and Bonnemain, 2013].

5.4.2 Variational formulation and FE discretization

We introduce the variational space $V = (H^1_{\Gamma_D}(\Omega))^3$ and by integrating by parts we come up with the following variational formulation: for each $\boldsymbol{\mu}$ and for each $t \in (0, T]$, we seek $\vec{d}(\boldsymbol{\mu}) \in V$ such that $\vec{d}(0, \boldsymbol{\mu}) = \frac{\partial \vec{d}(\boldsymbol{\mu})}{\partial t}\big|_{t=0} = \vec{0}$ and

$$\begin{aligned} \int_\Omega \Big(\frac{\partial^2 \vec{d}(\boldsymbol{\mu})}{\partial t^2} \cdot \vec{v} + \mathbf{\Pi}(\vec{d}(\boldsymbol{\mu}); \boldsymbol{\mu}) : \vec{v} \Big) d\Omega + \int_{\Gamma_{ex}} \Big(k_s \vec{d}(\boldsymbol{\mu}) \cdot \vec{v} \Big) d\Gamma_{ex} \\ + \int_{\Gamma_{ex}} \Big(c_s \frac{\partial \vec{d}(\boldsymbol{\mu})}{\partial t} \cdot \vec{v} \Big) d\Gamma_{ex} = - \int_{\Gamma_{ex}} \Big(p(t) \vec{n} \cdot \vec{v} \Big) d\Gamma_{ex} \qquad \forall \vec{v} \in V. \end{aligned} \tag{5.14}$$

By introducing a proper FE subspace $V_h \subset V$ and its basis $\left\{\vec{\varphi}_i\right\}_{i=1}^{N_h}$, we discretize problem (5.14) as follows: for each $\boldsymbol{\mu}$ and for each $t \in (0,T]$, we seek $\vec{d}_h(\boldsymbol{\mu}) \in V_h$ such that

$$\begin{aligned}\int_\Omega \Big(\frac{\partial^2 \vec{d}_h(\boldsymbol{\mu})}{\partial t^2} \cdot \vec{v}_h + \boldsymbol{\Pi}(\vec{d}_h(\boldsymbol{\mu}); \boldsymbol{\mu}) : \nabla \vec{v}_h \Big) d\Omega + \int_{\Gamma_{ex}} \Big(k_s \vec{d}_h(\boldsymbol{\mu}) \cdot \vec{v} \Big) d\Gamma_{ex} \qquad (5.15)\\ + \int_{\Gamma_{ex}} \Big(c_s \frac{\partial \vec{d}_h(\boldsymbol{\mu})}{\partial t} \cdot \vec{v} \Big) d\Gamma_{ex} = - \int_{\Gamma_{ex}} \Big(p(t) \vec{n} \cdot \vec{v} \Big) d\Gamma_{ex} \qquad \forall \vec{v} \in V,\end{aligned}$$

which corresponds to solving the following algebraic dynamical system

$$\mathbf{M}(\boldsymbol{\mu}) \frac{d^2 \mathbf{d}(\boldsymbol{\mu})}{dt^2} + \mathbf{M}^{\Gamma_{ex}}(\boldsymbol{\mu}) \frac{d\mathbf{d}(\boldsymbol{\mu})}{dt} + \mathbf{A}(\boldsymbol{\mu}) \mathbf{d}(\boldsymbol{\mu}) = \mathbf{p}(\boldsymbol{\mu}) \qquad t \in (0,T], \qquad (5.16)$$

with $\mathbf{d}(0;\boldsymbol{\mu}) = \frac{d\mathbf{d}(\boldsymbol{\mu})}{dt}\big|_{t=0} = \mathbf{0}$. The FE vector $\mathbf{d}(\boldsymbol{\mu})$ is the vector representation of $\vec{d}_h(\boldsymbol{\mu})$, the matrices $\mathbf{M}(\boldsymbol{\mu})$, $\mathbf{M}^{\Gamma_{ex}}(\boldsymbol{\mu})$ and $\mathbf{A}(\boldsymbol{\mu})$ such as

$$\begin{aligned}(\mathbf{M}(\boldsymbol{\mu}))_{i,j} &= \int_\Omega \vec{\varphi}_i \cdot \vec{\varphi}_j d\Omega \qquad (\mathbf{M}^{\Gamma_{ex}}(\boldsymbol{\mu}))_{i,j} = c_s \int_{\Gamma_{ex}} \vec{\varphi}_i \cdot \vec{\varphi}_j d\Omega\\ (\mathbf{A}(\boldsymbol{\mu}))_{i,j} &= \int_\Omega \Big(\boldsymbol{\Pi}(\vec{\varphi}_j; \boldsymbol{\mu}) : \vec{\varphi}_i \Big) d\Omega + k_s \int_{\Gamma_{ex}} \vec{\varphi}_j \cdot \vec{\varphi}_i d\Gamma_{ex}, \qquad i,j = 1, \dots, N_h\end{aligned}$$

while $\mathbf{p}(\boldsymbol{\mu})$ encodes the contribution of the arterial pressure

$$(\mathbf{p}(\boldsymbol{\mu}))_i = - \int_{\Gamma_{in}} \Big(p(t) \vec{n} \cdot \vec{\varphi}_i \Big) d\Gamma_{ex}, \qquad i = 1, \dots, N_h.$$

Problem (5.16) is then discretized in time by employing the BDF scheme of order 1 or 2 for second order ordinary differential equations. Towards this goal, the first derivative is approximated as in (1.19), whereas the second derivative is approximated as

$$\frac{d^2 \mathbf{d}(\boldsymbol{\mu})}{dt^2} \approx \frac{\alpha_2 \mathbf{d}^{n+1}(\boldsymbol{\mu}) - \mathbf{d}^{n,\sigma_2}(\boldsymbol{\mu})}{\Delta t^2}, \qquad (5.17)$$

where the BDF scheme of order $\sigma_2 = 1,\ 2$ is identified by

$$\mathbf{d}^{n,\sigma_2}(\boldsymbol{\mu}) = \begin{cases} 2\mathbf{d}^n(\boldsymbol{\mu}) - \mathbf{d}^{n-1}(\boldsymbol{\mu}), & n \geq 0 \text{ and } \sigma_2 = 1 \\ 5\mathbf{d}^n(\boldsymbol{\mu}) - 4\mathbf{d}^{n-1}(\boldsymbol{\mu}) + \mathbf{d}^{n-2}(\boldsymbol{\mu}), & n \geq 1 \text{ and } \sigma_2 = 2 \end{cases} \qquad (5.18)$$

and $\alpha_2 = 1$ for $\sigma_2 = 1$ and $\alpha_2 = 2$ for $\sigma_2 = 2$, yielding the following sequence of parametrized linear systems to be solved for any $n = 0, \dots, N_t - 1$

$$\begin{aligned}\Big(\frac{\alpha_2}{\Delta t^2} \mathbf{M}(\boldsymbol{\mu}) + \frac{\alpha_1}{\Delta t} \mathbf{M}^{\Gamma_{ex}}(\boldsymbol{\mu}) + \mathbf{A}(\boldsymbol{\mu}) \Big) \mathbf{d}^{n+1}(\boldsymbol{\mu}) = \qquad (5.19)\\ \mathbf{M}(\boldsymbol{\mu}) \frac{\mathbf{d}^{n,\sigma_2}(\boldsymbol{\mu})}{\Delta t^2} + \mathbf{M}^{\Gamma_{ex}}(\boldsymbol{\mu}) \frac{\mathbf{d}^{n,\sigma_1}(\boldsymbol{\mu})}{\Delta t} + \mathbf{p}^{n+1}(\boldsymbol{\mu}),\end{aligned}$$

supplied with null initial conditions. The BDF order σ_1, for the discretization of the first derivative, is chosen as equal to σ_2.

5.4.3 Numerical results with MSRB preconditioner

Even if system (5.19) arises from the spatial and time discretization of an elastodynamics problem as (5.9), for the sake of preconditioning with the MSRB technique, it can be treated as the one resulting from the spatial and time discretization of parabolic problems, see Section 2.5. In the numerical results presented below, we employ the computational grid shown in Figure (5.12b) and either $\mathcal{P}_1$ or $\mathcal{P}_2$ finite elements basis functions, leading to $N_h = 115'542$ and $N_h = 856'973$, respectively. We employ 32 computing cores in the former case and 256 in the latter. Here $T = 0.8$ seconds and $\Delta t = 0.01$ is set as time step.

An Additive Schwarz preconditioner, with either 1 or 2 layers of overlapped elements and denoted $\mathbf{P}_{\mathrm{AS}\,1}$ and $\mathbf{P}_{\mathrm{AS}\,2}$, respectively, is chosen as fine component of the MSRB preconditioner; the RB coarse operators are trained on 50 parameter instances, for which $\mathbf{P}_{\mathrm{AS}\,2}$ is used to compute the corresponding snapshots, randomly chosen in $\mathcal{D}$. This choice yields a total of 4000 snapshots if both parameter and time variability are considered, therefore, in order to speed up the subsequent computation of the PODs used to build the RB spaces, we employ a time slab partitioning approach by dividing the time interval $[0, T]$ in $\mathcal{S} = 4$ time slabs, each of length $\Delta\tau = 0.2$.

On each time slab, the MSRB preconditioner is built during the offline phase by selecting a final tolerance $\varepsilon_r = 10^{-8}$ and by using a fixed dimension approach with $N_k = 25$ RB functions for each RB coarse correction. Such a setting yields the construction of $L_s = 4$ RB coarse operators in the time slabs $s = 0, 1, 2$ and with $L_s = 3$, for $s = 3$. A smaller number of RB coarse corrections is needed in the last slab to reach the same tolerance due to the unload in the last part of the time interval (cf. Figure 5.12d). Furthermore, an affine decomposition of the matrices in (5.19) is readily available, therefore MDEIM is not employed to recover an approximate one in this case. The system is solved online with FGMRES up to a tolerance of 10^{-6} on the rescaled FE residual, the results in terms of iteration count and computational times of the computation are reported in Table (5.4) and (5.5), when using $\mathbf{P} = \mathbf{P}_{\mathrm{AS}\,2}$ and $\mathbf{P} = \mathbf{P}_{\mathrm{AS}\,1}$, respectively, as fine grid component. The results are compared with the option of using $\mathbf{P}_{\mathrm{AS}\,2}$ alone, which is the one providing the better performance compared to $\mathbf{P}_{\mathrm{AS}\,1}$. As a matter of fact, each linear system is solved on average in 2 or 3 iterations and in a very competitive computational time. In particular, choosing $\mathbf{P} = \mathbf{P}_{\mathrm{AS}\,1}$ as fine grid component results in the most convenient choice in this respect, leading to a speed up of 6.7 for $\mathcal{P}_2$ FE basis functions, if compared with the case in which $\mathbf{P}_{\mathrm{AS}\,2}$ is used standalone. The break-even point (BEP), which is

Table 5.4 – Results with MSRB preconditioner with ($N_k = 25$), $\mathcal{S} = 4$ time slabs and $\mathbf{P} = \mathbf{P}_{\mathrm{AS}\,2}$. Computational times are expressed in seconds and the online time refer to the solution of one time step.

	$t^{\mathrm{onl}}_{\mathrm{MSRB}}(It)$	$t_{\mathrm{AS2}}(It)$	t_{off}	SPEEDUP	BEP
$\mathcal{P}_1$	1.06 (2)	2.86	11096.8	2.7	78
$\mathcal{P}_2$	3.44 (3)	14.43	45059.4	4.2	52

Table 5.5 – Results with MSRB preconditioner with ($N_k = 25$), $\mathcal{S} = 4$ time slabs and $\mathbf{P} = \mathbf{P}_{\mathrm{AS}\,1}$. Computational times are expressed in seconds and the online time refer to the solution of one time step.

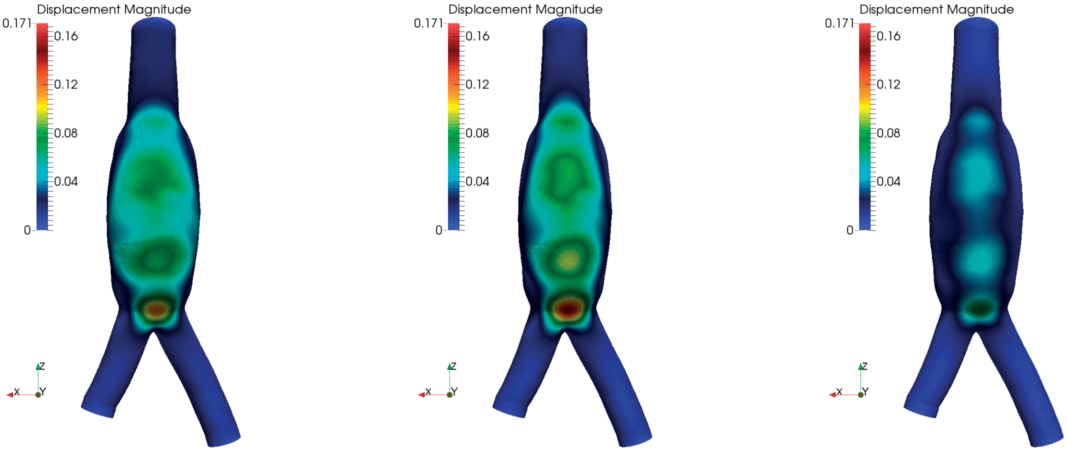
	$t^{\mathrm{onl}}_{\mathrm{MSRB}}(It)$	$t_{\mathrm{AS2}}(It)$	t_{off}	SPEEDUP	BEP
$\mathcal{P}_1$	0.81 (2)	2.86 (28)	11094.7	3.5	68
$\mathcal{P}_2$	2.16 (2)	14.43 (43)	46225.4	6.7	48

defined as

$$\mathrm{BEP} = \frac{t_{\mathrm{off}}}{N_t(t^{\mathrm{onl}}_{\mathrm{MSRB}} - t_{\mathrm{AS2}})},$$

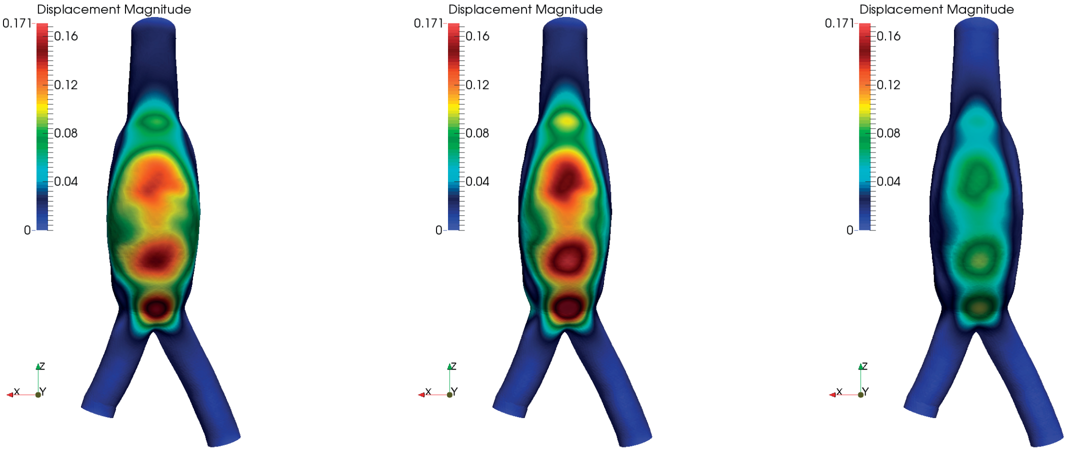
is reported in Table (5.4) and (5.5) as well, and in all considered cases it ranges from 48 to 78 online evaluations, confirming that the larger the FE dimension, the more convenient the use of the MSRB preconditioner.

Numerical results for some selected parameter instances at different times are reported in Figure 5.13 and 5.14, where the influence of the parameter is clearly highlighted. The region undergoing larger displacement is located in correspondence with the bifurcation, with different intensity according to the value of $\boldsymbol{\mu}$. To further investigate the role of parameters, we report in Figure 5.15 the time evolution of the displacement magnitude in three different location, two of them (P1 and P2) located in the region with a parametrized Young's modulus E_A. By comparing Figure 5.15a and 5.15c, we see how the dynamics in P1 and P2 changes according to the value E_A, the higher its value, the smaller the corresponding displacement magnitude. The values of k_s and μ_p also play a role: by taking Figure 5.15a and 5.15d, we see that by decreasing k_s, the displacement magnitude increases of about 10%, moreover, by comparing Figure 5.15c and 5.15f where k_s is decreased and μ_p increased at the same time, an even larger displacement is found.



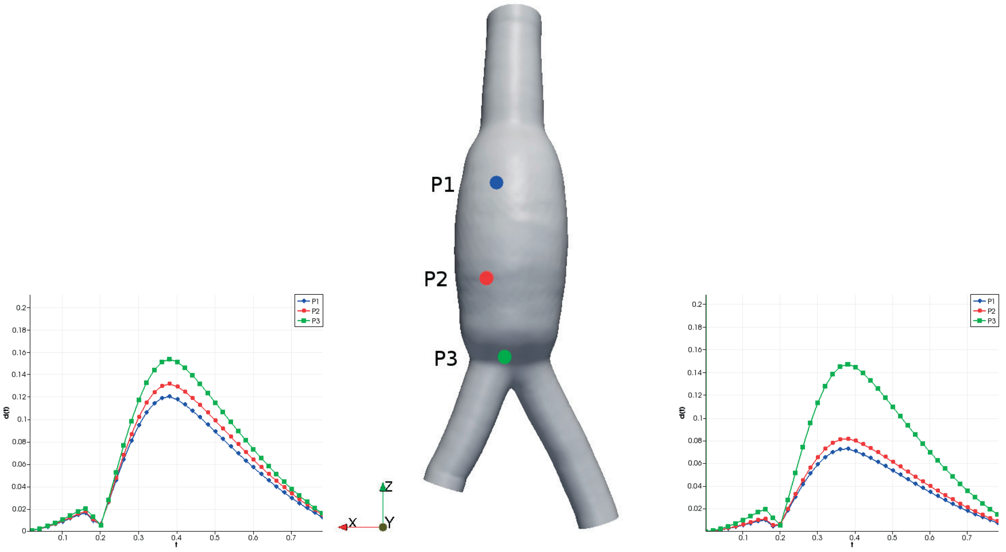
(a) $t = 0.3$.
(b) $t = 0.44$.
(c) $t = 0.6$.

Figure 5.13 – Displacement for $\boldsymbol{\mu} = (1.85 \cdot 10^6, 1.0625, 1.05 \cdot 10^5)$ at different times.

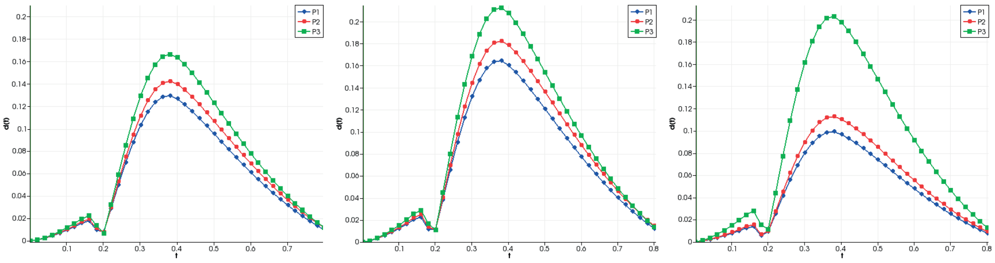


(a) $t = 0.3$.
(b) $t = 0.44$.
(c) $t = 0.6$.

Figure 5.14 – Displacement for $\boldsymbol{\mu} = (1.05 \cdot 10^6, 1.2, 7 \cdot 10^4)$ at different times.



(a) $\boldsymbol{\mu} = (1.05 \cdot 10^6, 1.0625, 1.05 \cdot 10^5)$.
(b) P1, P2, P3 locations.
(c) $\boldsymbol{\mu} = (2.95 \cdot 10^6, 1.0625, 1.05 \cdot 10^5)$.



(d) $\boldsymbol{\mu} = (1.05 \cdot 10^6, 1.0625, 9 \cdot 10^4)$.
(e) $\boldsymbol{\mu}(1.05 \cdot 10^6, 1.2, 7 \cdot 10^4)$.
(f) $\boldsymbol{\mu}(2.95 \cdot 10^6, 1.2, 7 \cdot 10^4)$.

Figure 5.15 – Time evolution of magnitude displacement for different parameter values in locations P1, P2, P3.

6 Conclusions

In this dissertation we have proposed a new two-level preconditioner based on the combination of a RB coarse component and a fine grid preconditioner for large-scale linear systems arising from the FE discretization of parametrized PDEs. The driving idea lies in building a sequence of RB low-rank solvers which are tailored to provide an accurate approximation to the error equation originating at each step from the chosen iterative method, whence the name MSRB preconditioner. This strategy provides an iteration-dependent operator enabling to tune the decay of the error at each step of the iterative method, by properly selecting the desired accuracy. We have initially outlined the MSRB preconditioner and analyzed its properties in the amenable case of affinely parametrized second order elliptic problems.

The employed RB coarse operators are obtained by (Petrov) Galerkin projection onto the RB subspaces, and for their practical construction the affine property of the systems has been suitably exploited; on the other hand, when such assumption is not verified, an approximated RB coarse operator has been devised by employing MDEIM to affinely approximate the FE matrix. The resulting preconditioner depends on the affine approximation in a milder way than standard RB methods, thus overcoming the bottleneck given by the nonaffine parameter dependence. The main reason behind this feature lies in the fact that at each iteration even a coarse affine representation of the FE matrix yields a negligible error compared to the one entailed by the RB approximation, thus without affecting the overall accuracy.

Extensions have been proposed for time-dependent and (nonlinear) saddle-point problems in CFD; for the former, a version exploiting the partitioning of the time interval in time slabs has been devised, whereas a suitable RB formulation, relying on either an enriched velocity RB space or a Petrov Galerkin projection, has been used for the Stokes equations. Furthermore, the unsteady parametrized Navier-Stokes (NS) equations have been taken into account: we have developed a MSRB preconditioning strategy which minimizes the computational complexity by employing a time slab formulation to treat the time dependence and a double POD algorithm to compute an affine approximation of the nonlinear convective term.

Several numerical examples have been presented, showing the generality and computational efficiency of the proposed methodology in a variety of contexts: linear affine and nonaffine second order elliptic PDEs, parabolic and elastodynamics problems, (non)linear saddle-point systems. In all cases, the MSRB preconditioning strategy works well in involved scenarios, regarding both parameter dependence and physics. In particular, we summarize in the following the situations where the MSRB preconditioner has shown to be particularly effective.

- *Large dimension of the FE linear system*: the RB coarse operators are not affected by the underlying discretization (neither by the grid size nor by the local polynomial degree); their accuracy and efficiency solely depend on the given physical problem and its parameter dependence.
- *Large-scale simulations*: even when the number of cores gets large, the RB coarse operators feature a very small dimension, making them efficiently applicable in an HPC environment.
- *Fine preconditioner with an expensive cost per iteration*: the application of some fine grid preconditioners, such as the SIMPLE in CFD applications, may lead to heavy computational costs; combining them with a RB coarse operator in a MSRB preconditioner allows to cut such cost by reaching convergence in only few iterations.
- *Involved (and possibly nonaffine) parameter dependence*: the accuracy and efficiency of standard RB methods deteriorate when facing problems with a severe parameter dependence, due to either a nonaffine nature or a wide parameter variability, such as the advection-diffusion cases presented in Chapter 2 or the Stokes equations in parametrized domains tacked in Chapter 3. This bottleneck is overcome when using a MSRB preconditioner with a fixed dimension approach to construct the RB coarse operators. This limits indeed the number of RB functions used within the coarse operator and allows the use of a coarse affine approximation without compromising the overall efficiency.

With the goal of constructing the MSRB preconditioner, other contributions related to RB methods, which are summarized in the following, have been developed.

- *Algebraic LSRB method for linear saddle-point problems*: we have developed and analyzed a new RB method which is a suitable modification of the standard LSRB method. The applicability of this latter is extended by properly substituting the matrix-norm which defines the RB test space. The resulting RB formulation provides a well-posed RB solver whose application leads to a more efficient and accurate method than the ones commonly used in this context. The proposed method has been exploited for the reduction of parametrized Stokes equations, however its formulation and applicability have a general scope for parametrized (linear and nonlinear) saddle-point problems.

- *A ROM framework for the unsteady NS equations in parametrized geometries*: we have extended the state-of-the-art techniques for the reduction of unsteady parametrized NS equations by proposing a hyper-reduction strategy which exploits a double POD algorithm to efficiently treat the nonlinear convective term. Mesh motion techniques have been embedded to effectively deal with parametrized geometries. The resulting technique has been successfully applied to three-dimensional flows.

Overall, the techniques developed in this thesis represent a powerful tool to tackle parametrized systems of large size, yet further extensions can be devised in several directions. In particular, the results obtained on advection-diffusion equations suggest that the proposed MSRB preconditioners can be effectively used in the field, e.g., of parametrized multi-scale modeling in porous media, where the problem is typically characterized by heterogeneous coefficients with high variations.

Furthermore, from the viewpoint of CFD, different improvements can be undertaken. At first, the developed NS-HROM framework for the computational reduction the NS equations in parametrized domains can be theoretically analyzed to shed light on the impact of the use of successive RB approximations for the deformation and the fluid flow, possibly considering the use of a different RB formulation as well. In this perspective, an adaptation to the NS equations of the algebraic LSRB method, proposed here for linear steady saddle-point problems, can be investigated.

On the other hand, when considering the MSRB preconditioners for the NS equations, at first a detailed theoretical analysis can be carried out to clarify the implications of the approximations used to efficiently construct the preconditioner; secondly, a larger range of applications can be considered. Indeed, MSRB preconditioners have been employed in this work on laminar flows with relatively low Reynolds number, however not preventing their use, in principle, to treat more involved flow regimes. On a parallel track, the entire scope can also be further expanded to the case of time dependent deformations, embedding the use of mesh-motion techniques which comply with time variability. Such a development would allow to set the way, in a longer term perspective, to deal with parametrized fluid structure interaction (and in general multi-physics) problems, for which the construction of efficient and reliable ROM techniques are, at this time, still under investigation.

Bibliography

[Abdulle and Budáč, 2015] Abdulle, A. and Budáč, O. (2015). A Petrov–Galerkin reduced basis approximation of the Stokes equation in parameterized geometries. *C. R. Math. Acad. Sci. Paris*, 353(7):641–645.

[Ahuja et al., 2012] Ahuja, K., de Sturler, E., Gugercin, S., and Chang, E. R. (2012). Recycling BiCG with an application to model reduction. *SIAM Journal on Scientific Computing*, 34(4):1925–1949.

[Amsallem and Haasdonk, 2016] Amsallem, D. and Haasdonk, B. (2016). PEBL-ROM: Projection-error based local reduced-order models. *Advanced Modeling and Simulation in Engineering Sciences*, 3(1):6.

[Antiga et al., 2008] Antiga, L., Piccinelli, M., Botti, L., Ene-Iordache, B., Remuzzi, A., and Steinman, D. (2008). An image-based modeling framework for patient-specific computational hemodynamics. *Medical & Biological Engineering & Computing*, 46(11):1097–1112.

[Ascher and Petzold, 1998] Ascher, U. M. and Petzold, L. R. (1998). *Computer methods for ordinary differential equations and differential-algebraic equations*, volume 61. Siam.

[Axelsson and Neytcheva, 2003] Axelsson, O. and Neytcheva, M. (2003). Preconditioning methods for linear systems arising in constrained optimization problems. *Numerical Linear Algebra with Applications*, 10(1-2):3–31.

[Babuška, 1971] Babuška, I. (1971). Error-bounds for finite element method. *Numerische Mathematik*, 16(4):322–333.

[Bache et al., 2010] Bache, E., Vega, J., and Velazquez, A. (2010). Model reduction in the back step fluid–thermal problem with variable geometry. *International Journal of Thermal Sciences*, 49(12):2376–2384.

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

[Ballarin et al., 2015] Ballarin, F., Manzoni, A., Quarteroni, A., and Rozza, G. (2015). Supremizer stabilization of POD–Galerkin approximation of parametrized steady incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Engineering*, 102(5):1136–1161.

[Barrault et al., 2004] Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T. (2004). An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique Académie des Sciences Paris*, 339(9):667–672.

[Bazilevs et al., 2007] Bazilevs, Y., Calo, V. M., Tezduyar, T. E., and Hughes, T. J. (2007). Yzβ discontinuity capturing for advection-dominated processes with application to arterial drug delivery. *International Journal for Numerical Methods in Fluids*, 54(6-8):593–608.

[Bellavia et al., 2011] Bellavia, S., De Simone, V., Di Serafino, D., and Morini, B. (2011). Efficient preconditioner updates for shifted linear systems. *SIAM Journal on Scientific Computing*, 33(4):1785–1809.

[Benner and Feng, 2011] Benner, P. and Feng, L. (2011). On recycling Krylov subspaces for solving linear systems with successive right-hand sides with applications in model reduction. In *Model Reduction for Circuit Simulation. P. Benner, M. Hinze and E.J.W. Ter Maten, eds., Lecture Notes in Electrical Engineering*, volume 74, pages 125–140. Springer, Dordrecht.

[Benzi and Bertaccini, 2003] Benzi, M. and Bertaccini, D. (2003). Approximate inverse preconditioning for shifted linear systems. *BIT Numerical Mathematics*, 43(2):231–244.

[Benzi et al., 2005] Benzi, M., Golub, G. H., and Liesen, J. (2005). Numerical solution of saddle point problems. *Acta numerica*, 14:1–137.

[Benzi and Wathen, 2008] Benzi, M. and Wathen, A. J. (2008). Some preconditioning techniques for saddle point problems. In *Model order reduction: theory, research aspects and applications*, pages 195–211. Springer Berlin Heidelberg.

[Bergmann et al., 2009] Bergmann, M., Bruneau, C.-H., and Iollo, A. (2009). Enablers for robust POD models. *Journal of Computational Physics*, 228(2):516 – 538.

[Berkooz et al., 1993] Berkooz, G., Holmes, P., and Lumley, J. L. (1993). The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575.

[Bertaccini and Durastante, 2016] Bertaccini, D. and Durastante, F. (2016). Interpolating preconditioners for the solution of sequence of linear systems. *Computers & Mathematics with Applications*, 72(4):1118–1130.

[Boffi et al., 2013] Boffi, D., Brezzi, F., Fortin, M., et al. (2013). *Mixed finite element methods and applications*, volume 44. Springer.

[Bramble et al., 1990] Bramble, J. H., Pasciak, J. E., and Xu, J. (1990). Parallel multilevel preconditioners. *Mathematics of Computation*, 55(191):1–22.

[Brandt, 1977] Brandt, A. (1977). Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138):333–390.

[Brenan et al., 1995] Brenan, K. E., Campbell, S. L., and Petzold, L. R. (1995). *Numerical solution of initial-value problems in differential-algebraic equations.* SIAM.

[Brenner and Scott, 2007] Brenner, S. and Scott, R. (2007). *The mathematical theory of finite element methods*, volume 15. Springer Science & Business Media.

[Brezzi, 1974] Brezzi, F. (1974). On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 8(R2):129–151.

[Brezzi and Bathe, 1990] Brezzi, F. and Bathe, K.-J. (1990). A discourse on the stability conditions for mixed finite element formulations. *Computer methods in applied mechanics and engineering*, 82(1-3):27–57.

[Briggs et al., 2000] Briggs, W. L., Henson, V. E., and McCormick, S. F. (2000). *A multigrid tutorial.* SIAM.

[Buffa et al., 2012] Buffa, A., Maday, Y., Patera, A. T., Prud'homme, C., and Turinici, G. (2012). A priori convergence of the greedy algorithm for the parametrized reduced basis method. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46(3):595–603.

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

[Canuto et al., 2012] Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A. (2012). *Spectral methods in fluid dynamics.* Springer Science & Business Media.

[Caputo et al., 2013] Caputo, M., Chiastra, C., Cianciolo, C., Cutrì, E., Dubini, G., Gunn, J., Keller, B., Migliavacca, F., and Zunino, P. (2013). Simulation of oxygen transfer in stented arteries and correlation with in-stent restenosis. *International journal for numerical methods in biomedical engineering*, 29(12):1373–1387.

[Carlberg et al., 2011] Carlberg, K., Bou-Mosleh, C., and Farhat, C. (2011). Efficient nonlinear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181.

[Carlberg et al., 2013] Carlberg, K., Farhat, C., Cortial, J., and Amsallem, D. (2013). The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647.

[Carlberg et al., 2016] Carlberg, K., Forstall, V., and Tuminaro, R. (2016). Krylov-subspace recycling via the pod-augmented conjugate-gradient method. *SIAM Journal on Matrix Analysis and Applications*, 37(3):1304–1336.

[Chapman and Saad, 1997] Chapman, A. and Saad, Y. (1997). Deflated and augmented Krylov subspace techniques. *Numerical linear algebra with applications*, 4(1):43–66.

[Chaturantabut and Sorensen, 2010] Chaturantabut, S. and Sorensen, D. C. (2010). Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764.

[Ciarlet, 2002] Ciarlet, P. G. (2002). *The finite element method for elliptic problems*. SIAM.

[Colciago, 2014] Colciago, C. M. (2014). *Reduced order fluid-structure interaction models for haemodynamics applications*. PhD thesis, EPFL.

[Colciago et al., 2014] Colciago, C. M., Deparis, S., and Quarteroni, A. (2014). Comparisons between reduced order models and full 3d models for fluid–structure interaction problems in haemodynamics. *Journal of Computational and Applied Mathematics*, 265:120–138.

[Coppola and Caro, 2008] Coppola, G. and Caro, C. (2008). Oxygen mass transfer in a model three-dimensional artery. *Journal of The Royal Society Interface*, 5(26):1067–1075.

[Cortes et al., 2018] Cortes, G. D., Vuik, K., and Jansen, J. (2018). On pod-based deflation vectors for dpcg applied to porous media problems. *Journal of Computational and Applied Mathematics*, 330:193–213.

[Dahmen et al., 2012] Dahmen, W., Huang, C., Schwab, C., and Welper, G. (2012). Adaptive Petrov–Galerkin methods for first order transport equations. *SIAM Journal on Numerical Analysis*, 50(5):2420–2445.

[Dal Santo et al., 2017a] Dal Santo, N., Deparis, S., and Manzoni, A. (2017a). A numerical investigation of multi space reduced basis preconditioners for parametrized elliptic advection-diffusion equations. *Communications in Applied and Industrial Mathematics*, 8(1):282–297.

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

[Dal Santo et al., 2018a] Dal Santo, N., Deparis, S., Manzoni, A., and Quarteroni, A. (2018a). Multi space reduced basis preconditioners for large-scale parametrized PDEs. *Mathicse 32.2016. Accepted for publication on SIAM Journal on Scientific Computing.*

[Dal Santo et al., 2018b] Dal Santo, N., Deparis, S., Manzoni, A., and Quarteroni, A. (2018b). Multi space reduced basis preconditioners for parametrized Stokes equations. Technical Report 03.2018, MATHICSE–EPFL.

[De Sturler, 1996] De Sturler, E. (1996). Nested Krylov methods based on GCR. *Journal of Computational and Applied Mathematics*, 67(1):15–41.

[De Sturler, 1999] De Sturler, E. (1999). Truncation strategies for optimal Krylov subspace methods. *SIAM Journal on Numerical Analysis*, 36(3):864–889.

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

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

[Deparis and Rozza, 2009] Deparis, S. and Rozza, G. (2009). Reduced basis method for multi-parameter-dependent steady Navier-Stokes equations: Applications to natural convection in a cavity. *Journal of Computational Physics*, 228(12):4359–4378.

[Dohrmann, 2003] Dohrmann, C. R. (2003). A preconditioner for substructuring based on constrained energy minimization. *SIAM Journal on Scientific Computing*, 25(1):246–258.

[Drohmann et al., 2012a] Drohmann, M., Haasdonk, B., and Ohlberger, M. (2012a). Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation. *SIAM Journal on Scientific Computing*, 34(2):A937–A969.

[Drohmann et al., 2012b] Drohmann, M., Haasdonk, B., and Ohlberger, M. (2012b). Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation. *SIAM Journal on Scientific Computing*, 34(2):A937–A969.

[Díez et al., 2017] Díez, P., Zlotnik, S., and Huerta, A. (2017). Generalized parametric solutions in Stokes flow. *Computer Methods in Applied Mechanics and Engineering*, 326:223 – 240.

[Eiermann et al., 2000] Eiermann, M., Ernst, O. G., and Schneider, O. (2000). Analysis of acceleration strategies for restarted minimal residual methods. *Journal of Computational and Applied Mathematics*, 123(1):261 – 292. Numerical Analysis 2000. Vol. III: Linear Algebra.

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

[Elman et al., 2008] Elman, H., Howle, V. E., Shadid, J., Shuttleworth, R., and Tuminaro, R. (2008). A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 227(3):1790–1808.

[Elman and Forstall, 2015] Elman, H. C. and Forstall, V. (2015). Preconditioning techniques for reduced basis methods for parameterized elliptic partial differential equations. *SIAM Journal on Scientific Computing*, 37(5):S177–S194.

[Elman and Forstall, 2017] Elman, H. C. and Forstall, V. (2017). Numerical solution of the parameterized steady-state navier–stokes equations using empirical interpolation methods. *Computer Methods in Applied Mechanics and Engineering*, 317:380 – 399.

[Elman and Silvester, 1996] Elman, H. C. and Silvester, D. (1996). Fast nonsymmetric iterations and preconditioning for Navier–Stokes equations. *SIAM Journal on Scientific Computing*, 17(1):33–46.

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

[Erhel et al., 1996] Erhel, J., Burrage, K., and Pohl, B. (1996). Restarted GMRES preconditioned by deflation. *Journal of computational and applied mathematics*, 69(2):303–318.

[Farhat et al., 1994] Farhat, C., Crivelli, L., and Roux, F. X. (1994). Extending substructure based iterative solvers to multiple load and repeated analyses. *Computer Methods in Applied Mechanics and Engineering*, 117(1-2):195–209.

[Farhat et al., 2001] Farhat, C., Lesoinne, M., LeTallec, P., Pierson, K., and Rixen, D. (2001). FETI-DP: a dual–primal unified FETI method—part i: A faster alternative to the two-level FETI method. *International Journal for Numerical Methods in Engineering*, 50(7):1523–1544.

[Farhat and Roux, 1991] Farhat, C. and Roux, F.-X. (1991). A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering*, 32(6):1205–1227.

[Ferronato et al., 2012] Ferronato, M., Janna, C., and Pini, G. (2012). Shifted FSAI preconditioners for the efficient parallel solution of non-linear groundwater flow models. *International Journal for Numerical Methods in Engineering*, 89(13):1707–1719.

[Fillinger et al., 2002] Fillinger, M. F., Raghavan, M. L., Marra, S. P., Cronenwett, J. L., and Kennedy, F. E. (2002). In vivo analysis of mechanical wall stress and abdominal aortic aneurysm rupture risk. *Journal of vascular surgery*, 36(3):589–597.

[Fleming et al., 2005] Fleming, C., Whitlock, E. P., Beil, T. L., and Lederle, F. A. (2005). Screening for abdominal aortic aneurysm: a best-evidence systematic review for the US preventive services task force. *Annals of Internal Medicine*, 142(3):203–211.

[Formaggia et al., 2010] Formaggia, L., Quarteroni, A., and Veneziani, A. (2010). *Cardiovascular Mathematics: Modeling and simulation of the circulatory system*, volume 1. Springer Science & Business Media.

[Forti, 2016] Forti, D. (2016). *Parallel algorithms for the solution of large-scale fluid-structure interaction problems in hemodynamics*. PhD thesis, EPFL.

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

[Gasser et al., 2010] Gasser, T. C., Auer, M., Labruto, F., Swedenborg, J., and Roy, J. (2010). Biomechanical rupture risk assessment of abdominal aortic aneurysms: model complexity versus predictability of finite element simulations. *European Journal of Vascular and Endovascular Surgery*, 40(2):176–185.

[Gee et al., 2006] Gee, M. W., Siefert, C. M., Hu, J. J., Tuminaro, R. S., and Sala, M. G. (2006). Ml 5.0 smoothed aggregation user's guide. Technical report, SAND2006-2649, Sandia National Laboratories.

[Gerner and Veroy, 2012] Gerner, A.-L. and Veroy, K. (2012). Certified reduced basis methods for parametrized saddle point problems. *SIAM Journal on Scientific Computing*, 34(5):A2812–A2836.

[Gervasio et al., 2006] Gervasio, P., Saleri, F., and Veneziani, A. (2006). Algebraic fractional-step schemes with spectral methods for the incompressible navier–stokes equations. *Journal of Computational Physics*, 214(1):347–365.

[Geuzaine and Remacle, 2009] Geuzaine, C. and Remacle, J.-F. (2009). Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331.

[Ghia et al., 1982] Ghia, U., Ghia, K. N., and Shin, C. (1982). High-re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48(3):387–411.

[Girault and Raviart, 2012] Girault, V. and Raviart, P.-A. (2012). *Finite element methods for Navier-Stokes equations: theory and algorithms*, volume 5. Springer Science & Business Media.

[Greenbaum, 1997] Greenbaum, A. (1997). *Iterative methods for solving linear systems*. SIAM.

[Grepl et al., 2007] Grepl, M. A., Maday, Y., Nguyen, N. C., and Patera, A. T. (2007). Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 41(3):575–605.

[Grepl and Patera, 2005] Grepl, M. A. and Patera, A. T. (2005). A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 39(1):157–181.

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

[Hackbusch, 2013] Hackbusch, W. (2013). *Multi-grid methods and applications*, volume 4. Springer Science & Business Media.

[Heroux et al., 2003] Heroux, M., Bartlett, R., Hoekstra, V. H. R., Hu, J., Kolda, T., Lehoucq, R., Long, K., Pawlowski, R., Phipps, E., Salinger, A., Thornquist, H., Tuminaro, R., Willenbring, J., and Williams, A. (2003). An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories.

[Heroux et al., 2005] Heroux, M. A., Bartlett, R. A., Howle, V. E., Hoekstra, R. J., Hu, J. J., Kolda, T. G., Lehoucq, R. B., Long, K. R., Pawlowski, R. P., Phipps, E. T., et al. (2005). An overview of the trilinos project. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):397–423.

[Hestenes and Stiefel, 1952] Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(1).

[Hesthaven et al., 2016] Hesthaven, J. S., Rozza, G., and Stamm, B. (2016). Certified reduced basis methods for parametrized partial differential equations. *SpringerBriefs in Mathematics*.

[Hesthaven et al., 2014] Hesthaven, J. S., Stamm, B., and Zhang, S. (2014). Efficient greedy algorithms for high-dimensional parameter spaces with applications to empirical interpolation and reduced basis methods. *ESAIM: Mathematical Modelling and Numerical Analysis*, 48(1):259–283.

[Kay et al., 2002] Kay, D., Loghin, D., and Wathen, A. (2002). A preconditioner for the steady-state Navier–Stokes equations. *SIAM Journal on Scientific Computing*, 24(1):237–256.

[Kerschen et al., 2005] Kerschen, G., Golinval, J., Vakakis, A. F., and Bergman, L. A. (2005). The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview. *Nonlinear Dynamics*, 41(1):147–169.

[Klawonn and Pavarino, 1998] Klawonn, A. and Pavarino, L. F. (1998). Overlapping schwarz methods for mixed linear elasticity and Stokes problems. *Computer Methods in Applied Mechanics and Engineering*, 165(1):233–245.

[Klawonn and Pavarino, 2000] Klawonn, A. and Pavarino, L. F. (2000). A comparison of overlapping schwarz methods and block preconditioners for saddle point problems. *Numerical Linear Algebra with Applications*, 7(1):1–25.

[Kressner and Tobler, 2011] Kressner, D. and Tobler, C. (2011). Low-rank tensor Krylov subspace methods for parametrized linear systems. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1288–1316.

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

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

[Lancellotti et al., 2017] Lancellotti, R. M., Vergara, C., Valdettaro, L., Bose, S., and Quarteroni, A. (2017). Large eddy simulations for blood dynamics in realistic stenotic carotids. *International journal for numerical methods in biomedical engineering*, 33(11).

[LeVeque, 2002] LeVeque, R. J. (2002). *Finite volume methods for hyperbolic problems*, volume 31. Cambridge University Press.

[Li, 2002] Li, J. (2002). *Dual-primal FETI Methods for Stationary Stokes and Navier-Stokes Equations.* New York University, Graduate School of Arts and Science.

[Li, 2005] Li, J. (2005). A dual-primal FETI method for incompressible Stokes equations. *Numerische Mathematik*, 102(2):257–275.

[Little and Saad, 2003] Little, L. and Saad, Y. (2003). Block preconditioners for saddle point problems. *Numerical Algorithms*, 33(1-4):367–379.

[Maday et al., 2009] Maday, Y., Nguyen, N. C., Patera, A. T., and Pau, S. H. (2009). A general, multipurpose interpolation procedure: the magic points. *Communications on Pure & Applied Analysis*, 8(1):383–404.

[Maier et al., 2010] Maier, A., Gee, M., Reeps, C., Pongratz, J., Eckstein, H.-H., and Wall, W. (2010). A comparison of diameter, wall stress, and rupture potential index for abdominal aortic aneurysm rupture risk prediction. *Annals of Biomedical Engineering*, 38(10):3124–3134.

[Malossi and Bonnemain, 2013] Malossi, A. C. I. and Bonnemain, J. (2013). Numerical comparison and calibration of geometrical multiscale models for the simulation of arterial flows. *Cardiovascular Engineering and Technology*, 4(4):440–463.

[Mandel and Dohrmann, 2007] Mandel, J. and Dohrmann, C. (2007). Multilevel BDDC in theory and practice. *HARRACHOV 2007*, page 63.

[Mandel and Dohrmann, 2003] Mandel, J. and Dohrmann, C. R. (2003). Convergence of a balancing domain decomposition by constraints and energy minimization. *Numerical linear algebra with applications*, 10(7):639–659.

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

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

[Manzoni et al., 2012a] Manzoni, A., Quarteroni, A., and Rozza, G. (2012a). Model reduction techniques for fast blood flow simulation in parametrized geometries. *International journal for numerical methods in biomedical engineering*, 28(6-7):604–625.

[Manzoni et al., 2012b] Manzoni, A., Quarteroni, A., and Rozza, G. (2012b). Shape optimization for viscous flows by reduced basis methods and free-form deformation. *International Journal for Numerical Methods in Fluids*, 70(5):646–670.

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

[Morgan, 2002] Morgan, R. B. (2002). GMRES with deflated restarting. *SIAM Journal on Scientific Computing*, 24(1):20–37.

[Morgan, 2005] Morgan, R. B. (2005). Restarted block-GMRES with deflation of eigenvalues. *Applied Numerical Mathematics*, 54(2):222–236.

[Nabben and Vuik, 2006] Nabben, R. and Vuik, C. (2006). A comparison of deflation and the balancing preconditioner. *SIAM Journal on Scientific Computing*, 27(5):1742–1759.

[Necas, 1967] Necas, J. (1967). Les méthodes directes en théorie des équations elliptiques.

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

[Negri et al., 2015a] Negri, F., Manzoni, A., and Amsallem, D. (2015a). Efficient model reduction of parametrized systems by matrix discrete empirical interpolation. *Journal of Computational Physics*, 303:431–454.

[Negri et al., 2015b] Negri, F., Manzoni, A., and Rozza, G. (2015b). Reduced basis approximation of parametrized optimal flow control problems for the Stokes equations. *Computers & Mathematics with Applications*, 69:319–336.

[Pagani, 2017] Pagani, S. (2017). *Reduced-order models for inverse problems and uncertainty quantification in cardiac electrophysiology.* PhD thesis, Politecnico di Milano.

[Pagani et al., 2017] Pagani, S., Manzoni, A., and Quarteroni, A. (2017). Numerical approximation of parametrized problems in cardiac electrophysiology by a local reduced basis method. Technical Report 25.2017, EPFL.

[Paige and Saunders, 1975] Paige, C. C. and Saunders, M. A. (1975). Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629.

[Parks et al., 2006] Parks, M. L., De Sturler, E., Mackey, G., Johnson, D. D., and Maiti, S. (2006). Recycling Krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5):1651–1674.

[Patankar, 1980] Patankar, S. (1980). *Numerical heat transfer and fluid flow.* CRC press.

[Paul-Dubois-Taine and Amsallem, 2015] Paul-Dubois-Taine, A. and Amsallem, D. (2015). An adaptive and efficient greedy procedure for the optimal training of parametric reduced-order models. *International Journal for Numerical Methods in Engineering*, 102(5):1262–1292.

[Peherstorfer et al., 2014] Peherstorfer, B., Butnaru, D., Willcox, K., and Bungartz, H.-J. (2014). Localized discrete empirical interpolation method. *SIAM Journal on Scientific Computing*, 36(1):A168–A192.

[Prud'homme et al., 2002] Prud'homme, C., Rovas, D. V., Veroy, K., and Patera, A. T. (2002). A mathematical and computational framework for reliable real-time solution of parametrized partial differential equations. *ESAIM Math. Modelling Numer. Anal.*, 36(5):747–771.

[Prud'homme et al., 2002] Prud'homme, C., Rovas, D. V., Veroy, K., Machiels, L., Maday, Y., Patera, A. T., and Turinici, G. (2002). Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods. *Journal of Fluids Engineering*, 124(1):70–80.

[Quarteroni, 2014] Quarteroni, A. (2014). *Numerical Models for Differential Problems*, volume 9 of *Modeling, Simulation and Applications (MS&A).* Springer-Verlag Italia, Milano, 2nd edition.

[Quarteroni et al., 2016a] Quarteroni, A., Manzoni, A., and Negri, F. (2016a). *Reduced Basis Methods for Partial Differential Equations: An Introduction.* Springer.

[Quarteroni et al., 2017] Quarteroni, A., Manzoni, A., and Vergara, C. (2017). The cardiovascular system: mathematical modelling, numerical algorithms and clinical applications. *Acta Numerica*, 26:365–590.

[Quarteroni and Rozza, 2007] Quarteroni, A. and Rozza, G. (2007). Numerical solution of parametrized Navier–Stokes equations by reduced basis methods. *Numerical Methods for Partial Differential Equations*, 23(4):923–948.

[Quarteroni et al., 2007] Quarteroni, A., Sacco, R., and Saleri, F. (2007). *Numerical mathematics*, volume 37. Springer Verlag.

[Quarteroni and Valli, 1999] Quarteroni, A. and Valli, A. (1999). *Domain decomposition methods for partial differential equations*, volume 10. Oxford University Press.

[Quarteroni and Valli, 2008] Quarteroni, A. and Valli, A. (2008). *Numerical approximation of partial differential equations.* Springer Science & Business Media.

[Quarteroni et al., 2016b] Quarteroni, A., Veneziani, A., and Vergara, C. (2016b). Geometric multiscale modeling of the cardiovascular system, between theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 302:193–252.

[Quarteroni et al., 2002] Quarteroni, A., Veneziani, A., and Zunino, P. (2002). Mathematical and numerical modeling of solute dynamics in blood flow and arterial walls. *SIAM Journal on Numerical Analysis*, 39(5):1488–1511.

[Raghavan and Vorp, 2000] Raghavan, M. and Vorp, D. A. (2000). Toward a biomechanical tool to evaluate rupture potential of abdominal aortic aneurysm: identification of a finite strain constitutive model and evaluation of its applicability. *Journal of Biomechanics*, 33(4):475–482.

[Ramage, 1999] Ramage, A. (1999). A multigrid preconditioner for stabilised discretisations of advection–diffusion problems. *Journal of computational and applied mathematics*, 110(1):187–203.

[Rehman et al., 2011] Rehman, M., Geenen, T., Vuik, C., Segal, G., and MacLachlan, S. (2011). On iterative methods for the incompressible Stokes problem. *International Journal for Numerical Methods in Fluids*, 65(10):1180–1200.

[Reid et al., 1990] Reid, D., Welch, G., and Pollock, J. (1990). Abdominal aortic aneurysm: a preventable cause of death? *Journal of the Royal College of Surgeons of Edinburgh*, 35(5):284–288.

[Risler and Rey, 2000] Risler, F. and Rey, C. (2000). Iterative accelerating algorithms with Krylov subspaces for the solution to large-scale nonlinear problems. *Numerical Algorithms*, 23(1):1.

[Roux, 1995] Roux, F. X. (1995). 10. parallel implementation of a domain decomposition method for non-linear elasticity problems. In *Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering*, pages 161–175. SIAM.

[Rowley, 2006] Rowley, C. W. (2006). Model reduction for fluids, using balanced proper orthogonal decomposition. In *Modeling And Computations In Dynamical Systems: In Commemoration of the 100th Anniversary of the Birth of John von Neumann*, pages 301–317. World Scientific.

[Rozza, 2009] Rozza, G. (2009). Reduced basis methods for Stokes equations in domains with non-affine parameter dependence. *Computing and Visualization in Science*, 12(1):23–35.

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

[Rozza and Veroy, 2007] Rozza, G. and Veroy, K. (2007). On the stability of the reduced basis method for Stokes equations in parametrized domains. *Computer Methods in Applied Mechanics and Engineering*, 196(7):1244–1260.

[Saad, 1981] Saad, Y. (1981). Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of Computation*, 37(155):105–126.

[Saad, 1987] Saad, Y. (1987). On the Lanczos method for solving symmetric linear systems with several right-hand sides. *Mathematics of computation*, 48(178):651–662.

[Saad, 1993] Saad, Y. (1993). A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469.

[Saad, 1997] Saad, Y. (1997). Analysis of augmented Krylov subspace methods. *SIAM Journal on Matrix Analysis and Applications*, 18(2):435–449.

[Saad, 2003] Saad, Y. (2003). *Iterative methods for sparse linear systems.* SIAM.

[Saad and Schultz, 1986] Saad, Y. and Schultz, M. H. (1986). GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869.

[Saad et al., 2000] Saad, Y., Yeung, M., Erhel, J., and Guyomarc'h, F. (2000). A deflated version of the conjugate gradient algorithm. *SIAM Journal on Scientific Computing*, 21(5):1909–1926.

[Sala and Heroux, 2005] Sala, M. and Heroux, M. (2005). Robust algebraic preconditioners using ifpack 3.0. *Sandia National Laboratories p*, 123.

[Salsa, 2016] Salsa, S. (2016). *Partial differential equations in action: from modelling to theory*, volume 99. Springer.

[Sargsyan et al., 2016] Sargsyan, S., Brunton, S. L., and Kutz, J. N. (2016). Online interpolation point refinement for reduced order models using a genetic algorithm. *arXiv preprint arXiv:1607.07702*.

[Schäfer et al., 1996] Schäfer, M., Turek, S., Durst, F., Krause, E., and Rannacher, R. (1996). Benchmark computations of laminar flow around a cylinder. In *Flow simulation with high-performance computers II*, pages 547–566. Springer.

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

[Silvester et al., 2001] Silvester, D., Elman, H. C., Kay, D., and Wathen, A. (2001). Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow. *Journal of Computational and Applied Mathematics*, 128(1):261–279.

[Silvester and Wathen, 1994] Silvester, D. and Wathen, A. (1994). Fast iterative solution of stabilised stokes systems part ii: using general block preconditioners. *SIAM Journal on Numerical Analysis*, 31(5):1352–1367.

[Simoncini and Szyld, 2007] Simoncini, V. and Szyld, D. B. (2007). Recent computational developments in krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, 14(1):1–59.

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

[Smith et al., 1996] Smith, B. F., Bjørstad, P. E., and Gropp, W. D. (1996). *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, New York, NY, USA.

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

[Stein et al., 2004] Stein, K., Tezduyar, T. E., and Benney, R. (2004). Automatic mesh update with the solid-extension mesh moving technique. *Computer Methods in Applied Mechanics and Engineering*, 193(21-22):2019–2032.

[Stewart and Sun, 1990] Stewart, G. and Sun, J.-g. (1990). Matrix perturbation theory. *Academic Press, New York*.

[Stüben, 2001] Stüben, K. (2001). An introduction to algebraic multigrid. *Multigrid*, pages 413–532.

[Taylor and Figueroa, 2009] Taylor, C. A. and Figueroa, C. (2009). Patient-specific modeling of cardiovascular mechanics. *Annual review of biomedical engineering*, 11:109–134.

[Temam, 1984] Temam, R. (1984). *Navier-Stokes Equations*, volume 2. North-Holland Amsterdam.

[Toselli and Widlund, 2005] Toselli, A. and Widlund, O. B. (2005). *Domain decomposition methods: algorithms and theory.* Springer series in computational mathematics. Springer, Berlin.

[Trottenberg et al., 2000] Trottenberg, U., Oosterlee, C. W., and Schuller, A. (2000). *Multigrid.* Academic press.

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

[ur Rehman et al., 2009] ur Rehman, M., Vuik, K., and Segal, G. (2009). Block preconditioners for the incompressible Stokes problem. In *Lirkov I., Margenov S., Waśniewski J. (eds) Large-Scale Scientific Computing. LSSC 2009. Lecture Notes in Computer Science*, volume 5910, pages 829–836. Springer, Berlin, Heidelberg.

[Van der Vorst, 2003] Van der Vorst, H. A. (2003). *Iterative Krylov methods for large linear systems*, volume 13. Cambridge University Press.

[Van der Vorst and Vuik, 1994] Van der Vorst, H. A. and Vuik, C. (1994). GMRESR: a family of nested GMRES methods. *Numerical Linear Algebra with Applications*, 1(4):369–386.

[Van Doormaal and Raithby, 1984] Van Doormaal, J. and Raithby, G. (1984). Enhancements of the SIMPLE method for predicting incompressible fluid flows. *Numerical Heat Transfer*, 7(2):147–163.

[Veroy and Patera, 2005] Veroy, K. and Patera, A. (2005). 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.

[Volkwein, 2013] Volkwein, S. (2013). Proper orthogonal decomposition: Theory and reduced-order modelling. *Lecture Notes, University of Konstanz*, 4(4).

[Vuik et al., 2009] Vuik, C., Segal, G., et al. (2009). SIMPLE-type preconditioners for the Oseen problem. *International Journal for Numerical Methods in Fluids*, 61(4):432–452.

[Vuik et al., 2000] Vuik, K., Saghir, A., and Boerstoel, G. (2000). The Krylov accelerated SIMPLE (R) method for flow problems in industrial furnaces. *International Journal for Numerical Methods in Fluids*, 33(7):1027–1040.

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

[Wathen, 2015] Wathen, A. J. (2015). Preconditioning. *Acta Numerica*, 24:329–376.

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

[Wesseling, 2009] Wesseling, P. (2009). *Principles of Computational Fluid Dynamics*, volume 29. Springer Science & Business Media.

[Wesseling and Oosterlee, 2001] Wesseling, P. and Oosterlee, C. W. (2001). Geometric multigrid with applications to computational fluid dynamics. *Journal of Computational and Applied Mathematics*, 128(1):311–334.

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

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

[Xu and Zikatanov, 2003] Xu, J. and Zikatanov, L. (2003). Some observations on Babuška and Brezzi theories. *Numerische Mathematik*, 94(1):195–202.

[Xu and Zikatanov, 2017] Xu, J. and Zikatanov, L. (2017). Algebraic multigrid methods. *Acta Numerica*, 26:591–721.

[Yavneh et al., 1998] Yavneh, I., Venner, C. H., and Brandt, A. (1998). Fast multigrid solution of the advection problem with closed characteristics. *SIAM Journal on Scientific Computing*, 19(1):111–125.

[Zahm and Nouy, 2016] Zahm, O. and Nouy, A. (2016). Interpolation of inverse operators for preconditioning parameter-dependent equations. *SIAM Journal on Scientific Computing*, 38(2):A1044–A1074.

[Zulehner, 2002] Zulehner, W. (2002). Analysis of iterative methods for saddle point problems: a unified approach. *Mathematics of computation*, 71(238):479–505.

Niccolò Dal Santo

Curriculum Vitæ et Studiorum

Date and place of birth: 18th August 1990, Verona, Italy
Nationality: Italian
Address: Rue de Genève 77, Lausanne, VD, Switzerland
Mobile: +41 78 9234508, +39 342 1570842
E-mail: niccolo.dalsanto@epfl.ch, ncl.dalsanto@gmail.com

Work experiences

01.2015 - present	**Doctoral assistant**, Chair of Modeling and Scientific Computing, École Polytechnyque Féderale de Lausanne, Lausanne, Switzerland.
01.2015 - present	**Software developer and main author of rb-LifeV**, the module of LifeV (www.lifev.org) for reduced basis methods. LifeV is an open source C++ library for the numerical solution of PDEs with the Finite Element method.
01.2017 - present	**Volunteer as project manager** at Innovation Forum Lausanne (IFL). About IFL: non-profit student association with the mission of fostering next generation of scientist-entrepreneurs (lausanne.inno-forum.org).

Education

01.2015 - present	**Ph.D.** in Mathematics, École Polytechnyque Féderale de Lausanne. Thesis: Multi space reduced basis preconditioners for parametrized partial differential equations. Directors: Prof. A. Quarteroni, Dr. A. Manzoni.
09.2012 - 12.2014	**Master degree** in Mathematical Engineering, Politecnico di Milano. Thesis: An adaptive discontinuous Galerkin spectral element method for systems of ordinary differential equations with applications to elastodynamics. Directors: Prof. A. Quarteroni, Dr. P. F. Antonietti. Final grade: 110/110 cum Laude.
02.2013 - 12.2014	**Alta Scuola Politecnica** (www.asp-poli.it), PoliMi and PoliTo. Project: Aced-IoT, Safe cities through Cloud and Internet of Things. **Master degree** in Mathematical Engineering, Politecnico di Torino, Final grade: 110/110 cum Laude.

10.2013 - 03.2014	**Exchange student**, Erasmus program. Computational Science and Engineering, Technische Universität München, Munich, Germany.
09.2009 - 09.2012	**Bachelor degree** in Mathematical Engineering, Politecnico di Milano Thesis: Model and Solutions for the Korteweg-de Vries equation. Director: Dr. G. Verzini. Final grade: 110/110 cum Laude.

Research interests

Finite Element method, Reduced Order Modeling, Reduced Basis method, Preconditioning techniques, High Performance Computing, Computational Fluid Dynamics.

Publications

- P.F. Antonietti, N. Dal Santo, I. Mazzieri, A. Quarteroni. A high-order discontinuous Galerkin approximation to ordinary differential equations with applications to elastodynamics. *IMA Journal of Numerical Analysis*, 2017.
- N. Dal Santo, S. Deparis, A. Manzoni, A. Quarteroni. Multi space reduced basis preconditioners for large-scale parametrized PDEs. SIAM Journal on Scientific Computing, 40(2):A954-A983, 2018.
- N. Dal Santo, S. Deparis, A. Manzoni. A numerical investigation of multi space reduced basis preconditioners for parametrized elliptic advection-diffusion equations. *Communications in Applied and Industrial Mathematics*, 8(1):282-297, 2017.
- N. Dal Santo, S. Deparis, A. Manzoni, A. Quarteroni. An algebraic least squares reduced basis method for parametrized Stokes equations. Mathicse Report 21.2017. Submitted.
- N. Dal Santo, S. Deparis, A. Manzoni, A. Quarteroni. Multi space reduced basis preconditioners for parametrized Stokes equations. Mathicse Report 03.2018. Submitted.
- N. Dal Santo, A. Manzoni. Hyper-reduced order models for parametrized unsteady Navier-Stokes equations on domains with variable shape. Submitted.

Conferences and workshops

- Model Reduction of Parametrized Systems IV, École Centrale Nantes, Nantes, France, 10-13 April 2018 (Poster).
- Preconditioning 2017 (31.07 - 02.08.2017), University of British Columbia, BC, Canada (Talk in contributed session).
- SIAM CSE 2017 (27.02 - 03.03.2017), Hilton Atlanta, Atlanta, GA (Talk in two MiniSymposia).
- Recent developments in numerical methods for model reduction (07 - 10.11.2016), IHP quarter on Numerical Methods for PDEs, IHP, Paris, France (Poster).
- Reduced Basis summer School 2016 (03 - 07.10.2016), Max Planck Insitute for dynamics of complex and technical systems Magdeburg, Kloster Hedersleben, Hedersleben, Germany. (Talk)
- SIMAI 2016 (13 - 16.09.2016), Politecnico di Milano, Milano, Italy (Talk in a MiniSymposium).
- Corso Estivo SMI: Reduced Basis, Optimal Control & Application to Cardiovascular Modeling (27 - 31.07.2015), Cortona, Italy.
- Spectral Elements in Elastodynamics: applications to seismic wave propagation problems (09.04.2015), MOX, Politecnico di Milano (Invited talk).

Teaching activities

- Semester project co-supervisor: Reduced basis method for flows in moving domains, Clément Lefebvre, Master Program in Computational Science and Engineering, Spring Sem. 16/17, EPFL.
- Semester project co-supervisor: Reduced basis solver for time dependent problems, Vincent Pollet, Master Program in Computational Science and Engineering, Spring Sem. 16/17, EPFL.
- Semester project co-supervisor: Élastodynamique linéaire pour problème géophysique et dispersion numérique, Julie Favre, Bachelor Program in Mathematics, Spring Sem. 14/15, EPFL.
- Teaching Assistant at EPFL, classes: Numerical Analysis, Bachelor Program in Biomedical engineering, Spring Sem. 16/17; Numerical Analysis & Computational Mathematics, Master Program in Computational Science and Engineering; Introduction to Finite Elements method, Bachelor Program in Mathematics and Master Program in Computational Science and Engineering, Winter Sem. 15/16; Numerical Analysis, Bachelor Program in Mathematics, Spring Sem. 14/15.