Applied Mathematics and Computation 442 (2023) 127722

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Approximation of optimal control problems for the Navier-Stokes equation via multilinear HJB-POD

Maurizio Falcone[a], Gerhard Kirsten[b], Luca Saluzzi[c,*]

[a] Università di Roma La Sapienza, P. Aldo Moro, 5, Roma, 00185, Italy
[b] Dipartimento di Matematica, Università di Bologna, Piazza P.ta S. Donato, Bologna, 51000, Italy
[c] Department of Mathematics, Imperial College London, South Kensington Campus, SW7 2AZ, London, United Kingdom

ARTICLE INFO

Keywords:
Dynamic programming
Optimal control
Tree structure
Model order reduction

ABSTRACT

We consider the approximation of some optimal control problems for the Navier-Stokes equation via a Dynamic Programming approach. These control problems arise in many industrial applications and are very challenging from the numerical point of view since the semi-discretization of the dynamics corresponds to an evolutive system of ordinary differential equations in very high-dimension. The typical approach is based on the Pontryagin maximum principle and leads to a two point boundary value problem. Here we present a different approach based on the value function and the solution of a Bellman equation, a challenging problem in high-dimension. We mitigate the curse of dimensionality via a recent multilinear approximation of the dynamics coupled with a dynamic programming scheme on a tree structure. We discuss several aspects related to the implementation of this new approach and we present some numerical examples to illustrate the results on classical control problems studied in the literature.

1. Introduction

The control of fluids is an important issue in many industrial problems, e.g. in aerospace and naval industries. The approximation of the fluid around complex geometries usually requires a very careful construction of the grid and is based on finite elements or finite differences/volumes schemes (see e.g. Pironneau [1], Strikwerda [2] and the references therein). This is known to be a huge computational problem so model reduction techniques are often applied to compute the solution [3,4] and study the physical properties of the flow varying various parameters (e.g. bifurcation phenomena as in Quarteroni and Rozza [5], Stabile and Rozza [6], Pichi et al. [7]). A typical example is given by the Navier-Stokes (NS) equation for incompressible fluids that we use here as our model problem.

These problems have been studied by many authors from the theoretical point of view analysing controllability properties of the system via Carlemann estimates. The interested reader can find in [8,9] a comprehensive presentation of these results.

For our optimal control problems the numerical approximation of the Navier-Stokes equation is the starting point. In fact, we want to solve a huge optimization problem where the controlled solution should minimize a cost functional, e.g. to

* Corresponding author.
E-mail addresses: falcone@mat.uniroma1.it (M. Falcone), gpkirsten@gmail.com (G. Kirsten), l.saluzzi@ic.ac.uk (L. Saluzzi).

https://doi.org/10.1016/j.amc.2022.127722

minimize the drag or to stay close to some reference solution. Many authors have contributed to these problems following the pioneering work of J.L. Lions in Lions [10] and the numerical approach is mainly based on the Pontryagin Maximum Principle (PMP) that gives a necessary condition characterizing the optimal couple trajectory/control of the problem. The numerical solution of the PMP leads to a two point boundary value problem that is feasible in high-dimension but typically produces open-loop controls (see Bardi and Capuzzo-Dolcetta [11]). In practice this approach can be difficult to implement since it requires a starting guess for the optimal trajectory and for the optimal control (the co-state) that is not available, in particular the co-state is usually hard to initialize. Furthermore, open-loop controllers are typically unstable under perturbations and hence not suitable for real-time applications. The interested reader can find in the book Tröltzsch [12] and in the lecture notes Casas [13] a general presentation of the results. We just recall that for the PMP approach two different strategies have been proposed: "optimize then discretize" and "discretize then optimize". The first is based on the discretization of the system of optimality conditions obtained for the continuous problem whereas the second starts with the discretization of the optimal control problem and then solves the optimality condition for the finite dimensional problem (see Hinze et al. [14] for a general presentation of these numerical strategies).

As we said, here we follow the Dynamic Programming (DP) approach based on the characterization of the value function as the unique viscosity solution of a Hamilton-Jacobi-Bellman (HJB) equation. This approach is more interesting since it produces a characterization of optimal controls in feedback form via the knowledge of the value function, however its application to the control of PDEs has been very limited due to the "curse of dimensionality". In fact, adopting the strategy "discretize then optimize", we need to solve the Hamilton-Jacobi-Bellman equation in high-dimension (i.e. the dimension of the discrete state space after the semi-discretization of the continuous problem). It is known that this nonlinear partial differential equation gives the characterization of the value function as its unique viscosity solution in many optimal control problems (see e.g. Bardi and Capuzzo-Dolcetta [11]). It is interesting to note that this problem is difficult also in low dimension since the value function is only Lipschitz continuous also when the dynamics and the cost are assumed to be very regular, but in low dimension several methods have been proposed ranging from finite difference methods [15], semi-lagrangian schemes [16] to finite volumes. Here we propose a new method for the numerical solution of control problems of Navier-Stokes equations based on the DP approach. The novelty in the technique used is to mitigate the curse of dimensionality via the coupling of two recent methods: a multilinear approximation of the NS equation developed in Kirsten and Simoncini [17] and the dynamic programming method for the finite horizon problem on a tree originally developed for nonlinear ordinary differential equations in Alla et al. [18] obtaining also a-priori error estimates in Saluzzi et al. [19]. The first method allows to produce a numerical solution in a very compact form via tensor notations whereas the Tree Structure Algorithm (TSA) exploits the compact representation of the systems and can be coupled with a model reduction approach based on Proper Orthogonal Decomposition (POD). In fact the tree structure method is rather flexible, we refer to Alla and Saluzzi [20] and Alla et al. [21] for recent developments including high-order approximation, the coupling with model reduction techniques, problems with state constraints. To set this paper into perspective, let us also mention that the coupling of HJB equation with POD for the approximation of optimal control problems with PDE constraints has been proposed by Kunisch and co-authors in a series of papers [22–24] (see also Hinze and Volkwein [25]). They have analyzed optimal control problems mainly for linear parabolic equation and the Burgers equation. Let us also mention that more recently in Breiten and Kunisch [26], Breiten et al. [27] the authors have attacked a control problems for the Navier-Stokes system via the Lyapunov function technique. The numerical method proposed here is different since it is based on the above mentioned building blocks allowing to mitigate the "curse of dimensionality". Other techniques have been introduced in the last decades in this direction, among them we mention in particular sparse grids [28], tensor decomposition techniques. Dolgov et al. [29], Oster et al. [30], Dolgov et al. [31] and some very recent approaches based on neural networks [32,33].

Our main goal here is to describe the coupling between our building blocks, explain how they can be implemented and show our first numerical results on classical control problems for NS equations. We believe that the simulations presented in the last section illustrate that DP is now feasible from a computational point of view also for fluids and we hope that this can open the way to its application in real industrial applications.

The paper is organized as follows. In the second section, we will introduce some classical control problems for NS equations and recall the results available in the literature for continuous problems. Section 3 will be devoted to the presentation of the multilinear approximation and its implementation. In Section 4 we present the TSA and the coupling with the multilinear approximation. In the last section we present some numerical experiments on a number of challenging test problems studied in the literature illustrating the main features of our approach.

Notation and Tensor basics

In the present work all matrices are represented by capital bold-face letters, whereas scalars are given by standard lowercase letters. In the context of model reduction, all matrices with a $\widehat{\ }$ on top, represent low-dimensional quantities.

The Kronecker product of two matrices $\boldsymbol{M} \in \mathbb{R}^{m_1 \times m_2}$ and $\boldsymbol{N} \in \mathbb{R}^{n_1 \times n_2}$ is defined as

$$\boldsymbol{M} \otimes \boldsymbol{N} = \begin{pmatrix} M_{1,1}\boldsymbol{N} & \cdots & M_{1,m_2}\boldsymbol{N} \\ \vdots & \ddots & \vdots \\ M_{m_1,1}\boldsymbol{N} & \cdots & M_{m_2,m_2}\boldsymbol{N} \end{pmatrix} \in \mathbb{R}^{m_1 n_1 \times m_2 n_2},$$

and the vec(·) operator maps the entries of a matrix, into a long vector, by stacking the columns of the matrix one after the other. Moreover, we will often make use of the property

$$(\boldsymbol{M} \otimes \boldsymbol{N})\mathrm{vec}(\boldsymbol{X}) = \mathrm{vec}(\boldsymbol{N}\boldsymbol{X}\boldsymbol{M}^{\top}).$$

Furthermore, the matrix operation $\boldsymbol{M} \bullet \boldsymbol{N}$, where $\boldsymbol{M}$ and $\boldsymbol{N}$ have the same size, is known as the Hadamard product, which is an element-element multiplication of the two matrices.

2. The optimal control problem for the Navier-Stokes equation and its discretization

We introduce some optimal control problems for the Navier-Stokes equation giving also some hints on its numerical solution via finite differences (FD). The approach we present is not limited to this FD approximation and can be extended to other numerical methods as Finite Elements or Finite Volumes since the multilinear discretization described in the next section applies to the semi-discrete system.

2.1. The Navier-Stokes dynamical system and its discretization

Let us assume that the physical domain is a regular bounded connected open set Ω in $\mathbb{R}^2$ whose boundary will be denoted by Γ. We denote by $\eta(x)$ the exterior normal vector to a point $x \in \Gamma$. The time variable t will be taken on the interval $(0, T)$ with $T > 0$. The standard uncontrolled dynamics will be given by

$$\begin{cases} z_t - \varepsilon \Delta z + (z \cdot \nabla) z + \nabla p = f & \text{in } \Omega \times (0,T), \\ \text{div } z = 0 & \text{in } \Omega \times (0,T), \\ z = 0 & \text{on } \Gamma \times (0,T), \\ z(0,x,y) = z_0(x,y) & \text{in } \Omega. \end{cases} \tag{1}$$

In the sequel we will consider as a model problem the following form for the Navier-Stokes equation

$$\begin{cases} u_t - \frac{1}{r}(u_{xx} + u_{yy}) + u u_x + v u_y + p_x = f & \text{in } \Omega \times (0,T), \\ v_t - \frac{1}{r}(v_{xx} + v_{yy}) + u v_x + v v_y + p_y = f & \text{in } \Omega \times (0,T), \\ u_x + v_y = 0 & \text{in } \Omega \times (0,T), \\ u(t,x,y) = v(t,x,y) = 0 & \text{on } \Gamma \times (0,T), \\ u(0,x,y) = u_0(x,y), v(0,x,y) = v_0(x,y) & \text{in } \Omega \times (0,T), \end{cases} \tag{2}$$

where (x, y) are the space coordinates and we set $\varepsilon = 1/r$, $z = (u, v)$, where $u, v : \Omega \times (0, T) \to \mathbb{R}$, with $\Omega \equiv [0, b_x] \times [0, b_y] \subset \mathbb{R}^2$ are the velocities to be determined and the normalized pressure $p : \overline{\Omega} \to \mathbb{R}$ is a Lagrange multiplier introduced to satisfy the incompressibility condition. The boundary condition can be chosen according to the problem we want to solve and, as we will see in the next section, can also include some control terms (this will be the case for the boundary control problem). As an example, we can consider the Dirichlet homogeneous boundary condition $z = 0$ or the no-slip boundary conditions on each wall, that is

$$\begin{aligned} &u(t,x,b_y) = u_N(x), \quad u(t,x,0) = u_S(x), \quad u(t,0,y) = 0, \quad u(t,b_x,y) = 0, \\ &v(t,b_x,y) = v_E(y), \quad v(t,0,y) = v_W(y), \quad v(t,x,0) = 0, \quad v(t,x,b_y) = 0. \end{aligned}$$

For the general results on the Navier-Stokes equation we refer to the book Temam [34] whereas for the analysis of some stabilization problems in this framework we refer to Barbu and Triggiani [35].

2.2. Some control problems for the Navier-Stokes equation

The control of non linear dynamical systems over a finite horizon is usually treated via direct methods based on Pontryagin maximum principle that results in the numerical solution of a two point boundary value problem. This results in an open-loop control and often requires a long work to choose the initial conditions for the state and the co-state since the convergence is local. Moreover, the PMP just gives necessary conditions of optimality and the setting of sufficient conditions is much more technical in this framework. As we said we are going to present a different approach for some classical control problems that we briefly review here.

1. Distributed control in Ω

A first way to introduce a control term is to add a term to the equation, so we write

$$z_t - \varepsilon \Delta z + (z \cdot \nabla) z + \nabla p + \sum_{i=1}^{m} \alpha_i \, \psi_i(x) = f \qquad \text{in } \Omega \times (0,T). \tag{3}$$

In this problem the control is a measurable vector $\alpha : [0, T) \to A$ where A is a compact subset of $\mathbb{R}^m$ and the functions ψ_i are some predefined shape functions. It is possible to fix the shape function as a characteristic function $\mathbf{1}_\omega$ with $\omega \subset \Omega$, obtaining a control term which will apply only on the subdomain ω.

2. A boundary control problem on $\Gamma_0 \subset \Gamma$

A second way is to consider a controlled dynamics where the control appears in the boundary condition, so we take the dynamics (1) but we modify the boundary Dirichlet condition as

$$z = \alpha \quad \text{on } \Gamma_0 \times (0,T), \tag{4}$$

$$z = g \quad \text{on } (\Gamma \setminus \Gamma_0) \times (0,T), \tag{5}$$

and the control must satisfy the compatibility condition

$$\int_{\Gamma_0} \alpha \cdot \eta \, d\Gamma_0 = 0, \tag{6}$$

and α will be our control function defined on a small subset Γ_0 of the boundary Γ.

We will always denote by $y(\cdot,t;\alpha)$ the unique solution of the dynamical system corresponding to the choice of the control α. Then we are going to introduce the cost functional to complete the definition of our control problem. A general form of the cost functional we want to minimize is

$$J(\alpha) = \int_0^T \|y(\cdot,t;\alpha) - \bar{y}(\cdot,t)\|^2_{L^2(\Omega)} + \gamma\|\alpha\|^2)e^{-\lambda t}dt + \|y(\cdot,T;\alpha) - \bar{y}(\cdot,T)\|^2_{L^2(\Omega)}, \tag{7}$$

where the two parameters γ and λ are positive and the running cost includes the distance of the controlled solution from a reference trajectory $\bar{y}$ plus a penalization on the control α.

In the second and third experiment presented in Section 5 $\bar{y}$ will represent the solution of the stationary problem, so the meaning is to stabilize the problem and reach as soon possible the stationary solution.

For the boundary control problem we will use the same cost functional but $\bar{y}$ will represent the solution of the problem where we have set a specific control $\overline{\alpha}$.

2.3. The finite horizon discrete optimal control problem

Let us recall for the reader's convenience the classical Dynamic Programming approach for the *finite horizon optimal control problem* for ordinary differential equations, that we use as a model problem. The system is driven by

$$\begin{cases} \dot{y}(s) = f(y(s), \alpha(s), s), \quad s \in (t,T], \\ y(t) = x \in \mathbb{R}^d, \end{cases} \tag{8}$$

and we denote by $y : [t,T] \to \mathbb{R}^d$ the solution, by $\alpha : [t,T] \to \mathbb{R}^m$ the control, by $f : \mathbb{R}^d \times \mathbb{R}^m \times [t,T] \to \mathbb{R}^d$ the dynamics and by

$$\mathcal{A} = \{\alpha : [t,T] \to A, \text{measurable}\}$$

the set of admissible controls where $A \subset \mathbb{R}^m$ is a compact set. The cost functional for the finite horizon control problem is given by

$$J_{x,t}(\alpha) := \int_t^T L(y(s,\alpha), \alpha(s), s)e^{-\lambda(s-t)}\, ds + g(y(T))e^{-\lambda(T-t)}, \tag{9}$$

where $L : \mathbb{R}^d \times \mathbb{R}^m \times [t,T] \to \mathbb{R}$ is the running cost and $\lambda \geq 0$ is the discount factor.

The typical assumptions on the functions f, L, g are:

$$|f(x,a,s)| \leq M_f, \qquad |L(x,a,s)| \leq M_L, \quad |g(x)| \leq M_g, \quad \forall\, x \in \mathbb{R}^d, a \in A \subset \mathbb{R}^m, s \in [t,T], \tag{10}$$

the functions f, L and g are Lipschitz-continuous with respect to the first variable

$$|w(x,a,s) - w(y,a,s)| \leq L_w|x-y|, \text{ for } w = f, L \text{ and } \forall\, x, y \in \mathbb{R}^d, a \in A \subset \mathbb{R}^m, s \in [t,T], \tag{11}$$

and finally the cost g is also Lipschitz-continuous:

$$|g(x) - g(y)| \leq L_g|x-y|, \quad \forall x, y \in \mathbb{R}^d. \tag{12}$$

Furthermore, assuming the existence of a global solution in time, all the trajectories will live in a compact set since we have a finite time evolution. We refer to Giga et al. [36] for a complete statement on the global existence of the 2D incompressible NS equation. In this case the boundedness assumptions (10) can be avoided and the hyphotesis (11)-(12) can be considered locally. The goal is to find a state-feedback control law $\alpha(t) = \Phi(y(t),t)$, in terms of the state equation $y(t)$, where Φ is the feedback map. To derive optimality conditions we use the well-known Dynamic Programming Principle (DPP) due to Bellman. We first define the value function for an initial condition $(x,t) \in \mathbb{R}^d \times [t,T]$:

$$v(x,t) := \inf_{\alpha \in \mathcal{A}} J_{x,t}(\alpha). \tag{13}$$

A classical result (see Bardi and Capuzzo-Dolcetta [11] shows that under our assumptions the value function for the finite horizon problem is the unique viscosity solution of the following Hamilton-Jacobi-Bellman equation

$$\begin{cases} -\dfrac{\partial v}{\partial s}(x,s) + \lambda v(x,s) + \max\limits_{a \in A}\{-L(x,a,s) - \nabla v(x,s) \cdot f(x,a,s)\} = 0, & \text{for } x \in \mathbb{R}^d\, s \in [t,T), \\ v(x,T) = g(x) & \text{for } x \in \mathbb{R}^d. \end{cases} \tag{14}$$

Once the value function is known, by e.g. (14), then it is possible to compute the optimal feedback control as:

$$\alpha^*(t) := \arg\max_{a\in A} \{-L(x,a,t) - \nabla v(x,t)\cdot f(x,a,t)\}, \tag{15}$$

and this is one of the most important features of the DP approach to control problems. From the numerical approximation of the feedback we can apply (15) replacing the continue value function v by our numerical approximation and its gradient by discrete gradients (see [16] for more details on this point).

3. Multilinear approximation of the Navier-Stokes equation

In this section we present the matrix-oriented discretization of the NS equation used to set-up the finite dimensional optimal control problem. The technique presented here allows us to significantly speed up the numerical time integration; see the first test presented in Section 5. More details on the procedure presented here can be found in Kirsten [37].

3.1. Matrix-oriented discretization and 2S-POD-DEIM for general PDEs

Consider a semilinear evolutive PDE of the form

$$u_t = \mathcal{L}(u) + f(u,t), \quad u = u(\boldsymbol{x},t) \quad \text{with } \boldsymbol{x}\in\Omega\subset\mathbb{R}^2, \ \ t\in\mathcal{T}, \tag{16}$$

with suitable boundary conditions. We assume that the differential operator $\mathcal{L}$ is linear in u with separable coefficients, typically a second order operator in the space variables, while $f: S\times\mathcal{T}\to\mathbb{R}$ is a nonlinear function, where S is an appropriate space with $u\in S$, and $\mathcal{T}$ is the timespan. Under these assumptions, if $\mathcal{L}$ is discretized by means of a tensor basis, such as finite differences on rectangular domains, certain finite element methods and certain spectral methods, then the physical domain can be mapped to a reference hypercubic domain. More precisely, if we consider finite differences on a rectangular domain, then

$$\mathcal{L} = \boldsymbol{I}_{n_y}\otimes\boldsymbol{A}_1 + \boldsymbol{A}_2^\top\otimes I_{n_x},$$

where $\boldsymbol{A}_1\in\mathbb{R}^{n_x\times n_x}$ and $\boldsymbol{A}_2\in\mathbb{R}^{n_y\times n_y}$ are matrices containing the coefficients for the derivatives and n_x and n_y are the number of discretization nodes in the $x-$ and $y-$ directions respectively. As a result if we define $\boldsymbol{U}(t)\in\mathbb{R}^{n_x\times n_y}$ as a matrix containing an approximation to the solution $u(t)$ at each discretization node, then the discrete version of (16) can be expressed in matrix form as

$$\dot{\boldsymbol{U}}(t) = \boldsymbol{A}_1\boldsymbol{U}(t) + \boldsymbol{U}(t)A_2 + \boldsymbol{F}(\boldsymbol{U},t). \tag{17}$$

In addition to a better structural interpretation of the discrete quantities, this formulation can also lead to reduced memory requirements and computational costs. A summary of this and related matrix-oriented procedures can be found in [38,39]. Furthermore, standard numerical integration schemes, such as semi-implicit schemes and exponential integrators, can be performed directly in matrix form to approximate the solution of (17) throughout the timespan [37,40].

As it is well-known in the vector formulation of discretized PDEs, the discrete matrices are often very large and sparse and require a large computational effort to solve the resulting linear systems at each time step. To this end model order reduction techniques such as POD [3,25] and DEIM [41] have been successfully applied to reduce the complexity of solving several linear systems throughout the timespan. In [17,42] the POD and DEIM methods have been extended so that they can be applied directly to the matrix differential Eq. (17), without requiring any mapping from matrices to vectors.

In short, consider a set of n_s time-dependent snapshot solutions and nonlinear snapshots of (17), given by $\{\boldsymbol{U}^i\}_{i=1}^{n_s}$ and $\{\boldsymbol{F}(\boldsymbol{U}^i)\}_{i=1}^{n_s}$ respectively. The 2S-POD algorithm from [17] is applied to the set of snapshot solutions to form two tall matrices $\boldsymbol{U}_\ell\in\mathbb{R}^{n_x\times k_\ell}$ and $\boldsymbol{U}_r\in\mathbb{R}^{n_y\times k_r}$ ($k_\ell,\ k_r\ll n$) with orthornormal columns. The parameters k_ℓ and k_r refer to the number of selected dominant singular values such that the projection error is bounded by a prescribed tolerance *tol*. The span of these columns respectively approximate the row and column space of the snapshot solutions. To this end, we approximate the solution $\boldsymbol{U}(t)$ of (17) by $\boldsymbol{U}(t)\approx\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top =: \widetilde{\boldsymbol{U}}(t)$, for $t\in\mathcal{T}$, where $\widehat{\boldsymbol{U}}(t)\in\mathbb{R}^{k_\ell\times k_r}$ satisfies the following low-dimensional matrix ODE:

$$\begin{cases} \dot{\widehat{\boldsymbol{U}}}(t) = \widehat{\boldsymbol{A}}_1\widehat{\boldsymbol{U}}(t) + \widehat{\boldsymbol{U}}(t)\widehat{\boldsymbol{A}}_2 + \widehat{F}(\widetilde{\boldsymbol{U}}(t),t), \quad t\in\mathcal{T}, \\ \widehat{\boldsymbol{U}}(0) = \widehat{\boldsymbol{U}}_0 = \boldsymbol{U}_\ell^\top\boldsymbol{U}_0\boldsymbol{U}_r, \end{cases} \tag{18}$$

where $\widehat{\boldsymbol{A}}_1 = \boldsymbol{U}_\ell^\top\boldsymbol{A}_1\boldsymbol{U}_r$ and $\widehat{\boldsymbol{A}}_2 = \boldsymbol{U}_\ell^\top\boldsymbol{A}_2\boldsymbol{U}_r$ and $\widehat{F}(\widetilde{\boldsymbol{U}}(t),t) = \boldsymbol{U}_\ell^\top\boldsymbol{F}(\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top,t)\boldsymbol{U}_r$. Despite the fact that $\widehat{F}(\widetilde{\boldsymbol{U}}(t),t)$ is considered a low-dimensional quantity, the calculation of it still results in a computational bottleneck, since the nonlinear function first needs to be evaluated at all the entries of $\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(s)\boldsymbol{U}_r^\top\in\mathbb{R}^{n_x\times n_y}$ before it is projected onto the low-dimensional space. To overcome this bottleneck we apply the 2S-DEIM method from Kirsten and Simoncini [17].

We consider the set of nonlinear snapshots $\{\boldsymbol{F}(\boldsymbol{U}^i)\}_{i=1}^{n_s}$ and use the 2S-POD algorithm to form two tall matrices $\boldsymbol{\Phi}_\ell\in\mathbb{R}^{n_x\times p_1}$ and $\boldsymbol{\Phi}_r\in\mathbb{R}^{n_y\times p_2}$ ($p_i\ll n$) with orthornormal columns. We aim to approximate the nonlinear term by far smaller matrices, that is $\boldsymbol{F}(\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(s)\boldsymbol{U}_r^\top)\approx\boldsymbol{\Phi}_\ell\widehat{\boldsymbol{F}}(t)\boldsymbol{\Phi}_r^\top$, where $\widehat{\boldsymbol{F}}(t)\in\mathbb{R}^{p_1\times p_2}$ is a matrix of time-dependent coefficients. This leads to a 2S-DEIM approximation of the form:

$$\widehat{F}(\widetilde{\boldsymbol{U}}(t),t)\approx\boldsymbol{\Phi}_\ell(\boldsymbol{D}_\ell^\top\boldsymbol{\Phi}_\ell)^{-1}\boldsymbol{D}_\ell^\top\boldsymbol{F}(\widetilde{\boldsymbol{U}}(t),t)\boldsymbol{D}_r(\boldsymbol{\Phi}_r^\top\boldsymbol{D}_r)^{-1}\boldsymbol{\Phi}_r^\top, \tag{19}$$

where $\boldsymbol{D}_\ell \in \mathbb{R}^{n_x \times p_1}$ and $\boldsymbol{D}_r \in \mathbb{R}^{n_y \times p_2}$ respectively contain a subset of p_1 and p_2 columns of the $n_x \times n_y$ identity matrix. The indices at which these columns are selected is determined by respectively applying the Q-DEIM algorithm from [43] to the matrices $\boldsymbol{\Phi}_\ell$ and $\boldsymbol{\Phi}_r$. In the elegant case where the nonlinear function is evaluated elementwise at the indices of $\widetilde{\boldsymbol{U}}(t)$, the respective interpolation indices can be selected by taking $\boldsymbol{D}_\ell$ and $\boldsymbol{D}_r$ inside the nonlinear function such that

$$\widehat{F}(\widetilde{\boldsymbol{U}}(t), t) \approx \boldsymbol{U}_\ell^\top \boldsymbol{\Phi}_\ell (\boldsymbol{D}_\ell^\top \boldsymbol{\Phi}_\ell)^{-1} \boldsymbol{F}(\boldsymbol{D}_\ell^\top \widetilde{\boldsymbol{U}}(t) \boldsymbol{D}_r, t)(\boldsymbol{\Phi}_r^\top \boldsymbol{D}_r)^{-1} \boldsymbol{\Phi}_r^\top \boldsymbol{U}_r \tag{20}$$

can be completely evaluated in low-dimension. In the case that the nonlinear term is not evaluated elementwise, more complex techniques may be required [41]. This situation is also encountered with the NS equation and will be discussed in the following section.

In what follows we aim to extend these matrix-oriented discretization, integration and model reduction strategies to the setting of the NS equation.

3.2. The NS equation in full dimension

The method considered for the time and space discretization of (2) is finite differences on a staggered grid. A discussion of the scheme and a Matlab implementation in the vector setting can be respectively found in [44] and [45]. Here we aim to take explicit advantage of the rectangular domain, to directly treat the equation in matrix form, both for the reduction and integration phases of the method.

For the space discretization, we consider n_x gridpoints in the $x-$direction and n_y gridpoints in the $y-$direction. For the staggered grid, the velocities u are placed on the vertical cell interfaces, v on the horizontal cell interfaces and the pressure p in the centre of the cells. That is, the discretized quantities are given by the matrices $\boldsymbol{U}(t) \in \mathbb{R}^{(n_x-1)\times n_y}$, $\boldsymbol{V}(t) \in \mathbb{R}^{n_x \times (n_y-1)}$ and $\boldsymbol{P}(t) \in \mathbb{R}^{n_x \times n_y}$. Given $* \in \{U, V\}$, we consider the matrices $\boldsymbol{A}_{1,*}$ and $\boldsymbol{A}_{2,*}$ with corresponding dimensions to respectively contain the coefficients for the second derivative in the $x-$ and $y-$ directions and the matrices $\boldsymbol{B}_{1,*}$ and $\boldsymbol{B}_{2,*}$ that of the first derivatives. The discrete version of (2) is then given by

$$\begin{cases} \dot{\boldsymbol{U}} - \boldsymbol{A}_{1,U}\boldsymbol{U} + \boldsymbol{U}\boldsymbol{A}_{2,U}^\top + \boldsymbol{B}_{1,U}^\top \boldsymbol{P} + \boldsymbol{F}_U(\boldsymbol{U}, \boldsymbol{V}, t) &= 0, \\ \dot{\boldsymbol{V}} - \boldsymbol{A}_{1,V}\boldsymbol{V} + \boldsymbol{V}\boldsymbol{A}_{2,V}^\top + \boldsymbol{P}\boldsymbol{B}_{2,V} + \boldsymbol{F}_V(\boldsymbol{U}, \boldsymbol{V}, t) &= 0, \\ \boldsymbol{B}_{1,U}\boldsymbol{U} + \boldsymbol{V}\boldsymbol{B}_{2,V}^\top &= 0, \end{cases} \tag{21}$$

where $\boldsymbol{F}_U(\boldsymbol{U}, \boldsymbol{V}, t) = \boldsymbol{B}_{1,U}\boldsymbol{U} \bullet \boldsymbol{U} + \boldsymbol{U}\boldsymbol{B}_{2,U}^\top \bullet \boldsymbol{V}$ and $\boldsymbol{F}_V(\boldsymbol{U}, \boldsymbol{V}, t) = \boldsymbol{B}_{1,V}\boldsymbol{V} \bullet \boldsymbol{U} + \boldsymbol{V}\boldsymbol{B}_{2,V}^\top \bullet \boldsymbol{V}$. For the time discretization we consider a simple semi-implicit Euler scheme, so that the viscosity terms are treated implicitly, the nonlinear terms explicitly, and the pressure term is treated implicitly via a Chorin Projection scheme (see [46]). That is, at each time iteration the approximations $\boldsymbol{U}^{(j+1)} \approx \boldsymbol{U}(t_{j+1})$ and $\boldsymbol{V}^{(j+1)} \approx \boldsymbol{V}(t_{j+1})$ are determined by solving the Sylvester equations

$$\begin{cases} (\boldsymbol{I} - \Delta t \boldsymbol{A}_{1,U})\boldsymbol{U}^{(j+1)} + \boldsymbol{U}^{(j+1)}(-\Delta t \boldsymbol{A}_{2,U}^\top) = \boldsymbol{U}^{(j)} - \boldsymbol{B}_{1,U}^\top \boldsymbol{P}^{(j+1)} - \Delta t \boldsymbol{F}_U(\boldsymbol{U}^{(j)}, \boldsymbol{V}^{(j)}, t), \\ (\boldsymbol{I} - \Delta t \boldsymbol{A}_{1,V})\boldsymbol{V}^{(j+1)} + \boldsymbol{V}^{(j+1)}(-\Delta t \boldsymbol{A}_{2,V}^\top) = \boldsymbol{V}^{(j)} - \boldsymbol{P}^{(j+1)}\boldsymbol{B}_{2,V} - \Delta t \boldsymbol{F}_V(\boldsymbol{U}^{(j)}, \boldsymbol{V}^{(j)}, t). \end{cases} \tag{22}$$

Keeping the pressure term at the next time step on the right hand side of the Sylvester equations is a slight abuse of notation. In fact, the pressure is determined by a pressure correction to enforce the incompressibility.

More precisely, consider the implicit time discretization of the pressure, that is $\boldsymbol{U}^{(j+1)} - \boldsymbol{U}^{(j)} = \Delta t \boldsymbol{B}_{1,U}^\top \boldsymbol{P}^{(j+1)}$ and $\boldsymbol{V}^{(j+1)} - \boldsymbol{V}^{(j)} = \Delta t \boldsymbol{P}^{(j+1)}\boldsymbol{B}_{2,V}$. If we multiply the first equation from the left by $\boldsymbol{B}_{1,U}$ and the second from the right by $\boldsymbol{B}_{2,V}^\top$ adding the two equations together, we obtain a Sylvester equation of the form

$$\boldsymbol{A}_{1,U}\boldsymbol{P}^{(j+1)} + \boldsymbol{P}^{(j+1)}\boldsymbol{A}_{2,V}^\top = \boldsymbol{B}_{1,U}\boldsymbol{U}^{(j)} + \boldsymbol{V}^{(j)}\boldsymbol{B}_{2,V}^\top, \tag{23}$$

to be solved for $\boldsymbol{P}^{(j+1)}$. This equation is obtained by enforcing the incompressibility such that $\boldsymbol{B}_{1,U}\boldsymbol{U}^{(j+1)} + \boldsymbol{V}^{(j+1)}\boldsymbol{B}_{2,V}^\top = 0$. Determining the pressure correction does not cause any problems with respect to the staggered grid, since the divergence of the velocity lies in the cell centres, similar to the pressure. Determining the nonlinear terms is, however, more complicated.

More precisely, due to the staggered grid, the nodes of the of $\boldsymbol{U}$ and $\boldsymbol{V}$ are located at different positions so that, for example, the product $\boldsymbol{U} \bullet \boldsymbol{V}$ is not defined. This is circumvented by means of interpolation, for which we refer to [45, section 5] for details. To this end, following [45, section 5] to incorporate the boundary conditions in the nonlinear terms, we define $\overline{\boldsymbol{U}} \in \mathbb{R}^{(n_x+1)\times n_y}$ as the matrix $\boldsymbol{U}$ padded with boundary conditions in the top and bottom rows. Similarly, $\overline{\boldsymbol{V}} \in \mathbb{R}^{n_x \times n_y+1}$ is padded with boundary conditions in the first and last columns. Then, defining $\boldsymbol{C} \in \mathbb{R}^{(n_x-1)\times(n_x+1)}$ as a matrix with $1/2$ on the main and upper diagonal as an averaging matrix, the nonlinear term, evaluated on the staggered grid, can be expressed in fully matricial form as

$$\boldsymbol{F}_U(\boldsymbol{U}, \boldsymbol{V}, t) = \boldsymbol{B}_{1,U}^\top \left((\boldsymbol{C}^\top \overline{\boldsymbol{U}})^2 - \gamma |\boldsymbol{C}^\top \overline{\boldsymbol{U}}| \bullet \left(\frac{h_x}{2} \boldsymbol{B}_{1,U}^\top \overline{\boldsymbol{U}} \right) \right) + \overline{\boldsymbol{B}}_{1,U}^\top \left((\overline{\boldsymbol{U}\boldsymbol{C}}) \bullet (\boldsymbol{C}^\top \overline{\boldsymbol{V}}) - \gamma |(\overline{\boldsymbol{U}\boldsymbol{C}}| \bullet \left(\frac{h_x}{2} \overline{\boldsymbol{B}}_{2,V}^\top \overline{\boldsymbol{V}} \right) \right), \tag{24}$$

where $\frac{h_x}{2}\boldsymbol{B}_{1,U}^\top$ is responsible for differencing and cofficient matrices containing a superscripted bar are merely conforming to the dimension of $\overline{\boldsymbol{U}}$ or $\overline{\boldsymbol{V}}$. A similar form can be derived for $\boldsymbol{F}_V(\boldsymbol{U}, \boldsymbol{V}, t)$. In this setting, $\gamma \in (0, 1)$ is responsible for a smooth transition between upwinding and centered differencing, and is defined as

$$\gamma = \min\Big(1.2\Delta t \max\big(\max|\boldsymbol{U}_{i,j}|, \max|\boldsymbol{V}_{i,j}|\big), 1\Big);$$

see Seibold [45, section 5] for further details.

A summary of the procedure for solving (2) on a staggered grid in fully matricial form can be found in Algorithm 1. The most computationally expensive step is the solution of three Sylvester equations (one at Step 3, and two at step 5) at each timestep. However, the coefficient matrices remain constant throughout the timespan. To this end, an a-priori eigenvalue decomposition of the six coefficient matrices can be performed so that the Sylvester equations can be solved by using only substitution and matrix-matrix multiplication. We refer the reader to [37,38,40] for further details.

Algorithm 1 Full model Algorithm.

1: Choose the initial conditions $(\boldsymbol{U}(0), \boldsymbol{V}(0))$ and the number of time steps n_t
2: **for** $i = 0, \dots, n_t - 1$ **do**
3: Solve (23) for $\boldsymbol{P}^{(i+1)}$
4: Compute $\boldsymbol{F}_U(\boldsymbol{U}, \boldsymbol{V}, t)$ and $\boldsymbol{F}_V(\boldsymbol{U}, \boldsymbol{V}, t)$
5: Solve (21)
6: **end for**

3.3. The 2S-POD-DEIM reduced NS equation

For the reduced model we consider the 2S-POD-DEIM model reduction procedure for systems of matrix differential equations from Kirsten [42], Kirsten and Simoncini [47], discussed above, to reduce (21) in a fully matricial way. To this end we consider n_s snapshots of the full dimensional solutions $\boldsymbol{U}_i, \boldsymbol{V}_i$ and $\boldsymbol{P}_i$, $i = 1, 2, \dots, n_s$, in order to construct the low dimensional, orthonormal basis matrices $\boldsymbol{U}_* \in \mathbb{R}^{(n_x-1)\times k_{1,*}}$, $\boldsymbol{V}_* \in \mathbb{R}^{n_y-1\times k_{2,*}}$ and $\boldsymbol{P}_* \in \mathbb{R}^{n_x\times k_{3,*}}$, where $* = \{\ell, r\}$. This leads to the approximations

$$\boldsymbol{U} \approx \boldsymbol{U}_\ell \widehat{\boldsymbol{U}} \boldsymbol{U}_r^\top =: \widetilde{\boldsymbol{U}}, \qquad \boldsymbol{V} \approx \boldsymbol{V}_\ell \widehat{\boldsymbol{V}} \boldsymbol{V}_r^\top =: \widetilde{\boldsymbol{V}} \quad \text{and} \quad \boldsymbol{P} \approx \boldsymbol{P}_\ell \widehat{\boldsymbol{P}} \boldsymbol{P}_r^\top =: \widetilde{\boldsymbol{P}}.$$

3.3.1. Solving for $\widehat{U}^{(j+1)}$ and $\widehat{V}^{(j+1)}$

Substituting the above approximations into (21) yields a reduced Navier-Stokes equations, where the reduced solutions $\widehat{\boldsymbol{U}}^{j+1}$ and $\widehat{\boldsymbol{V}}^{j+1}$ are determined by solving the (reduced) coupled Sylvester equations

$$\begin{cases} (\boldsymbol{I} - \Delta t \widehat{\boldsymbol{A}}_{1,U}) \widehat{\boldsymbol{U}}^{(j+1)} + \widehat{\boldsymbol{U}}^{(j+1)} (-\Delta t \widehat{\boldsymbol{A}}_{2,U}^\top) = \widehat{\boldsymbol{U}}^{(j)} - \boldsymbol{U}_\ell^\top \boldsymbol{B}_{1,U}^\top \boldsymbol{P}_\ell \widehat{\boldsymbol{P}}^{(j+1)} \boldsymbol{P}_r^\top \boldsymbol{U}_r - \Delta t \widehat{\boldsymbol{F}}_U(\widetilde{\boldsymbol{U}}^{(j)}, \widetilde{\boldsymbol{V}}^{(j)}, t), \\ (\boldsymbol{I} - \Delta t \widehat{\boldsymbol{A}}_{1,V}) \widehat{\boldsymbol{V}}^{(j+1)} + \widehat{\boldsymbol{V}}^{(j+1)} (-\Delta t \widehat{\boldsymbol{A}}_{2,V}^\top) = \widehat{\boldsymbol{V}}^{(j)} - \boldsymbol{V}_\ell^\top \boldsymbol{P}_\ell \widehat{\boldsymbol{P}}^{(j+1)} \boldsymbol{P}_r^\top \boldsymbol{B}_{2,V} \boldsymbol{V}_r - \Delta t \widehat{\boldsymbol{F}}_V(\widetilde{\boldsymbol{U}}^{(j)}, \widetilde{\boldsymbol{V}}^{(j)}, t). \end{cases} \tag{25}$$

Here we have used the same numerical integration scheme as for the full-dimensional equation, and all the matrices that have a $\widehat{}$ on the top are left and right projections of the original coefficient matrices onto the relevant subspaces, and hence they are all low-dimensional. Furthermore, notice that the terms multiplying the pressure term from the left and right in both equations are low-dimensional and time-independent, hence they can be stored offline. Consequently, the remaining challenges lie in determining the pressure term $\widehat{\boldsymbol{P}}^{(j+1)}$ and evaluating the nonlinear functions in low-dimension at each timestep.

3.3.2. Solving for $\widehat{P}^{(j+1)}$

The pressure correction step requires the solution of the Sylvester Eq. (23). Inserting the approximation $\widetilde{\boldsymbol{P}}^{j+1}$ into (23) yields the low-dimensional Sylvester equation

$$\boldsymbol{P}_\ell^\top \boldsymbol{A}_{1,U} \boldsymbol{P}_\ell \widehat{\boldsymbol{P}}^{(j+1)} + \widehat{\boldsymbol{P}}^{(j+1)} \boldsymbol{P}_r^\top \boldsymbol{A}_{2,V}^\top \boldsymbol{P}_r = \boldsymbol{P}_\ell^\top \boldsymbol{B}_{1,U} \overline{\boldsymbol{U}_\ell \widehat{\boldsymbol{U}}^{(j)} \boldsymbol{U}_r^\top} \boldsymbol{P}_r + \boldsymbol{P}_\ell^\top \overline{\boldsymbol{V}_\ell \widehat{\boldsymbol{V}}^{(j)} \boldsymbol{V}_r^\top} \boldsymbol{B}_{2,V}^\top \boldsymbol{P}_r. \tag{26}$$

Here, $\overline{\boldsymbol{U}_\ell \widehat{\boldsymbol{U}}^{(j)} \boldsymbol{U}_r^\top}$ and $\overline{\boldsymbol{V}_\ell \widehat{\boldsymbol{V}}^{(j)} \boldsymbol{V}_r^\top}$ represent the lifted quantities padded with boundary conditions, as discussed before. As a result, the left-hand side of this Sylvester equation consists of only small matrices, but the right hand side, on the other hand, requires some more attention to avoid recomputing large matrices, due to the fact that the lifted quantities are padded by boundary conditions. To this end, by taking advantage of the fact that $\boldsymbol{B}_{1,U}$ and $\boldsymbol{B}_{2,V}$ only account for the differentiation, the first and last rows of $\boldsymbol{P}_\ell$ and $\boldsymbol{P}_r$ can be manipulated in such a way that they only act on the boundary conditions so that the internal blocks of $\boldsymbol{P}_\ell$ and $\boldsymbol{B}_{1,U}$ ($\boldsymbol{P}_r$ and $\boldsymbol{B}_{2,V}$) can be multiplied with $\boldsymbol{U}_\ell$ ($\boldsymbol{U}_r$) in order to form low-dimensional matrices that can be stored offline. A similar manipulation is done for the products between $\boldsymbol{U}_r^\top$ and $\boldsymbol{P}_r$ ($\boldsymbol{P}_\ell^\top$ and $\boldsymbol{V}_\ell$) such that only low-dimensional operations need to occur online.

3.3.3. Evaluating $\widehat{F}_U(\widetilde{U}, \widetilde{V}, t)$ and $\widehat{F}_V(\widetilde{U}, \widetilde{V}, t)$ with DEIM

Following the procedure in [17], we consider n_S snapshots of the nonlinear functions $\boldsymbol{F}_U$ and $\boldsymbol{F}_V$ to construct the low-dimensional, orthonormal matrices $\boldsymbol{\Phi}_{\boldsymbol{U},*} \in \mathbb{R}^{(n_x-1)\times p_{1,*}}$ and $\boldsymbol{\Phi}_{\boldsymbol{V},*} \in \mathbb{R}^{(n_y-1)\times p_{2,*}}$, $* = \{\ell, r\}$ used for the reduction of the nonlinear function by 2S-DEIM. If we define the orthonormal matrices $\boldsymbol{D}_{\boldsymbol{U},*} \in \mathbb{R}^{(n_x-1)\times p_{1,*}}$ and $\boldsymbol{D}_{\boldsymbol{V},*} \in \mathbb{R}^{(n_y-1)\times p_{2,*}}$ as matrices with a subset of columns of the identity matrix, then the 2S-DEIM approximation of the nonlinear terms is given by

$$\widehat{F}_U(\widetilde{\boldsymbol{U}}, \widetilde{\boldsymbol{V}}, t) \approx \boldsymbol{U}_\ell^\top \boldsymbol{\Phi}_{\boldsymbol{U},\ell} \left(\boldsymbol{D}_{\boldsymbol{U},\ell}^\top \boldsymbol{\Phi}_{\boldsymbol{U},\ell} \right)^{-1} \boldsymbol{D}_{\boldsymbol{U},\ell}^\top F_U(\widetilde{\boldsymbol{U}}, \widetilde{\boldsymbol{V}}, t) \boldsymbol{D}_{\boldsymbol{U},r} \left(\boldsymbol{\Phi}_{\boldsymbol{U},r}^\top \boldsymbol{D}_{\boldsymbol{U},r} \right)^{-1} \boldsymbol{\Phi}_{\boldsymbol{U},r}^\top \boldsymbol{U}_r, \tag{27}$$

and similar for $\widehat{F}_V(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}},t)$. The matrices $\boldsymbol{U}_\ell^\top\boldsymbol{\Phi}_{\boldsymbol{U},\ell}\left(\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{\Phi}_{\boldsymbol{U},\ell}\right)^{-1}$ and $\boldsymbol{U}_r^\top\boldsymbol{\Phi}_{\boldsymbol{U},r}\left(\boldsymbol{D}_{\boldsymbol{U},r}^\top\boldsymbol{\Phi}_{\boldsymbol{U},r}\right)^{-1}$ are low-dimensional and can be stored offline, however in this setting it is particularly challenging to evaluate the term $\boldsymbol{D}_{\boldsymbol{U},\ell}^\top F_U(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}},t)\boldsymbol{D}_{\boldsymbol{U},r}$ without first lifting and evaluating $F_U(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}},t)$ in full dimension. In what follows we briefly discuss how this is achieved. The same idea follows for $F_V(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}},t)$.

We want to determine (27) by using only small matrices. From (24), it can be seen that F_U consists of two terms summed together. Therefore:

$$\boldsymbol{D}_{\boldsymbol{U},\ell}^\top F_U(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}},t)\boldsymbol{D}_{\boldsymbol{U},r}=\boldsymbol{D}_{\boldsymbol{U},\ell}^\top F_{U,1}(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}},t)\boldsymbol{D}_{\boldsymbol{U},r}+\boldsymbol{D}_{\boldsymbol{U},\ell}^\top F_{U,2}(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}},t)\boldsymbol{D}_{\boldsymbol{U},r}.$$

In the following result we illustrate how the first of the two terms can be evaluated in low dimension. A similar strategy is used for the second term, but for the sake of presentation we omit the details.

Proposition 1. *The term $\boldsymbol{D}_{\boldsymbol{U},\ell}^\top F_{U,1}(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}},t)\boldsymbol{D}_{\boldsymbol{U},r}$ can be evaluated completely in low-dimension, independent of the full dimensions n_x and n_y, at each time step.*

Proof. From the definition of the full-dimensional nonlinear term, it can be seen that

$$\boldsymbol{D}_{\boldsymbol{U},\ell}^\top F_{U,1}(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}},t)\boldsymbol{D}_{\boldsymbol{U},r}=\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{B}_{1,U}^\top\left((\boldsymbol{C}^\top\widetilde{\boldsymbol{U}})^2-\gamma|\boldsymbol{C}^\top\widetilde{\boldsymbol{U}}|\bullet\left(\frac{h_x}{2}\boldsymbol{B}_{1,U}^\top\widetilde{\boldsymbol{U}}\right)\right)\boldsymbol{D}_{\boldsymbol{U},r}.$$

As a result,

$$\begin{aligned}&\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{B}_{1,U}^\top\left((\boldsymbol{C}^\top\widetilde{\boldsymbol{U}})^2-\gamma|\boldsymbol{C}^\top\widetilde{\boldsymbol{U}}|\bullet\left(\frac{h_x}{2}\boldsymbol{B}_{1,U}^\top\widetilde{\boldsymbol{U}}\right)\right)\boldsymbol{D}_{\boldsymbol{U},r}\\&=\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{B}_{1,U}^\top\left(\boldsymbol{C}^\top\widetilde{\boldsymbol{U}}\bullet\boldsymbol{C}^\top\widetilde{\boldsymbol{U}}\right)\boldsymbol{D}_{\boldsymbol{U},r}-\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{B}_{1,U}^\top\gamma|\boldsymbol{C}^\top\widetilde{\boldsymbol{U}}|\bullet\left(\frac{h_x}{2}\boldsymbol{B}_{1,U}^\top\widetilde{\boldsymbol{U}}\right)\boldsymbol{D}_{\boldsymbol{U},r}.\end{aligned}$$

Once again we look at the two terms on the right-hand side separately. More precisely, considering the first term, the role of $\boldsymbol{D}_{\boldsymbol{U},r}$ is to select columns after the scalar product. Therefore, it can be taken inside of the scalar product, such that

$$\begin{aligned}&\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{B}_{1,U}^\top\left(\boldsymbol{C}^\top\widetilde{\boldsymbol{U}}\bullet\boldsymbol{C}^\top\widetilde{\boldsymbol{U}}\right)\boldsymbol{D}_{\boldsymbol{U},r}=\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{B}_{1,U}^\top\left(\boldsymbol{C}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\bullet\boldsymbol{C}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\right)\boldsymbol{D}_{\boldsymbol{U},r}\\&=\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{B}_{1,U}^\top\left(\boldsymbol{C}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\boldsymbol{D}_{\boldsymbol{U},r}\bullet\boldsymbol{C}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\boldsymbol{D}_{\boldsymbol{U},r}\right).\end{aligned}$$

The term $\boldsymbol{U}_r\boldsymbol{D}_{\boldsymbol{U},r}$ is small and can be saved offline. The role of the term $\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{B}_{1,U}^\top$ is to select at which rows the derivatives is taken after the scalar product. Therefore for each row e_i selected we need to compute $\frac{e_{i+1}-e_i}{h_x}$. This means we only need the scalar product at rows i and $i+1$. Hence,

$$\begin{aligned}&\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{B}_{1,U}^\top\left(\boldsymbol{C}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\boldsymbol{D}_{\boldsymbol{U},r}\bullet\boldsymbol{C}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\boldsymbol{D}_{\boldsymbol{U},r}\right)\\&=\frac{1}{h_x}\left((\boldsymbol{D}_{\boldsymbol{U},\ell}^+)^\top\boldsymbol{C}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\boldsymbol{D}_{\boldsymbol{U},r}\bullet(\boldsymbol{D}_{\boldsymbol{U},\ell}^+)^\top\boldsymbol{C}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\boldsymbol{D}_{\boldsymbol{U},r}\right)\\&\quad-\frac{1}{h_x}\left(\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{C}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\boldsymbol{D}_{\boldsymbol{U},r}\bullet\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{C}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\boldsymbol{D}_{\boldsymbol{U},r}\right)\end{aligned}$$

can be computed via only low-dimensional evaluations, since the only time-dependent term is $\widehat{\boldsymbol{U}}(t)$ and all other matrix products result in low-dimensional coefficient matrices that can be stored offline. The matrix $\boldsymbol{D}_{\boldsymbol{U},\ell}^+$ contains the DEIM indices shifted by $+1$. The same idea works for the second term

$$\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{B}_{1,U}^\top\gamma|\boldsymbol{C}^\top\widetilde{\boldsymbol{U}}|\bullet\left(\frac{h_x}{2}\boldsymbol{B}_{1,U}^\top\widetilde{\boldsymbol{U}}\right)\boldsymbol{D}_{\boldsymbol{U},r},$$

which can be expressed as

$$\begin{aligned}&\frac{1}{h_x}\gamma\left|(\boldsymbol{D}_{\boldsymbol{U},\ell}^+)^\top\boldsymbol{C}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\boldsymbol{D}_{\boldsymbol{U},r}\right|\bullet\left(\frac{h_x}{2}(\boldsymbol{D}_{\boldsymbol{U},\ell}^+)^\top\boldsymbol{B}_{1,U}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\boldsymbol{D}_{\boldsymbol{U},r}\right)\\&-\frac{1}{h_x}\gamma\left|\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{C}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\boldsymbol{D}_{\boldsymbol{U},r}\right|\bullet\left(\frac{h_x}{2}\boldsymbol{D}_{\boldsymbol{U},\ell}^\top\boldsymbol{B}_{1,U}^\top\boldsymbol{U}_\ell\widehat{\boldsymbol{U}}(t)\boldsymbol{U}_r^\top\boldsymbol{D}_{\boldsymbol{U},r}\right).\end{aligned}$$

Once again all coefficient matrices are low-dimensional and stored offline, so that only low-dimensional matrix multiplications are necessary to obtain the term $\boldsymbol{D}_{\boldsymbol{U},\ell}^\top F_{U,1}(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}},t)\boldsymbol{D}_{\boldsymbol{U},r}$. This completes the proof. □

A brief summary of the reduced model phase has been sketched in Algorithm 2[1]

[1] A Matlab implementation of both the full and reduced matrix solvers for the discrete NS equation can be downloaded from https://www.sites.google.com/view/gerhard-kirsten/software-upon-acceptance-of-this-article.

Algorithm 2 Reduced model Algorithm.

Consider the projected initial conditions $(\widehat{\boldsymbol{U}}(0), \widehat{\boldsymbol{V}}(0))$ and choose the number of time steps n_t
for $i = 0, \dots, n_t - 1$ **do**
  Solve (26) for $\widehat{\boldsymbol{P}}^{(i+1)}$
  Evaluate $\widehat{F}_U(\widetilde{\boldsymbol{U}}, \widetilde{\boldsymbol{V}}, t)$ and $\widehat{F}_V(\widetilde{\boldsymbol{U}}, \widetilde{\boldsymbol{V}}, t)$ with DEIM
  Solve (25) for $\widehat{\boldsymbol{U}}^{(i+1)}$ and $\widehat{\boldsymbol{V}}^{(i+1)}$
end for

4. The tree structure algorithm for the NS equation

In this section we are going to couple the multilinear approximation of the Navier-Stokes equation with the Tree Structure Algorithm (TSA), an algorithm to approximate the HJB equation arising from the optimal control problem. We will first introduce briefly the general procedure for the TSA and next we are going to present the coupling of these two techniques.

4.1. Dynamic programming on a tree structure

We introduce the essential ingredients of the DP approach based on a tree built on the discrete dynamical system. The interested reader will find more details on the topic in Alla et al. [18].

We consider the discrete approximation of the DP principle. Fixed the number of time steps n_t and the time step $\Delta t := [(T-t)/n_t]$, the discrete DP reads

$$\begin{cases} V^n = \min\limits_{a\in A}[\Delta t\, L(x,a,t_n) + e^{-\lambda\Delta t} V^{n+1}(x + \Delta t f(x,a,t_n))], & n = n_t - 1, \dots, 0, \\ V^{n_t} = g(x), x \in \mathbb{R}^d, \end{cases} \tag{28}$$

where $t_n = t + n\Delta t$, $t_{\overline{N}} = T$, and $V^n := V(x, t_n)$. We discretize the control set A with step-size Δa obtaining a discrete control set with a finite number of controls $A^{\Delta u} = \{a_1, \dots, a_M\}$. In what follows we denote by A the discrete set to ease the notation.

Now, we start from the initial condition x and we follow the discrete dynamics employing the explicit Euler scheme and M discrete controls a_j

$$\zeta_j^1 = x + \Delta t\, f(x, a_j, t_0), \qquad j = 1, \dots, M. \tag{29}$$

Therefore, denoting the root of the tree with $\mathcal{T}^0 = \{x\}$, we get the first level of the tree $\mathcal{T}^1 = \{\zeta_1^1, \dots, \zeta_M^1\}$. The procedure can be iterated so that the $n-$th *time level* will be given by

$$\mathcal{T}^n = \{\zeta_i^{n-1} + \Delta t f(\zeta_i^{n-1}, a_j, t_{n-1})\}_{j=1}^M \quad i = 1, \dots, M^{n-1},$$

and the entire tree can be represented as

$$\mathcal{T} := \{\zeta_j^n\}_{j=1}^{M^n}, \quad n = 0, \dots, n_t,$$

where ζ_i^n is the evolution of the dynamics at time t_n using the controls $\{a_{j_k}\}_{k=0}^{n-1}$:

$$\zeta_{i_n}^n = \zeta_{i_{n-1}}^{n-1} + \Delta t f(\zeta_{i_{n-1}}^{n-1}, a_{j_{n-1}}, t_{n-1}) = x + \Delta t \sum_{k=0}^{n-1} f(\zeta_{i_k}^k, a_{j_k}, t_k),$$

with $\zeta^0 = x$, $i_k = \lfloor \frac{i_{k+1}}{M} \rfloor$ and $j_k \equiv i_{k+1} \bmod M$.

Although the TSA allows to deal with high-dimensional problems, the cardinality of tree grows exponentially in the time steps and in the number of nodes, *i.e.* $|\mathcal{T}| = O(M^{n_t})$, yielding problems in the memory allocations. For this reason we introduce a *pruning criteria* based on the distance between nodes. Therefore, two nodes ζ_i^n and ζ_j^n will be merged if

$$\|\zeta_i^n - \zeta_j^n\| \le \varepsilon_{\mathcal{T}}, \quad \text{with } i \neq j \text{ and } n = 0, \dots, n_t, \tag{30}$$

for a given threshold $\varepsilon_{\mathcal{T}} > 0$. In Saluzzi et al. [19] the authors show that the threshold $\varepsilon_{\mathcal{T}} > 0$ must scale quadratically in the time steps to ensure first order convergence. Furthermore, in Saluzzi et al. [19] the authors achieve a result on the cardinality of the pruned tree in case of linear dynamics. It turns out that this cardinality grows quadratically in the number of time steps and linearly in the number of discrete controls.

Once constructed the tree $\mathcal{T}$, we can pass to the computation of the numerical value function $V(x,t)$. The TSA defines a time dependent grid $\mathcal{T}^n = \{\zeta_j^n\}_{j=1}^{M^n}$ for $n = 0, \dots, n_t$ and (14) can be approximated as follows:

$$\begin{cases} V^n(\zeta_i^n) = \min\limits_{a\in A}\{e^{-\lambda\Delta t} V^{n+1}(\zeta_i^n + \Delta t f(\zeta_i^n, a, t_n)) + \Delta t\, L(\zeta_i^n, a, t_n)\}, \\ \qquad\qquad\qquad\qquad\qquad\qquad \zeta_i^n \in \mathcal{T}^n, n = n_t - 1, \dots, 0, \\ V^{n_t}(\zeta_i^{n_t}) = g(\zeta_i^{n_t}), \qquad\qquad\qquad\qquad \zeta_i^{n_t} \in \mathcal{T}^{n_t}. \end{cases} \tag{31}$$

The minimization in (31) is solved by comparison on the discrete set A.

4.2. Coupling TSA and 2S-POD-DEIM

Introduced the main ingredients for the multilinear approximation of the NS equation and the TSA, in this section we are going to show how to couple these two techniques in order to solve the optimal control problem. The procedure is divided into two steps: an offline and an online phase.

- **Offline Phase**
  In the offline phase we build the 2S-POD-DEIM basis and construct the reduced dynamics which will be employed in the online phase. In this step we explore the manifold of possible evolutions of the controlled dynamics and we aim to capture the main features of the dynamical system. We first fix a time step $\widehat{\Delta t}$ and $\widehat{M}$ number of discrete controls. Following Algorithm 1, the tree structure is constructed in the full dimension with the fixed parameters $\widehat{\Delta t}$ and $\widehat{M}$. Since at this stage the problem is high-dimensional, few time steps and few controls will be selected for the offline phase. At each node of the full-dimensional tree, a snapshot of the solution at that node is used to update the 2S-POD-DEIM basis matrices via the algorithm in [17, Section 3]; see step in Algorithm 3 below. As soon as the basis matrices have been

Algorithm 3 Offline Phase.

```
Consider M discrete controls and n_t discrete timesteps
for n = 0, ..., n_t − 1 do
    for i = 1, ..., M^n do
        for k = 1, ..., M do
            Consider control u_k and determine U_{i,k}^{n+1}, V_{i,k}^{n+1}, P_{i,k}^{n+1}, F_U(U_{i,k}^{n+1}, V_{i,k}^{n+1}, t_n) and F_V(U_{i,k}^{n+1}, V_{i,k}^{n+1}, t_n) using Algorithm 1
            Update the relevant basis vectors in U_*, V_*, P_*, Φ_{U,*}, Φ_{V,*}, by means of a dynamic manipulation of the leading singular vectors of U_{i,k}^{n+1}, V_{i,k}^{n+1}, P_{i,k}^{n+1}, F_U(U_{i,k}^{n+1}, V_{i,k}^{n+1}, t_n) and F_V(U_{i,k}^{n+1}, V_{i,k}^{n+1}, t_n).
        end for
    end for
end for
```

  updated, the snapshot is discarded and a further step is taken along the tree. In short, the offline procedure consists of the following steps:
  In Algorithm 3 we consider $* = \{\ell, r\}$ and the notation $\boldsymbol{Z}_{i,k}^{n+1}$ refers to the node constructed starting from the i-th node at time level n with control u_k. Once all tree nodes have been considered, the collected left and right singular vectors are pruned to further reduce the dimension of each basis relative to a selected tolerance tol. Furthermore, the permutation matrices $\boldsymbol{D}_{\boldsymbol{U},*} \in \mathbb{R}^{(n_x-1)\times p_{1,*}}$ and $\boldsymbol{D}_{\boldsymbol{V},*} \in \mathbb{R}^{(n_y-1)\times p_{2,*}}$ are respectively constructed using $\boldsymbol{\Phi}_{\boldsymbol{U},*} \in \mathbb{R}^{(n_x-1)\times p_{1,*}}$ and $\boldsymbol{\Phi}_{\boldsymbol{V},*} \in \mathbb{R}^{(n_y-1)\times p_{2,*}}$. Finally all matrix quantities that are not time or control dependent are computed and stored in this phase, avoiding their recomputation in the online phase.
- **Online Phase**
  In the online phase we solve the optimal control problem via TSA directly on the reduced model constructed in the offline phase. Since we reduced the dimension of the system, in this step it is possible to consider more time steps and/or more discrete controls with respect to the offline phase. First of all we construct the reduced tree following Algorithm 2. The nodes of the tree represent the grid for the numerical resolution of the DDP (31). At this point we can solve the DPP on the tree structure, obtaining the discrete value function $\{V_{red}^n(\zeta_i^n)\}_{i,n}$ on the tree. The last part of this phase concerns the reconstruction of the control signal and the controlled trajectory. Starting from the initial condition, *i.e.* the root of the tree, we can follow the branches returning the minimum

$$\alpha_*^n := \arg\min_{a\in U} \left\{ e^{-\lambda \Delta t} V_{red}^{n+1,\ell}(\zeta_{red}^n + \Delta t\, f_{red}(\zeta_{red}^n, a, t_n)) + \Delta t\, L_{red}(\zeta_{red}^n, a, t_n) \right\}.$$

  The computed control signal can now be plugged into the full-dimensional dynamics to obtain the sub-optimal trajectory in the original dimension.

5. Numerical experiments

In this section we are going to test the proposed technique in different settings. In the first test we compare the performances between the full dimension model and the low dimension one in terms of CPU time. Moreover, the vector and the matricial cases will be compared. In the second and third tests we pass to the optimal control problem of the NS equation, in which we are interested in reaching a particular solution target: the stationary configuration. More precisely, in the second test the control will act on the entire domain via the use of a shape function, while in the third example the control operates on a subdomain w located at the center of the domain. In the last example the control will operate on the boundaries and the reference solution will be represented by the trajectory obtained using a prefixed control. For all the numerical tests we will fix the space domain $\Omega = [0, 1]^2$ and Reynolds number $r = 100$. We consider semi-implicit methods for the time discretization, since they are stable under stability conditions independent of the viscous term, allowing to consider

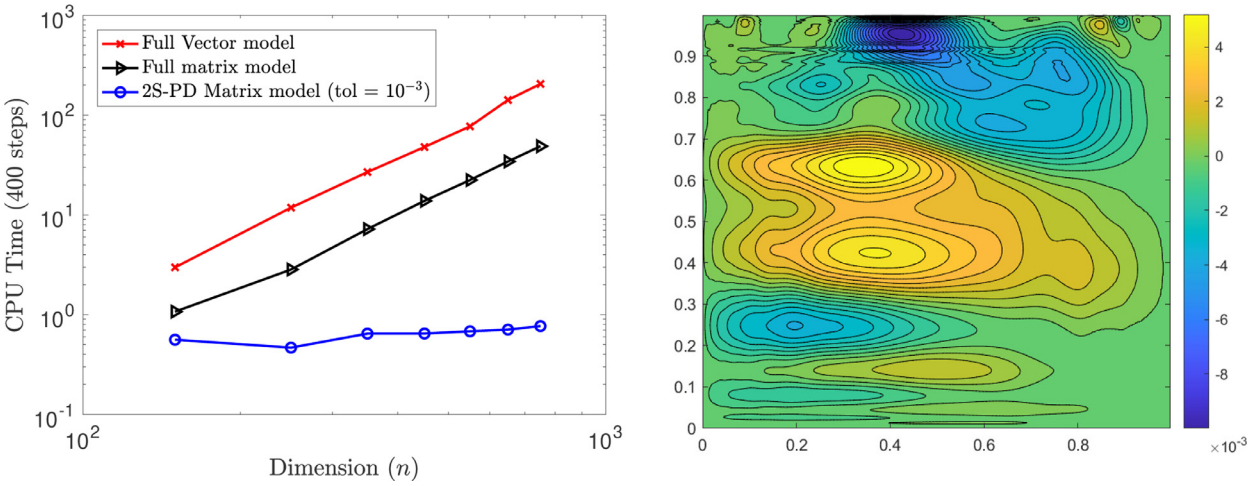

Fig. 1. Test 1: Comparison between the full vector model, the full matrix model and the 2S-POD-DEIM matrix model with $tol = 10^{-3}$ in terms of CPU time varying the space dimension (left) and difference between the full solution and the lifted solution at final time with $n = 750$ (right).

larger times steps also for high Reynolds numbers (see e.g. pg.486 in [48].) In all the tests we consider two discrete controls (the extremes of the interval) as control set for the offline stage. We also give some indications on the efficiency of the pruning technique via the *Pruning Ratio (*$Ratio_p$*)* that is defined as the ratio between the cardinality of the full tree and the cardinality of the pruned tree, *i.e.*

$$Ratio_p = \frac{(M^{n_t+1} - 1)/(M - 1)}{|\mathcal{T}_M|},$$

where $\mathcal{T}_M$ is the tree constructed using M discrete controls under the pruning criteria.

5.1. Test 1: Comparison full/low dimension

In this example we investigate the efficiency of, not only discretizing the NS equation in matrix form, but also reducing the dimension of the resulting matrix equation by 2S-POD-DEIM. To this end we consider the NS Eq. (2), and we fix $T = 20$, $\Delta t = 0.05$ and the tolerance for 2S-POD-DEIM equal to 10^{-3}. We consider as initial condition $u_0 = v_0 \equiv 0$ and the following boundary conditions

$$u(t, x, y) = 1,\ v(t, x, y) = 0, \quad (x, y) \in [0, 1] \times \{1\}, t \in (0, T]$$

and homogeneous Dirichlet conditions on the other walls.

For the experimental setup we consider $n = n_x = n_y$, for $n \in \{150, 250, 350, 450, 550, 650, 750\}$ and measure the CPU time required to evaluate the discrete NS equation at $n_t = \frac{T}{\Delta t} = 400$ timesteps for the full dimensional vector model, the full dimensional matrix model and the 2S-POD-DEIM reduced matrix model. The results for the full dimensional vector model are obtained by running the Matlab software from [45], whereas, for the 2S-POD-DEIM reduced model, we plot the CPU Time required for the online simulation (Algorithm 2) only, to indicate how rapidly the reduced model can be simulated for different parameters once the basis vectors have been constructed. The results are plotted in the left panel of Fig. 1.

It can be deduced from the plot that solving the full dimensional discrete NS equation in matrix form as opposed to vector form results in a good computational gain. This behaviour is typical due to the efficiency of the a-priori eigenvalue decomposition resulting in a simple solve by substitution for the Sylvester equations at each time step; see, e.g., [37,40]. Furthermore, as expected, we notice that the reduced model is several orders of magnitude faster than both full-order model, and the nearly-constant timings for increasing n indicates that the computational cost for the reduced model is indeed completely independent of the full dimension n. For the largest of the considered dimensions ($n = 750$), the dimensions of the basis matrices (given the tolerance $tol = 10^{-3}$) are

$$\boldsymbol{U}_\ell \in \mathbb{R}^{750\times 61}, \boldsymbol{U}_r \in \mathbb{R}^{750\times 58}, \boldsymbol{V}_\ell \in \mathbb{R}^{750\times 57}, \boldsymbol{V}_r \in \mathbb{R}^{750\times 55}, \boldsymbol{P}_\ell \in \mathbb{R}^{750\times 54}, \boldsymbol{P}_r \in \mathbb{R}^{750\times 52}$$

for the linear terms, and

$$\boldsymbol{\Phi}_{\boldsymbol{U},\ell} \in \mathbb{R}^{750\times 64}, \boldsymbol{\Phi}_{\boldsymbol{V},\ell} \in \mathbb{R}^{750\times 59}, \boldsymbol{\Phi}_{\boldsymbol{U},r} \in \mathbb{R}^{750\times 58}, \boldsymbol{\Phi}_{\boldsymbol{V},r} \in \mathbb{R}^{750\times 59}$$

for the nonlinear terms.

In the right panel of Fig. 1 we present the difference

$$err_{U,\hat{U}}(T) = \boldsymbol{U}(T) - \boldsymbol{U}_\ell \widehat{\boldsymbol{U}}(T) \boldsymbol{U}_r^\top,$$

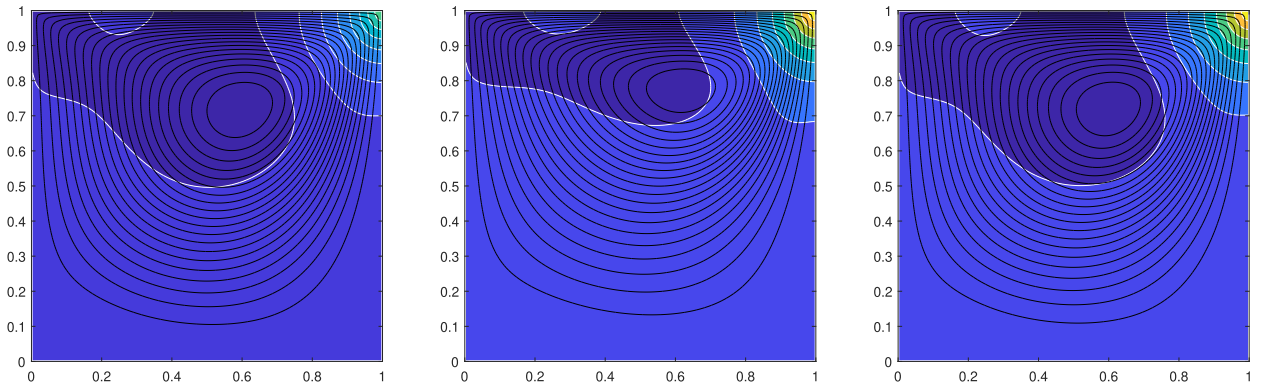

Fig. 2. Test 2: Stationary solution ($t = 20$) (left), uncontrolled solution at $t = 2$ (central) and controlled solution at $t = 2$ (right) for 2S-POD and $n = 201$.

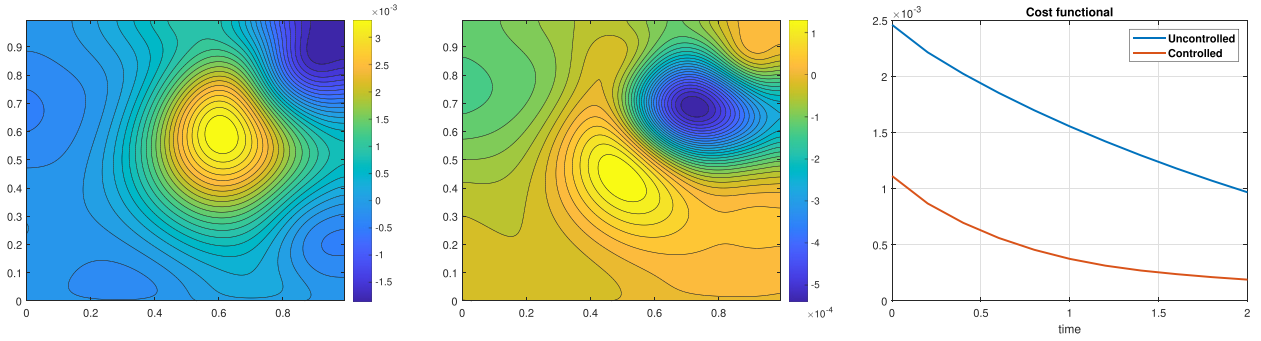

Fig. 3. Test 2: Pressure difference between stationary and uncontrolled solution (left), between stationary and controlled solution (central) at the final time and cost functional (right) for 2S-POD and $n_x = 201$.

which describes the error between the full order model solution and the lifted reduced one at the final time, having fixed $n = 750$. It is clear from the plot that this error has the same order as the chosen tolerance, *i.e.* almost 10^{-3}. At the top wall $err_{U,\hat{U}}(T)$ is slightly higher due to the non-homogeneous boundary condition.

These promising results show that a very fine discretization of the NS equation can be solved in a fraction of a second online with a small projection error of order 10^{-3}.

5.2. Test 2: The problem of long time behaviour of the solution

In this test we want to reach a target solution acting on a scalar control that appears in the Navier-Stokes equation as an additional term as in (3). The initial condition and the boundary conditions coincide with the previous example. We are interested in the stationary solution of the Navier-Stokes equation and it can be obtained by analyzing the long time behaviour of the uncontrolled dynamics. In this case the stationary solution $\bar{y}$ has been fixed as the uncontrolled solution at time $t = 20$, since the solution does not present relevant changes for larger time intervals. The optimal control problem is based on the minimization of the following cost functional

$$J(\alpha) = \int_0^T (\|y(\cdot,t;\alpha) - \bar{y}(\cdot,t)\|^2_{L^2(\Omega)} + \gamma\|\alpha\|^2)e^{-\lambda t}dt + \|y(\cdot,T;\alpha) - \bar{y}(\cdot,T)\|^2_{L^2(\Omega)},$$

with $\gamma = 10^{-3}$. The parameters of the discrete problem are the following: $\Delta t = 0.1$, $A = \{0, 0.5, 1\}$ and $T = 2$. A similar example has been studied in [49]. The geometric pruning criteria will be applied by selecting the threshold $\epsilon_{\mathcal{T}} = \Delta t^2$ to ensure first order convergence. We fix the gridpoints $n_x = n_y = 201$ and we apply the 2S-POD-DEIM technique with tolerance $\tau = 10^{-3}$. The reduction techniques provide the following basis: $\boldsymbol{U}_\ell \in \mathbb{R}^{200\times 61}$, $\boldsymbol{U}_r \in \mathbb{R}^{200\times 57}$, $\boldsymbol{V}_\ell, \boldsymbol{V}_r, \boldsymbol{P}_\ell, \boldsymbol{P}_r \in \mathbb{R}^{200\times 70}$, $\boldsymbol{\Phi}_{\boldsymbol{U},\ell}, \boldsymbol{\Phi}_{\boldsymbol{V},\ell} \in \mathbb{R}^{200\times 62}$ and $\boldsymbol{\Phi}_{\boldsymbol{U},r}, \boldsymbol{\Phi}_{\boldsymbol{V},r} \in \mathbb{R}^{200\times 52}$. In Fig. 2 we display respectively the pressure field computed in the stationary case (left panel) and the uncontrolled (central panel) and the controlled solution (right panel) at time $t = 2$. We show the contour lines of the pressure field and the closed contour lines of the stream function. It is possible to notice visually how the controlled dynamics looks similar to the stationary solution, while the uncontrolled is still far from the asymptotic behaviour. The difference in the pressure field at the final time between the uncontrolled solution and $\tilde{y}$ is displayed in the left panel of Fig. 3, while in the central panel we show the difference between the controlled and stationary solution. We notice that the order in latter case is $\approx 10^{-4}$, while in the first case is $\approx 10^{-3}$, demonstrating how the solution of the optimal control converges more rapidly to the stationary configuration. The right panel of Fig. 3 shows the comparison of the

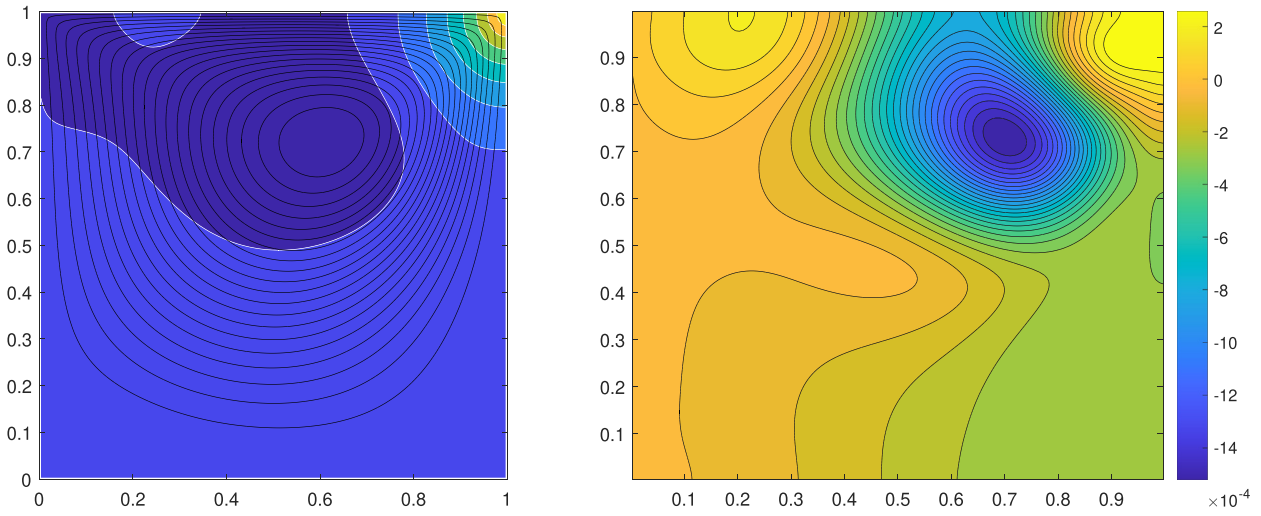

Fig. 4. Test 2: Controlled LQR solution (left) and pressure difference between stationary and LQR solution (right) at time $t = 2$.

Table 1
Test 2: Comparison between the FOM approach and the ROM one in terms of CPU time for the construction of the tree structure.

	$M = 2$	$M = 3$
FOM	234 s	1.79e4 s
ROM	21 s	1738 s

cost functional in the controlled and uncontrolled setting, where we can see again the faster convergence of the controlled dynamics to the stationary solution. In this case the application of the pruning criteria yields a cardinality of the tree equal to 2994, whereas the cardinality of the full tree is 88573, corresponding to a pruning ratio of almost 30. In Table 1 we show a comparison in terms of computational time between the Full Order Model (FOM) and the Reduced Order Model (ROM). We note that the introduction of the ROM approach yields a speed-up of almost 10 in all the test cases.

Now we want to show the efficiency of the TSA considering a comparison with a commonly used tool, the Linear Quadratic Regulator (LQR). We consider the approach presented in [50], where the authors consider a linearization of the NS equation and an application of a Leray projection to enforce the incompressibility constraint. Since we are dealing with a finite horizon optimal control problem, we have to solve a Differential Riccati Equation (DRE). To retain the matrix configuration of the DRE, we consider matrix generalizations of classic BDF methods (see [51,52] for further details). In order to have a reasonable comparison to the first order scheme used in our presented methods, we consider the first order BDF method, fixing the same time step considered for the construction of the tree structure. The results of the application of the LQR approach are shown in Fig. 4. We see that the final configuration of the controlled solution is visually similar to the right panel of Fig. 2, while we note that the pressure difference is greater than the one considered in the central panel of Fig. 3. The TSA represents a better approximation also in terms of total cost, since the LQR achieves a cost of 1.5e-3 while the TSA got 1.1e-3. This demonstrates that taking into account the nonlinear terms in the NS equation we can achieve better results.

5.3. Test 3: Control on an internal subdomain $\omega \subset \Omega$

In this experiment our aim is to reach a target solution acting on a scalar control that appears in the Navier-Stokes equation as an additional term concentrated on a subdomain $\omega \subset \Omega$ as in (3). We consider $\omega = [0.3, 0.7]^2$, *i.e.* the control will operate on a central smaller square. In this example we consider homogeneous Dirichlet boundary conditions for all the walls and our scope is to drive the solution to the equilibrium $\bar{y} \equiv 0$. In this case we select a cost functional depending only on the final cost

$$J(\alpha) = \|y(\cdot, T; \alpha)\|^2_{L^2(\Omega)}.$$

We consider $\Delta t = 0.1$, $T = 2, A = [0, 1]$ and we will vary the number of discrete controls. We consider the following initial condition

$$u_0 = v_0 = \sin(\pi x)\sin(\pi y), \quad (x, y) \in [0, 1]^2.$$

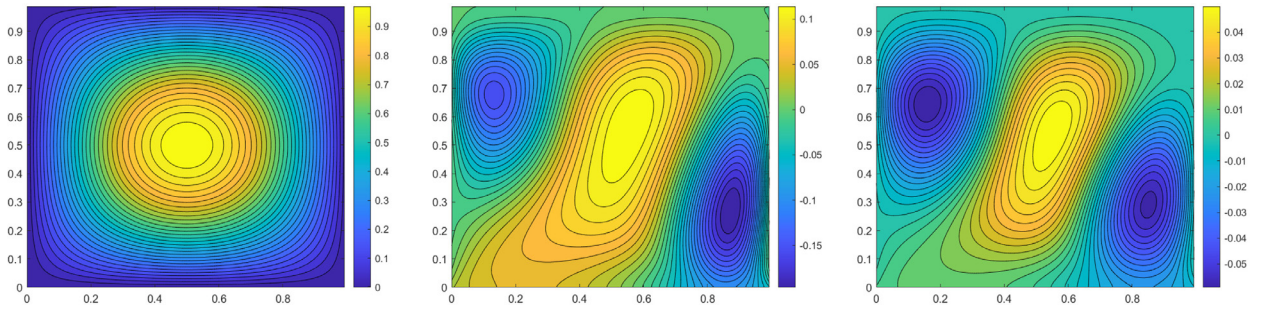

Fig. 5. Test 3: Uncontrolled solution at $t = 0$ (left), $t = 1$ (central) and $t = 2$ (right) for 2S-POD and $n_x = 201$.

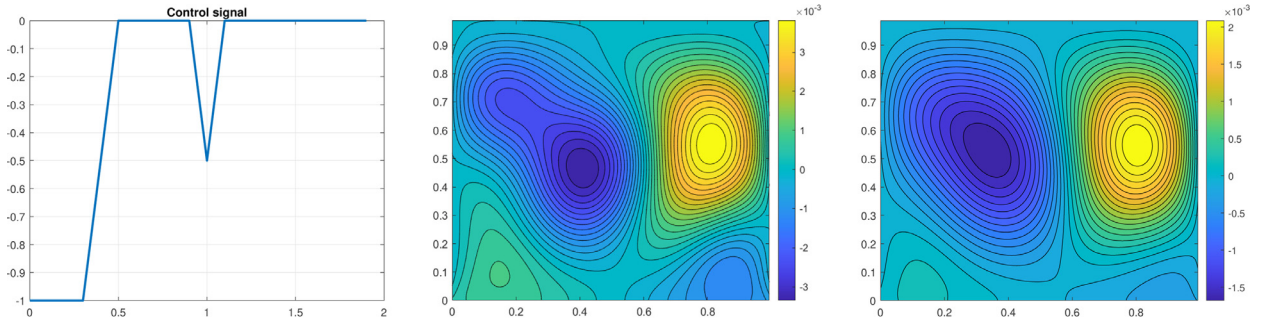

Fig. 6. Test 3: Control signal (left), controlled solution at $t = 1$ (central) and $t = 2$ (right) for 2S-POD and $n_x = 201$.

Table 2
Test 3: Comparison between the uncontrolled dynamics and the controlled one with 2 and 3 discrete controls.

	Cost functional	Nodes	$Ratio_p$
Uncontrolled	3.60e-4		
Controlled $M = 2$	5.44e-5	6064	345
Controlled $M = 3$	5.27e-5	83,273	6.2e4

Table 3
Test 3: Comparison between the FOM approach and the ROM one in terms of CPU time for the construction of the tree structure.

	$M = 2$	$M = 3$
FOM	334 s	2.42e4 s
ROM	31 s	2538 s

In Fig. 5 we show the behaviour of the uncontrolled solution $\boldsymbol{U}$ for different times. We note that the norm of the solution is decreasing due to the viscosity term. The aim of the corresponding optimal control problem is to accelerate this decay.

We apply the 2S-POD-DEIM approach and we construct the following basis: $\boldsymbol{U}_\ell, \boldsymbol{U}_r, \boldsymbol{V}_\ell, \boldsymbol{V}_r \in \mathbb{R}^{200\times 68}$, $\boldsymbol{P}_\ell, \boldsymbol{P}_r \in \mathbb{R}^{200\times 67}$, $\boldsymbol{\Phi}_{\boldsymbol{U},\ell}, \boldsymbol{\Phi}_{\boldsymbol{U},r}, \boldsymbol{\Phi}_{\boldsymbol{V},\ell}, \boldsymbol{\Phi}_{\boldsymbol{V},r} \in \mathbb{R}^{200\times 70}$. Fig. 6 displays the results obtained by the coupling of the TSA and 2S-POD-DEIM. The left panel shows the control signal which presents a non-decreasing behaviour at the beginning of the time interval but starts oscillating in the middle. The central and the right panels we report the controlled solution at the time instances $t \in \{1, 2\}$. We note that the maximum of the controlled solution is order 10^{-3} at time $t = 1$, whereas for the uncontrolled dynamics it is stuck to 10^{-1}. After $t = 1$, the control stops acting and the decrease is just due to viscous term in the equation. In Table 2 we present the comparison between the uncontrolled dynamics and the controlled solution varying the number of discrete controls. In term of the cost functional, the TSA gets almost one order of magnitude with respect to the uncontrolled case and we see an improvement increasing the number of controls. Moreover, we report the cardinality of the tree coupled with the pruning technique. Looking at the P-Ratio we note the pruning criteria yields to a great benefit in terms of memory storage and this improvement increases as we consider more discrete controls. In Table 3 we report the CPU times of the FOM approach and the ROM one. The speed-up ratio between the two approaches is almost 10, as seen in Test 2.

Table 4
Test 4: Comparison between the approximations of the optimal control problem varying the number of controls.

M	Cost functional	Nodes	$Ratio_p$
2	1.00e-6	228	9
3	9.19e-7	710	125
5	1.74-7	4541	2.7e3

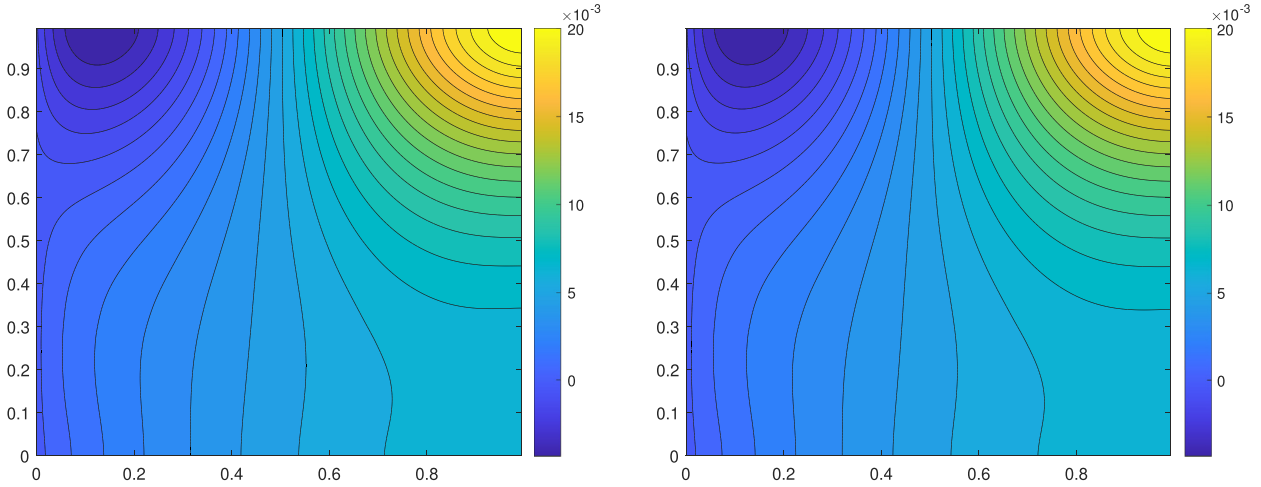

Fig. 7. Test 4: Final configuration of the reference solution (left) and final configuration of the controlled solution with 5 discrete controls (right).

5.4. Test 4: Dirichlet boundary control

In the final example we deal with a boundary optimal control problem, in particular we consider a scalar control acting on the Dirichlet boundary condition. We choose the same initial condition fixed in Test 3, $T = 1$, $\Delta t = 0.1$ complemented with homogeneous Dirichlet boundary conditions for all the walls of the square, except for the top wall where we set the following condition

$$u(t, x, \alpha) = g(x, t, \alpha), \quad (x_1, x_2) \in [0, 1] \times \{1\}, t \in (0, T].$$

To construct our reference trajectory, we run a simulation fixing $g(x, t, \alpha) = x(1 - x) \sin t$ and we compute the corresponding numerical solution that we denote by $\{(\tilde{\boldsymbol{U}}^i, \tilde{\boldsymbol{V}}^i, \tilde{\boldsymbol{P}}^i)\}_{i=1}^{n_t}$. Then, we set the optimal control problem considering the control set $A = [0, 1]$ and the following controlled boundary condition $g(x, t, \alpha) = x(1 - x)\alpha(t)$. In this case the aim of the control problem is to reach the final configuration of the pressure field $\tilde{\boldsymbol{P}}^{n_t}$, so we define the cost functional

$$J(\alpha) = \|P^{n_t}(\alpha) - \tilde{\boldsymbol{P}}^{n_t}\|^2,$$

where $P^{n_t}(\alpha)$ is the solution of the optimal control problem at final time with control α. Note that the running cost is 0 in this example.

The number of discrete controls in this example varies in the set $\{2, 3, 5\}$ and this will correspond to an increasing number of nodes in the tree. We want to examine the efficiency of the method in terms of the its pruning capacity and its accuracy in the approximation of the target solution.

The comparison of the performances of these three cases is reported in Table 4. Note that the cost functional is decreasing to $O(10^{-7})$ as we increase the number of controls. In this example the pruning method is rather efficient, as we can notice by the $Ratio_p$ column. As we increase the parameter M, the value $Ratio_p$ gets one order of magnitude in each step.

In Fig. 7 we show the configuration at final time of the reference solution $\tilde{\boldsymbol{P}}^{n_t}$ in the left panel and the controlled solution fixing $M = 5$ of the controlled solution in the right panel. Visually they look very similar, but this is also certified in the left panel of Fig. 8 showing that the difference between the reference and the controlled solution is order $O(10^{-5})$. This shows that the numerical method is able to reconstruct an optimal control driving the dynamics close to the reference solution. Finally, in the right panel of Fig. 8 the reference control and the numerical approximation are shown, where we can note that the optimal control is trying to mimic the reference signal. Finally, in Table 5 we show the computational costs for the computation of the tree structure in the full dimension (FOM) and in the reduced one (ROM). In this case the ratio between the two approaches is almost 20, demonstrating the faster performances of the proposed technique.

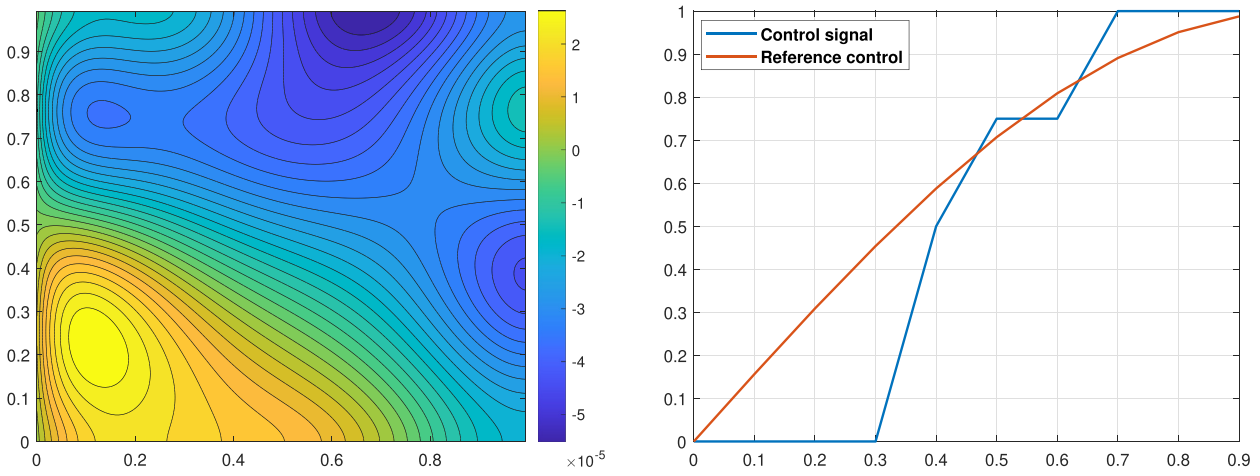

Fig. 8. Test 4: The difference between the reference solution and the controlled solution at final time (left) and comparison between the reference control and the computed optimal control (right).

Table 5
Test 4: Comparison between the FOM approach and the ROM one in terms of CPU time for the construction of the tree structure.

	$M = 2$	$M = 3$
FOM	126 s	5502 s
ROM	6.4 s	277 s

6. Conclusions

In this paper we have presented our first results on the numerical approximation of optimal control problems for the Navier-Stokes equation. The problem is discretized in space to obtain a system of ordinary differential equations, then we set the control problem on the finite dimensional system of ordinary differential equations corresponding to that semi-discretization. A crucial role is played by a very compact representation of the dynamical system and by a tree structure method to solve the problem via Dynamic Programming. More precisely, we have illustrated that by taking advantage of the rectangular domain, and a tensor-structured discretization basis, that the discrete NS equation can be written, integrated and reduced entirely in matrix form, to dramatically reduce the computational cost of integrating the discrete NS equation. On the other hand, the tree structure algorithm is used to counteract the curse of dimensionality arising from the optimal control problem and the Dynamic Programming.

The combination of these two methods shows that the Dynamic Programming approach can be used also in this area and that the synthesis of optimal feedbacks can also be obtained for these huge optimization problems. This is a good omen for the future and we plan to investigate more in detail the convergence of feedback controls and other optimal control problems for fluids.

Data availability

Data will be made available on request.

Acknowledgments

The first and the third authors are members of INDAM GNCS (Gruppo Nazionale di Calcolo Scientifico). This research has been partially supported by the PRIN 2017 project “Innovative Numerical Methods for Evolutionary Partial Differential Equations and Applications”, contract n. 2017KKJP4X.

The second and third authors would like to acknowledge and dedicate this paper to Prof. Maurizio Falcone who passed away shortly before publication. We are forever grateful for your mentorship and guidance. May your memory live long in the mathematical community.

References

[1] O. Pironneau, Finite Element Methods for Fluids, Wiley Chichester, 1989.
[2] J.C. Strikwerda, Finite Difference Schemes and Partial Differential Equations, SIAM, 2004.

[3] P. Benner, M. Ohlberger, A. Cohen, K. Willcox, Model Reduction and Approximation: Theory and Algorithms, SIAM, Philidelphia, 2017.
[4] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM Rev 57 (4) (2015) 483–531.
[5] A. Quarteroni, G. Rozza, Numerical solution of parametrized Navier-Stokes equations by reduced basis methods, Numer Methods Partial Differ Equ 23 (4) (2007) 923–948.
[6] G. Stabile, G. Rozza, Finite volume POD-Galerkin stabilized reduced order methods for the parametrized incompressible Navier-Stokes equations, Computers & Fluids 173 (2018) 923–948.
[7] F. Pichi, M. Strazzullo, F. Ballarin, G. Rozza, Driving bifurcating parametrized nonlinear PDEs by optimal control strategies: application to Navier–Stokes equations with model order reduction, ESAIM: Mathematical Modelling and Numerical Analysis 56 (4) (2022) 1361–1400.
[8] A. Fursikov, O. Imanuvilov, Controllability of Evolution Equations, Seul University Press, 1996.
[9] J.-P. Puel, Controllability of Navier-Stokes equations, in: Optimization with PDE constraints, Springer, 2014, pp. 379–402.
[10] J. Lions, Contrôle Optimal des Systemes Gouverné par des Equations aux Dérivées Partialles Dunod, Paris, 1969 English translation, 1971.
[11] M. Bardi, I. Capuzzo-Dolcetta, Optimal Control and Viscosity Solutions of Hamilton- Jacobi-Bellman Equations, Birkhäuser, 1997.
[12] F. Tröltzsch, Optimal control of Partial Differential Equations - Theory methods and applications, American Mathematical Society, 2010.
[13] E. Casas, Optimal control of PDE theory and numerical analysis, PhD Thesis. Optimization and Control. (2006).
[14] M. Hinze, R. Pinnau, M. Ulbrich, S. Ulbrich, Optimization with PDE Constraints, Mathematical Modelling: Theory and Applications, Springer, 2009.
[15] J. Sethian, Level set methods and fast marching methods, Mathematical Modelling: Theory and Applications, Cambridge University Press, 1999.
[16] M. Falcone, R. Ferretti, Semi-Lagrangian Methods for Linear and Hamilton-Jacobi Equations, SIAM, 2014.
[17] G. Kirsten, V. Simoncini, A matrix-oriented POD-DEIM algorithm applied to nonlinear differential matrix equations, 2020. ArXiv 2006.13289.
[18] A. Alla, M. Falcone, L. Saluzzi, An efficient DP algorithm on a tree-structure for finite horizon optimal control problems, SIAM J. Sci. Comput. 41 (4) (2019) 2384–2406.
[19] L. Saluzzi, A. Alla, M. Falcone, Error estimates for a tree structure algorithm solving finite horizon control problem, ESAIM: Control, Optimisation and Calculus of Variations 28 (2022) 69, doi:10.1051/cocv/2022067.
[20] A. Alla, L. Saluzzi, A HJB-POD approach for the control of nonlinear PDEs on a tree structure, Appl Numer Math. 155 (2020) 192–207.
[21] A. Alla, M. Falcone, L. Saluzzi, A tree structure algorithm for optimal control problems with state constraints, Rendiconti di Matematica e delle sue Applicazioni 41 (2020) 193–221.
[22] K. Kunisch, S. Volkwein, L. Xie, HJB-POD based feedback design for the optimal control of evolution problems, SIAM J. on Applied Dynamical Systems 4 (2004) 701–722.
[23] K. Kunisch, L. Xie, POD-based feedback control of Burgers equation by solving the evolutionary HJB equation, Computers and Mathematics with Applications 49 (2005) 1113–1126.
[24] K. Kunisch, S. Volkwein, Optimal snapshot location for computing POD basis functions, ESAIM: Mathematical Modelling and Numerical Analysis 44 (3) (2010) 509–529.
[25] M. Hinze, S. Volkwein, Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: error estimates and suboptimal control, in: Dimension reduction of Large-Scale systems, Springer, 2005, pp. 261–306.
[26] T. Breiten, K. Kunisch, Feedback stabilization of the threedimensional Navier-Stokes equations using generalized Lyapunov equations, Discrete and Continuous Dynamical Systems 40 (7) (2020) 4197–4229.
[27] T. Breiten, K. Kunisch, L. Pfeiffer, Feedback stabilization of the two-dimensional Navier-Stokes equations by value function approximation, Applied Mathematics & Optimization 80 (3) (2019) 599–641.
[28] J. Garcke, A. Kröner, Suboptimal feedback control of PDEs by solving HJB equations on adaptive sparse grids, J Sci Comput 70 (1) (2016) 1–28.
[29] S. Dolgov, D. Kalise, K.K. Kunisch, Tensor decomposition methods for high-dimensional Hamilton–Jacobi–Bellman equations, SIAM Journal on Scientific Computing 43 (3) (2021) A1625–A1650.
[30] M. Oster, L. Sallandt, R. Schneider, Approximating optimal feedback controllers of finite horizon control problems using hierarchical tensor formats, SIAM Journal on Scientific Computing 44 (3) (2022) B746–B770.
[31] S. Dolgov, D. Kalise, L. Saluzzi, Data-driven tensor train gradient cross approximation for Hamilton-Jacobi-Bellman equations, arXiv preprint arXiv:2205.05109 (2022).
[32] S. Mowlavi, S. Nabi, Optimal control of PDEs using physics-informed neural networks (PINNs), in: APS Division of Fluid Dynamics Meeting Abstracts, 2021, pp. H23–005.
[33] N. Margenberg, D. Hartmann, C. Lessig, T. Richter, A neural network multigrid solver for the Navier-Stokes equations, J Comput Phys 460 (9999) (2022) 110983.
[34] R. Temam, Navier-Stokes Equations: Theory and Numerical Analysis, American Mathematical Society, USA, 2001.
[35] V. Barbu, R. Triggiani, Internal Stabilization of Navier-Stokes equations with finite dimensional controllers, Indiana University Mathematical Journal 53 (2004) 1443–1494.
[36] Y. Giga, S. Matsui, O. Sawada, Global existence of two-dimensional Navier-Stokes flow with nondecaying initial velocity, J. Math. Fluid Mech. 3 (3) (2001) 302–315.
[37] G. Kirsten, Order reduction of semilinear differential matrix and tensor equations, Alma Mater Studiorum Università di Bologna, 2021 Ph.D. thesis.
[38] V. Simoncini, Computational methods for linear matrix equations, SIAM Rev 58 (3) (2016) 377–441.
[39] D. Palitta, V. Simoncini, Matrix-equation-based strategies for convection–diffusion equations, BIT Numerical Mathematics 56 (2) (2016) 751–776.
[40] M.C. D'Autilia, I. Sgura, V. Simoncini, Matrix-oriented discretization methods for reaction–diffusion PDEs: comparisons and applications, Computers & Mathematics with Applications 79 (7) (2020) 2067–2085.
[41] S. Chaturantabut, D.C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM J. Sci. Comput. 32 (5) (2010) 2737–2764.
[42] G. Kirsten, Multilinear POD-DEIM model reduction for 2D and 3D semilinear systems of differential equations, Journal of Computational Dynamics 9 (2) (2022) 159–183.
[43] Z. Drmač, S. Gugercin, A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions, SIAM J. Sci. Comput. 38 (2) (2016) A631–A648.
[44] G. Strang, Computational science and engineering, Wellesley-Cambridge Press, 2007.
[45] B. Seibold, A compact and fast matlab code solving the incompressible navier-Stokes equations on rectangular domains, Massachusetts In-stitute of Technology, 2008. https://www.math.mit.edu/~gs/cse/codes/mit18086_navierstokes.pdf
[46] A.J. Chorin, Numerical solution of the Navier-Stokes equations, Math Comput 22 (104) (1968) 745–762.
[47] G. Kirsten, V. Simoncini, Order reduction methods for solving large-scale differential matrix Riccati equations, SIAM J. Sci. Comput. 42 (4) (2020) A2182–A2205.
[48] A. Quarteroni, Numerical models for differential problems, volume 2, Springer, 2009.
[49] A. Alla, M. Hinze, HJB-POD feedback control for Navier-Stokes equations, in: European Consortium for Mathematics in Industry, Springer, 2014, pp. 861–868.
[50] E. Bänsch, P. Benner, J. Saak, H.K. Weichelt, Riccati-based boundary feedback stabilization of incompressible navier-stokes flow, SIAM Journal on Scientific Computing 37 (2) (2015) A832–A858.
[51] L. Dieci, Numerical integration of the differential Riccati equation and some related issues, SIAM J Numer Anal 29 (3) (1992) 781–815.
[52] H. Mena, Numerical solution of differential Riccati equations arising in optimal control problems for parabolic partial differential equations, Ph. D. thesis, Escuela Politecnica Nacional (2007).