LLNL-JRNL-849457

A fast and accurate domain-decomposition nonlinear manifold reduced order model

A. N. Diaz, Y. Choi, M. Heinkenschloss

May 23, 2023

Computer Methods in Applied Mechanics and Engineering

A fast and accurate domain decomposition nonlinear manifold reduced order model

Alejandro N. Diaz[a,*], Youngsoo Choi[b], Matthias Heinkenschloss[a]

[a]*Department of Compuational Applied Mathematics and Operations Research, Rice University, Houston, 77005, TX, United States of America*
[b]*Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, 94550, CA, United States of America*

Abstract

This paper integrates nonlinear-manifold reduced order models (NM-ROMs) with domain decomposition (DD). NM-ROMs approximate the full order model (FOM) state in a nonlinear-manifold by training a shallow, sparse autoencoder using FOM snapshot data. These NM-ROMs can be advantageous over linear-subspace ROMs (LS-ROMs) for problems with slowly decaying Kolmogorov n-width. However, the number of NM-ROM parameters that need to be trained scales with the size of the FOM. Moreover, for "extreme-scale" problems, the storage of high-dimensional FOM snapshots alone can make ROM training expensive. To alleviate the training cost, this paper applies DD to the FOM, computes NM-ROMs on each subdomain, and couples them to obtain a global NM-ROM. This approach has several advantages: Subdomain NM-ROMs can be trained in parallel, involve fewer parameters to be trained than global NM-ROMs, require smaller subdomain FOM dimensional training data, and can be tailored to subdomain-specific features of the FOM. The shallow, sparse architecture of the autoencoder used in each subdomain NM-ROM allows application of hyper-reduction (HR), reducing the complexity caused by nonlinearity and yielding computational speedup of the NM-ROM. This paper provides the first application of NM-ROM (with HR) to a DD problem. In particular, this paper details an algebraic DD reformulation of the FOM, training a NM-ROM with HR for each subdomain, and a sequential quadratic programming (SQP) solver to evaluate the coupled global NM-ROM. Theoretical convergence results for the SQP method and *a priori* and *a posteriori* error estimates for the DD NM-ROM with HR are provided. The proposed DD NM-ROM with HR approach is numerically compared to a DD LS-ROM with HR on the 2D steady-state Burgers' equation, showing an order of magnitude improvement in accuracy of the proposed DD NM-ROM over the DD LS-ROM.

Keywords: Reduced order model, domain decomposition, nonlinear manifold, sparse autoencoders, neural networks, least-squares Petrov-Galerkin

1. Introduction

Many applications in science and engineering require the high-fidelity numerical simulation of a parameterized, large-scale, nonlinear system, referred to as the full order model (FOM). For example, in the design of the airfoil of an aircraft, one repeatedly simulates the airflow around the wing to compute the lift and drag for a number of shapes to determine the optimal shape. Alternatively, in the case of digital twins, one simulates the high-fidelity FOM in real-time for given system inputs. To guarantee a high-fidelity simulation, a high-dimensional numerical model is required, resulting in high computational expense when simulating the FOM. Consequently, both many-query and real-time applications are infeasible for large-scale problems. Model reduction alleviates the computational burden of simulating

[*]Corresponding author
Email addresses: `and5@rice.edu` (Alejandro N. Diaz), `choi15@llnl.gov` (Youngsoo Choi), `heinken@rice.edu` (Matthias Heinkenschloss)

the high-dimensional FOM by replacing it with a low-dimensional, computationally inexpensive model, referred to as a reduced order model (ROM), that approximates the dynamics of the FOM within a tunable accuracy. This ROM can then be used in place of the FOM in real-time and many-query applications. In this work, we integrate model reduction, specifically the nonlinear-manifold ROM (NM-ROM) approach, with an algebraic domain-decomposition (DD) framework.

There are a large number of works that consider the integration of DD with model reduction. One family of approaches is based on the reduced basis element (RBE) method [1, 2], in which reduced bases are computed locally for each subdomain. In the RBE method, continuity of the reduced basis solution across subdomains can be enforced via Lagrange multipliers as in [3], while others consider a discontinuous Galerkin approach [4]. Several modifications to the RBE method have been proposed, including the so-called static condensation RBE method [5, 6, 7], which computes a reduced basis (RB) approximation of the Schur complement and provides rigorous *a posteriori* error estimators. The reduced basis hybrid method (RBHM) [3] is another modification of the RBE method, in which a global coarse-grid solution is included in the reduced basis to ensure continuity of normal fluxes at subdomain interfaces. For RBHM, this continuity is enforced using Lagrange multipliers. Another well-studied approach uses the alternating Schwarz method, which decomposes the physical domain into two or more subdomains with or without overlap, and produces a global solution by iteratively solving the PDE on separate subdomains with boundary conditions coming from the state of neighboring subdomains at the previous iteration. The Schwarz method has been developed for both FOM-ROM and ROM-ROM couplings in, e.g., [8, 9], where the ROM is projection-based using Proper Orthogonal Decomposition (POD). The approach in [10] also considers FOM-ROM and ROM-ROM couplings, but couple subdomain solutions using Lagrange multipliers, and compute bases such that the Schur complement system required for recovering interface solutions is nonsingular. The authors in [11] also consider a Schwarz approach, but use an optimization-based coupling that minimizes the jump between PDE state solutions on the interface of neighboring subdomains. The authors in [12] compute component-based ROMs based on a partition-of-unity to couple local solutions. Others have considered using DD to compute ROMs for problems with spatially localized nonlinearities [13], and for use in design optimization [14, 15]. While these approaches have been successful, they are often problem-specific. That is, both RBE- and Schwarz-based methods typically formulate the DD problem at the PDE level and decompose the physical domain into separate subdomains. In contrast, the authors in [16] integrate DD and ROM for a general nonlinear FOM at the fully discrete level rather than the PDE level, and *algebraically* decompose the FOM rather than considering a decomposition of the physical domain. The authors then use POD to compute ROMs for each subdomain, and use an optimization-based coupling to minimize the discrete PDE residual while enforcing compatibility constraints at the interfaces. In this paper, we extend the DD ROM framework of [16] to incorporate the NM-ROM approach.

We integrate NM-ROM with DD to reduce the *offline* computational cost required for training an NM-ROM, and to allow NM-ROMs to scale with increasingly large FOMs. Indeed, in the monolithic single-domain case, training NM-ROMs is expensive due to the high-dimensionality of the FOM training data, which results in a large number of neural network (NN) parameters requiring training. By coupling NM-ROM with DD, one can compute FOM training data on subdomains, thus reducing the dimensionality of subdomain NM-ROM training data, resulting in fewer parameters that need to be trained per subdomain NM-ROM. Furthermore, the subdomain NM-ROMs can be trained in parallel and adapted to subdomain-specific features of the FOM. We also note that couplings of NNs and DD for solutions of partial differential equations (PDEs) have been considered in previous work (e.g., [17, 18, 19, 20]). However, these approaches use deep learning to solve a PDE by representing its solution as a NN and minimizing a corresponding physics-informed loss function. In contrast, our work uses autoencoders to reduce the dimensionality of an existing numerical model. The autoencoders are pretrained in an *offline* stage to find low-dimensional representations of FOM snapshot data, and used in an *online* stage to significantly reduce the computational cost and runtime of numerical simulations. Our work is the first to couple autoencoders with DD in the reduced-order modeling context.

A number of current model reduction approaches approximate the FOM solution in a low-dimensional linear subspace. In this paper, we collectively refer to this class of approaches as linear subspace ROM (LS-ROM). The LS-ROM approach supposes that the state solutions of the FOM are contained in a low-dimensional linear subspace. A basis for the linear subspace is then computed, resulting in a ROM whose state consists of the generalized coordinates of the approximate state solution in the reduced subspace. ROM approaches that follow LS-ROM include the reduced basis (RB) method [21, 22], proper orthogonal decomposition (POD) [23, 24, 25, 26, 27], balanced truncation and balanced POD [28, 29], interpolation and moment-matching based approaches [30, 31, 32], the Loewner framework [33, 34, 35], and the space–time POD [36, 37, 38] that expands the POD modes to temporal domain. Although LS-ROM approaches have been successful for a number of applications, it is well known that for advection-dominated problems and problems with sharp gradients, LS-ROM based approaches cannot produce low-dimensional subspaces where the state is well-approximated. More precisely, LS-ROM struggles when applied to problems with slowly decaying Kolmogorov n-width [39].

In recent years, a number of model reduction approaches have been developed to address the Kolmogorov n-width barrier. For example, one class of approaches leverages knowledge of the advection behavior of the given problem to enhance the approximation capabilities of linear subspaces. These approaches include composing transport maps with the reduced bases [40, 41, 42], shifting the POD basis [43], transforming the physical domain of the snapshots [44], and computing a reduced basis for a Lagrangian formulation of the PDE [45]. Other approaches consider the use of multiple linear subspaces, where instead of using a global reduced basis, one constructs multiple subspaces for separate regions in the time domain [25, 26, 46, 47], physical domain [48], or state space [49, 50]. However, each of these approaches relies upon a substantial amount of *a priori* knowledge of the governing PDE in order to improve the local approximation capabilities of linear subspaces. In contrast, another class of methods circumvents these drawbacks by approximating the FOM solution in a low-dimensional nonlinear manifold rather than a low-dimensional linear subspace. While LS-ROM approaches map the low-dimensional ROM state space to the high-dimensional FOM state space via an affine mapping, the approaches in [51, 52] consider the use of quadratic manifolds, where the ROM state space is mapped to the FOM state space via a quadratic mapping. As a further generalization of this mapping, researchers have investigated the use of neural networks to represent general nonlinear mappings from the ROM state space to the FOM state space. In particular, the use of autoencoders in the context of model reduction was first considered in the papers [53, 54]. Autoencoders are a type of neural network that aims to learn the identity mapping by first *encoding* the inputs to some latent representation via the *encoder*, then *decoding* the latent representation to the original input space via the *decoder*. In [55], the authors consider the use of deep convolutional autoencoders, which augment the autoencoder architecture with convolutional layers. While their approach was successful in addressing the Kolmogorov n-width issue, the computational speedup was limited because hyper-reduction (HR) was not incorporated into their framework to properly reduce the complexity caused by nonlinear terms. The authors in [56, 57] successfully apply HR in the context of NM-ROM and achieve a considerable speed-up, and do so by choosing a shallow, wide, and sparse architecture for the autoencoder. The approach in [58] also incorporates HR into an NM-ROM approach, but do so by employing a teacher-student training approach, where an autoencoder is first trained to reduce the entire state, and a second decoder is trained to only reproduce the HR nodes. This approach also permits the use of more general autoencoder architectures than the shallow, wide, and sparse architectures of [56, 57]. However, an advantage of the approach in [56, 57] is that autoencoder training only happens once rather than requiring a teacher-student training approach. Furthermore, this approach allows for different choices of HR nodes after NM-ROM training, whereas the approach in [58] requires fixed HR nodes.

In this paper, we extend the work of [16] on DD LS-ROM and integrate the NM-ROM approach with HR discussed in [56]. We incorporate the NM-ROM approach into this framework because of its success when applied to problems with slowly decaying Kolmogorov n-width. Specifically, to build ROMs on each subdomain of the DD problem, we apply NM-ROM with HR by using wide, shallow, sparse-masked autoencoders. The wide, shallow, and sparse architecture allows for hyper-reduction to be efficiently applied, thus reducing the complexity caused by nonlinearity and yielding computational speedup. Additionally, we modify the wide, shallow, and sparse architecture used in [56] to also include a sparsity mask for the encoder input layer as well as the decoder output layer. The sparsity mask at the encoder input layer results in an architecture that is symmetric across the latent layer of the autoencoder. Using *sparse* linear layers also allows one to make the encoders and decoders very wide while keeping memory costs low. Integrating NM-ROM with DD allows one to compute the FOM training snapshots on subdomains, thus significantly reducing the number of NN parameters requiring training for each subdomain.

A summary of the key contributions from this paper are as follows.

- We develop the first application of NM-ROM with HR to a DD problem.
- We modify the autoencoder architecture discussed in [56] to also include sparsity in the encoder input layer as well as the decoder output layer.
- We develop an inexact Lagrange-Newton sequential quadratic programming (SQP) method for the DD NM-ROM, and provide a theoretical convergence result for the SQP solver.
- We provide *a priori* and *a posteriori* error estimates for the DD ROM which are valid for both LS-ROM and NM-ROM.
- We numerically compare DD LS-ROM with DD NM-ROM, both with and without HR, for a number of different problem configurations using the 2D steady-state Burgers' equation.

This paper is structured as follows. Section 2 discusses the algebraic DD FOM formulation that we consider. Section 3 discusses the constrained least-squares Petrov-Galerkin (LSPG) formulation for the ROM, which respects the DD FOM formulation. We then review the LS-ROM approach based on POD in Section 3.3, and detail the NM-ROM approach in Section 3.4. We develop an inexact Lagrange-Newton sequential quadratic programming (SQP) method for the constrained LSPG-ROM in Section 4, followed by standard theoretical convergence results for the SQP solver in Section 4.2. We then discuss the autoencoder architecture used in Section 5, the application of hyper-reduction in Section 5.3, and the construction of a HR subnet in Section 5.4. In Section 6, we provide both *a posteriori* and *a priori* error bounds for the ROM solution in Theorems 5 and 6, respectively. We numerically compare the DD LS-ROM and DD NM-ROM performance, both with and without HR, on the 2D steady-state Burgers' equation in Section 7 for a number of different problem configurations. Lastly, we conclude the paper and discuss future directions in Section 8.

2. Domain-decomposition FOM formulation

This section presents the algebraic domain-decomposition formulation [16]. We consider a FOM parameterized by $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^{N_\mu}$. Given $\boldsymbol{\mu} \in \mathcal{D}$, the FOM is expressed as a parametrized system of nonlinear algebraic equations

$$\boldsymbol{r}(\boldsymbol{x}(\boldsymbol{\mu}); \boldsymbol{\mu}) = \mathbf{0}, \tag{1}$$

where $\boldsymbol{r} : \mathbb{R}^{N_x} \times \mathcal{D} \to \mathbb{R}^{N_x}$ denotes the residual and $\boldsymbol{x}(\boldsymbol{\mu}) \in \mathbb{R}^{N_x}$ denotes the state. For notational simplicity, the dependence on $\boldsymbol{\mu}$ is suppressed until needed. Typically $\boldsymbol{r}$ corresponds to a discretized PDE (e.g., using finite differences or finite elements) and in our target applications $\boldsymbol{r}$ is nonlinear in $\boldsymbol{x}$.

Next we decompose the system (1) into n_Ω *algebraic subdomains*. Before giving the technical details, we describe the decomposition using a simple example illustrated in Figure 1. In this example, suppose the system (1) is obtained from a finite difference discretization with a 5-point stencil of a scalar PDE in a rectangular domain in $\mathbb{R}^2$ with Dirichlet boundary conditions. The situation would be similar if the PDE was discretized using linear finite elements on a regular grid. Moreover, the decomposition into algebraic subdomains is not limited to the finite difference discretization with a 5-point stencil; this discretization is simply used for illustration. In the left plot in Figure 1, the finite difference discretization uses a 14×5 grid of $N_x = 70$ nodes. Each node corresponds to a component in the vector of unknown states $\boldsymbol{x}$ and an equation in the system (1). The domain is subdivided into $n_\Omega = 2$ subdomains. Nodes near the interface between the two subdomains are marked by filled circles. Because the PDE is discretized using a 5-point stencil, residuals corresponding to nodes marked by filled circles in subdomain 1 depend on state variables corresponding to nodes marked by filled circles in subdomain 2. Similarly, residuals corresponding to nodes marked by filled circles in subdomain 2 depend on state variables corresponding to nodes marked by filled circles in subdomain 1. In the right plot in Figure 1, these variables are duplicated as components of the vectors of the *interface states* $\boldsymbol{x}_1^\Gamma$ and $\boldsymbol{x}_2^\Gamma$. These have to satisfy $\boldsymbol{x}_1^\Gamma = \boldsymbol{x}_2^\Gamma$ and this compatibility condition will later be enforced via constraints. The other state variables are the *interior states* $\boldsymbol{x}_i^\Omega$, $i = 1, 2$. These are the state variables that only enter the residuals corresponding to subdomain i. Next we provide a detailed description of the general case.

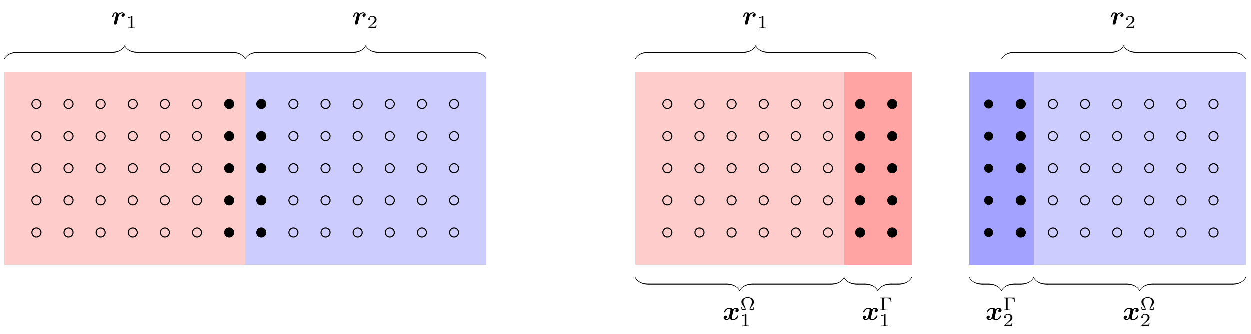

$\boldsymbol{r}_1$
$\boldsymbol{r}_2$
$\boldsymbol{r}_1$
$\boldsymbol{r}_2$
$\boldsymbol{x}_1^\Omega$
$\boldsymbol{x}_1^\Gamma$
$\boldsymbol{x}_2^\Gamma$
$\boldsymbol{x}_2^\Omega$

Figure 1: Left plot: Each node in the domain corresponds to an unknown $\boldsymbol{x}$ and an equation in the system (1). The domain is subdivided into $n_\Omega = 2$ subdomains. Residuals corresponding to nodes marked by filled circles near the bboundary in subdomain 1 depend on nodes marked by filled circles in subdomain 2, and residuals corresponding to nodes marked by filled circles near the boundary in subdomain 2 depend on nodes marked by filled circles in subdomain 1. Right plot: Variables that enter computations of residuals in one or more subdomains are duplicated as interface state variables $\boldsymbol{x}_1^\Gamma$ and $\boldsymbol{x}_2^\Gamma$. Variables that only enter the computations of the residuals in one subdomains are the interior state variables $\boldsymbol{x}_1^\Omega$ and $\boldsymbol{x}_2^\Omega$, respectively. Equality $\boldsymbol{x}_1^\Gamma = \boldsymbol{x}_2^\Gamma$ of interface state variables will be enforced via constraints.

We decompose the system (1) into $n_\Omega \le N_x$ *algebraic subdomains* by defining so-called residual sampling matrices $\boldsymbol{P}_i^r \in \{0,1\}^{N_i^r \times N_x}$ and computing subdomain residuals as

$$\boldsymbol{P}_i^r \boldsymbol{r}(\boldsymbol{x}) \in \mathbb{R}^{N_i^r}, \quad i = 1, \ldots, n_\Omega.$$

The residual sampling matrices are assumed to be *algebraically non-overlapping*, i.e.

$$\boldsymbol{P}_i^r (\boldsymbol{P}_j^r)^T = \boldsymbol{0}, \quad \forall\, i \neq j,$$

and $\sum_{i=1}^{n_\Omega} N_i^r = N_x$. For problems (1) arising from a PDE discretization, the sparsity structure of the monolithic residual function $\boldsymbol{r}$ implies that subdomain residuals $\boldsymbol{P}_i^r \boldsymbol{r}(\boldsymbol{x})$ only depend on a subset of the full state $\boldsymbol{x}$. Furthermore, the residual corresponding to points at the boundary of subdomain i depend on the state $\boldsymbol{x}$ at points within subdomain i and at points that belong to neighboring subdomains. Therefore, for subdomain i, we decompose the state components into *interior states*

$$\boldsymbol{x}_i^\Omega := \boldsymbol{P}_i^\Omega \boldsymbol{x} \in \mathbb{R}^{N_i^\Omega} \tag{2a}$$

and *interface states*

$$\boldsymbol{x}_i^\Gamma := \boldsymbol{P}_i^\Gamma \boldsymbol{x} \in \mathbb{R}^{N_i^\Gamma}, \tag{2b}$$

where $\boldsymbol{P}_i^\Omega \in \{0,1\}^{N_i^\Omega \times N_x}$ denotes the ith interior-state sampling matrix and $\boldsymbol{P}_i^\Gamma \in \{0,1\}^{N_i^\Gamma \times N_x}$ denotes the ith interface-state sampling matrix. The interior states $\boldsymbol{x}_i^\Omega := \boldsymbol{P}_i^\Omega \boldsymbol{x}$ only enter the evaluation of the ith subdomain residual $\boldsymbol{P}_j^r \boldsymbol{r}(\boldsymbol{x})$. The interface states $\boldsymbol{x}_i^\Gamma := \boldsymbol{P}_i^\Gamma \boldsymbol{x}$ also enter the evaluation of another subdomain residual $\boldsymbol{P}_j^r \boldsymbol{r}(\boldsymbol{x})$, $j \neq i$. Since the ith interior states only enter the evaluation of the ith subdomain, the interior-state sampling matrices are algebraically non-overlapping,

$$\boldsymbol{P}_i^\Omega (\boldsymbol{P}_j^\Omega)^T = \boldsymbol{0}, \quad \forall\, i \neq j.$$

The interface state variables are duplicated across one or more subdomains, and we will describe later how to enforce equality among duplicated interface state variables.

With these specifications we can now define subdomain residual functions $\boldsymbol{r}_i : \mathbb{R}^{N_i^\Omega} \times \mathbb{R}^{N_i^\Gamma} \to \mathbb{R}^{N_i^r}$ as

$$\boldsymbol{r}_i(\boldsymbol{x}_i^\Omega, \boldsymbol{x}_i^\Gamma) = \boldsymbol{P}_i^r \boldsymbol{r}\Big(\big(\boldsymbol{P}_i^\Omega\big)^T \boldsymbol{x}_i^\Omega + \big(\boldsymbol{P}_i^\Gamma\big)^T \boldsymbol{x}_i^\Gamma\Big). \tag{3}$$

Furthermore, the monolithic residual function (1) can be decomposed as

$$\boldsymbol{r}(\boldsymbol{x}) = \sum_{i=1}^{n_\Omega} \left(\boldsymbol{P}_i^r\right)^T \boldsymbol{r}_i(\boldsymbol{P}_i^\Omega \boldsymbol{x}, \boldsymbol{P}_i^\Gamma \boldsymbol{x}), \qquad \forall\, \boldsymbol{x} \in \mathbb{R}^{N_x}. \tag{4}$$

Equations (1), (3), and (4) imply that the solution (2) of (1) restricted to the ith subdomain satisfies

$$\boldsymbol{r}_i(\boldsymbol{x}_i^\Omega, \boldsymbol{x}_i^\Gamma) = \boldsymbol{0}, \quad i = 1, \ldots, n_\Omega. \tag{5}$$

In addition to (5), compatibility conditions must be imposed that enforce equality between overlapping interface states for neighboring subdomains. These compatibility conditions are enforced by defining n_p non-overlapping *ports*. Geometrically, the jth port is a subset of subdomains. The jth port has $N_j^p \leq N_x$ overlapping interface-state variables. The indices of subdomains that intersect with the jth port are $P(j) \subseteq \{1, \ldots, n_\Omega\}$. Figure 2 displays the ports for a 4-subdomain example configuration.

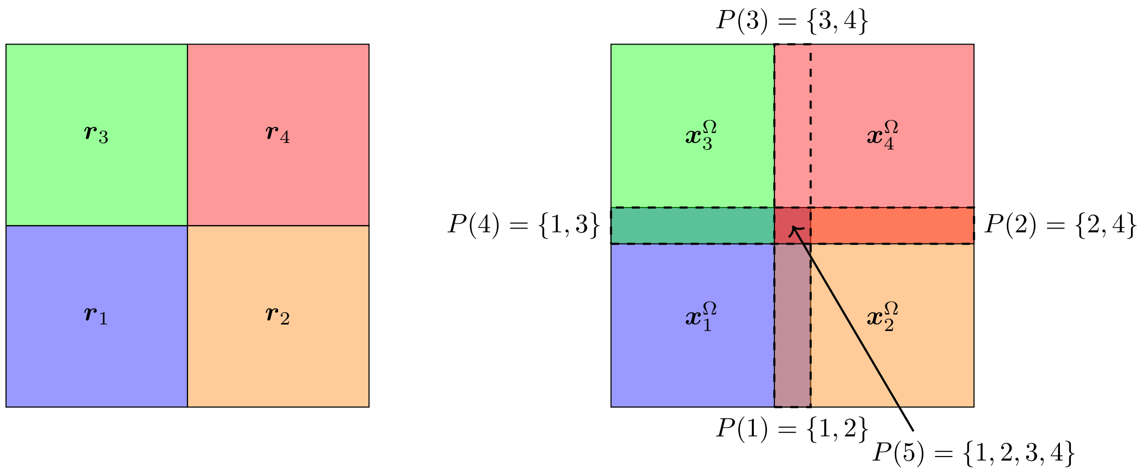


Figure 2: Left: Residual decomposition using 4 subdomains. Notice that the residuals do not overlap. Right: State decomposition. The regions without overlap correspond to interior states $\boldsymbol{x}_i^\Omega$ while regions with overlap correspond to interface states $\boldsymbol{x}_i^\Gamma$. The overlapping regions enclosed by black dashed lines represent the ports $P(j) \subset \{1, \ldots, n_\Omega\}$.

Using the ports, the compatibility conditions can be expressed as

$$\boldsymbol{P}_i^j \boldsymbol{x}_i^\Gamma = \boldsymbol{P}_\ell^j \boldsymbol{x}_\ell^\Gamma, \quad i, \ell \in P(j),\ j = 1, \ldots, n_p, \tag{6}$$

where $\boldsymbol{P}_i^j \in \{0,1\}^{N_j^p \times N_i^\Gamma}$ denotes the jth port sampling matrix for subdomain i. Because the ports are non-overlapping, if $Q(i) := \{j \mid i \in P(j)\}$ is the set of ports associated with subdomain i, then

$$\boldsymbol{P}_i^j (\boldsymbol{P}_i^\ell)^T = \boldsymbol{0}, \qquad \forall\, j, \ell \in Q(i),\ j \neq \ell, \tag{7}$$

and the sum of numbers of variables in the ports associated with subdomain i is equal to the number of interface variables in the ith subdomain, $\sum_{j \in Q(i)} N_j^p = N_i^\Gamma$.

As written, most conditions in (6) are redundant. Instead, for port j one needs $(|P(j)|-1)N_j^p$ conditions, where $|P(j)|$ denotes cardinality of $P(j)$. For example, in Figure 2 one needs the conditions $\boldsymbol{P}_1^1\boldsymbol{x}_1^\Gamma = \boldsymbol{P}_2^1\boldsymbol{x}_2^\Gamma$ for the first port $P(1)=\{1,2\}$, and the conditions $\boldsymbol{P}_1^5\boldsymbol{x}_1^\Gamma = \boldsymbol{P}_2^5\boldsymbol{x}_2^\Gamma$, $\boldsymbol{P}_2^5\boldsymbol{x}_2^\Gamma = \boldsymbol{P}_3^5\boldsymbol{x}_3^\Gamma$, $\boldsymbol{P}_3^5\boldsymbol{x}_3^\Gamma = \boldsymbol{P}_4^5\boldsymbol{x}_4^\Gamma$ for the fifth port $P(5)=\{1,2,3,4\}$. Removing redundant conditions (6), the port compatibility conditions (6) can be written as

$$\sum_{i=1}^{n_\Omega} \boldsymbol{A}_i\boldsymbol{x}_i^\Gamma = \boldsymbol{0}, \tag{8}$$

where the $\boldsymbol{A}_i \in \{-1,0,1\}^{N_A\times N_i^\Gamma}$ denote the constraint matrices associated with the port compatibility conditions (6) and the total number of compatibility conditions is $N_A = \sum_{j=1}^{n_p}(|P(j)|-1)N_j^p$. The matrix $(\boldsymbol{A}_1,\ldots,\boldsymbol{A}_{n_\Omega})$ has full row rank.

In summary, the algebraic DD formulation of the FOM (1) is given by

$$\boldsymbol{r}_i(\boldsymbol{x}_i^\Omega,\boldsymbol{x}_i^\Gamma) = \boldsymbol{0}, \qquad i=1,\ldots,n_\Omega, \tag{9a}$$

$$\sum_{i=1}^{n_\Omega} \boldsymbol{A}_i\boldsymbol{x}_i^\Gamma = \boldsymbol{0}. \tag{9b}$$

Associated with (9) we also consider the nonlinear least-squares problem with equality constraints,

$$\min_{(\boldsymbol{x}_i^\Omega,\boldsymbol{x}_i^\Gamma),i=1,\ldots,n_\Omega} \quad \frac{1}{2}\sum_{i=1}^{n_\Omega}\left\|\boldsymbol{r}_i\left(\boldsymbol{x}_i^\Omega,\boldsymbol{x}_i^\Gamma\right)\right\|_2^2 \tag{10a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_\Omega} \boldsymbol{A}_i\boldsymbol{x}_i^\Gamma = \boldsymbol{0}. \tag{10b}$$

The connections between the formulations (1), (9), and (10) are summarized in the following theorem.

Theorem 1. *If $\boldsymbol{x}$ solves the FOM* (1) *then $(\boldsymbol{x}_i^\Omega,\boldsymbol{x}_i^\Gamma)$ with $\boldsymbol{x}_i^\Omega := \boldsymbol{P}_i^\Omega\boldsymbol{x}$ and $\boldsymbol{x}_i^\Gamma := \boldsymbol{P}_i^\Gamma\boldsymbol{x}$, $i=1,\ldots,n_\Omega$, solves the algebraic DD formulation of the FOM* (9) *and vice versa. A solution of* (9) *also solves* (10)*, and a solution of* (10) *with objective function value equal to zero solves* (9)*.*

The proof of Theorem 1 follows immediately from the construction of (9).

In the FOM context, the constrained nonlinear least-squares problem formulation (10) is not needed, but we include it here because it will become important for the model reduction derivation in Section 3, where we use a least squares formulation for the subdomain ROMs. The constrained nonlinear least-squares formulation (10) of the FOM could be solved using a Lagrange-Newton sequential quadratic programming (SQP) method with Gauss-Newton Hessian approximation and the constrained nonlinear least-squares formulation of the NM-ROM corresponding to (10) will be solved using the Lagrange-Newton SQP method discussed in Section 4.

In principle, an alternative DD formulation of the FOM is possible, which reverses the role of satisfying the subdomain equations and of the compatibility conditions. Instead of (9), one can impose the subdomain equations $\boldsymbol{r}_i(\boldsymbol{x}_i^\Omega,\boldsymbol{x}_i^\Gamma)=\boldsymbol{0}$, $i=1,\ldots,n_\Omega$, as constraints and use a least squares formulation of the compatibility conditions as the objective. This is used, e.g., in [11, 59]. However, since we use LSPG-ROMs, which in general do not have zero residual, these cannot be incorporated as equality constraints. In contrast, the formulation (10) can be used with subdomain LSPG-ROMs, as we will describe in the next section.

3. Domain-decomposition ROM

The ROM construction is built on the assumption that the high-dimensional subdomain state variables $\boldsymbol{x}_i^\Omega \in \mathbb{R}^{N_i^\Omega}$ and $\boldsymbol{x}_i^\Gamma \in \mathbb{R}^{N_i^\Gamma}$, $i=1,\ldots,n_\Omega$, can be approximated using low-dimensional variables $\widehat{\boldsymbol{x}}_i^\Omega \in \mathbb{R}^{n_i^\Omega}$, $n_i^\Omega \ll N_i^\Omega$ and

$\widehat{\boldsymbol{x}}_i^\Gamma \in \mathbb{R}^{N_i^\Gamma}$, $n_i^\Gamma \ll N_i^\Gamma$, $i = 1, \dots, n_\Omega$, respectively. Specifically, we assume that for each subdomain i there exist maps $\boldsymbol{g}_i^\Omega : \mathbb{R}^{n_i^\Omega} \to \mathbb{R}^{N_i^\Omega}$ and $\boldsymbol{g}_i^\Gamma : \mathbb{R}^{n_i^\Gamma} \to \mathbb{R}^{N_i^\Gamma}$ such that

$$\boldsymbol{x}_i^\Omega \approx \boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \qquad \boldsymbol{x}_i^\Gamma \approx \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma), \qquad i = 1, \dots, n_\Omega. \tag{11}$$

In the traditional LS-ROM the maps $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$ are linear, whereas in our NM-ROM these maps are computed via autoencoders/decoders. Assuming that we have maps $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$ such that (11) holds, we discuss how to construct the ROM in Section 3.1. Specifically, our ROM is based on the constrained nonlinear least-squares formulation (10) of the FOM. One issue in the ROM construction based on (10) is the formulation of compatibility constraints for the ROM. In Section 3.1 we use a formulation following [16] and in Section 3.2 we provide an alternative formulation of the ROM compatibility constraints by constructing the maps $\boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)$ in a suitable way. The detailed construction of maps $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$ such that (11) holds is discussed in Sections 3.3, 3.4, and 5. Specifically, we will review the traditional LS-ROM in Section 3.3. Sections 3.4 and 5 discuss how to compute these maps via the NM-ROM approach.

3.1. Least-squares formulation

Given maps $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$ such that (11) holds, a naive way of computing the ROM is to simply replace $\boldsymbol{x}_i^\Omega$ and $\boldsymbol{x}_i^\Gamma$ in the constrained nonlinear least-squares formulation (10) of the FOM by $\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega)$ and $\boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)$. An evaluation of this ROM requires the solution of

$$\min_{(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma), i=1,\dots,n_\Omega} \quad \frac{1}{2} \sum_{i=1}^{n_\Omega} \left\| \boldsymbol{r}_i \left(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma) \right) \right\|_2^2 \tag{12a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_\Omega} \boldsymbol{A}_i \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma) = \boldsymbol{0}. \tag{12b}$$

This corresponds to a (naive) LSPG-ROM. There are two issues with this formulation.

The first issue is that, just as in the case of the classical LSPG-ROM, the complexity of the evaluation of the subdomain residuals, i.e., $\big(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma\big) \to \big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big) \to \boldsymbol{r}_i\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big)$ scales with the size N_i^Ω and N_i^Γ of the FOM. This issue is addressed using so-called hyper-reduction (HR). See, e.g., [60] for an overview. HR replaces the residual $\boldsymbol{r}_i\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big)$ in (12a) by $\boldsymbol{B}_i \boldsymbol{r}_i\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big)$, where $\boldsymbol{B}_i \in \mathbb{R}^{N_i^B \times N_i^r}$, $N_i^B \le N_i^r$, is determined by the HR approach. For example, $\boldsymbol{B}_i = \boldsymbol{I}$ corresponds to vanilla LSPG, $\boldsymbol{B}_i = \boldsymbol{Z}_i$, where $\boldsymbol{Z}_i \in \{0,1\}^{N_i^z \times N_i^r}$, $N_i^z < N_i^r$, denotes a row-sampling matrix, corresponds to collocation HR, and $\boldsymbol{B}_i = (\boldsymbol{Z}_i \boldsymbol{\Phi}_i^r)^\dagger \boldsymbol{Z}_i$, where $\boldsymbol{Z}_i$ is as before, $\boldsymbol{\Phi}_i^r \in \mathbb{R}^{N_i^r \times n_i^r}$, $i = 1, \dots, n_\Omega$, denotes a reduced subspace for the corresponding subdomain residual and the superscript $\dagger$ denotes the Moore-Penrose pseudoinverse, corresponds to gappy POD HR [61, 62, 63]. Further details on HR for our DD NM-ROM are discussed in Section 5.3. For the application of HR to DD LS-ROM, we refer the reader to [16].

The second issue with (12) is that it involves the same number of constraints (12b) as the FOM (10), but fewer degrees of freedom to satisfy them. In the extreme case, it may be impossible to satisfy the constraints (12b). One approach, following [16], is to replace $\boldsymbol{A}_i$ in (12b) by $\boldsymbol{C}\boldsymbol{A}_i$, where $\boldsymbol{C} \in \mathbb{R}^{n_C \times N_A}$, $n_C \ll N_A$, is a test matrix that converts (12b) into a so-called "weak compatibility constraint". We will choose $\boldsymbol{C}$ to be a Gaussian matrix, but in principle other choices of $\boldsymbol{C}$ can be used.

To summarize, given maps $\boldsymbol{g}_i^\Omega : \mathbb{R}^{n_i^\Omega} \to \mathbb{R}^{N_i^\Omega}$ and $\boldsymbol{g}_i^\Gamma : \mathbb{R}^{n_i^\Gamma} \to \mathbb{R}^{N_i^\Gamma}$ such that (11) holds, given HR matrices $\boldsymbol{B}_i \in \mathbb{R}^{N_i^B \times N_i^r}$, $N_i^B \le N_i^r$, $i = 1, \dots, n_\Omega$, and given $\boldsymbol{C} \in \mathbb{R}^{n_C \times N_A}$, $n_C \ll N_A$, our DD-LSPG-ROM is evaluated by solving

$$\min_{(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma), i=1,\dots,n_\Omega} \quad \frac{1}{2} \sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i \left(\boldsymbol{g}_i^\Omega\left(\widehat{\boldsymbol{x}}_i^\Omega\right), \boldsymbol{g}_i^\Gamma\left(\widehat{\boldsymbol{x}}_i^\Gamma\right) \right) \right\|_2^2 \tag{13a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_\Omega} \boldsymbol{C}\boldsymbol{A}_i \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma) = \boldsymbol{0}. \tag{13b}$$

The DD-LSPG-ROM formulation (13) will be referred to as the weak FOM-port constraint (WFPC) formulation.

While the FOM (9) or (10) has linear constraints, the WFPC formulation has nonlinear constraints in general. Corresponding to $\boldsymbol{g}_i^\Gamma : \mathbb{R}^{n_i^\Gamma} \to \mathbb{R}^{N_i^\Gamma}$ is a function $\boldsymbol{h}_i^\Gamma : \mathbb{R}^{N_i^\Gamma} \to \mathbb{R}^{n_i^\Gamma}$ such that $\|\boldsymbol{g}_i^\Gamma(\boldsymbol{h}_i^\Gamma(\boldsymbol{x}_i^{\Gamma,\mathrm{train}})) - \boldsymbol{x}_i^{\Gamma,\mathrm{train}}\|$ is small for some training/snapshot data $\boldsymbol{x}_i^{\Gamma,\mathrm{train}}$, $i = 1, \ldots, n_\Omega$, that satisfy the linear FOM constraints (9b). See Sections 3.3 and 3.4. Thus $\sum_{i=1}^{n_\Omega} \boldsymbol{C}\boldsymbol{A}_i\boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)$ is guaranteed to be small at these training/snapshot data. Existence of points that satisfy (13b) in the nonlinear case is still an open issue. However, in our numerical examples we have not observed any issues related to existence of feasible points for (13). The existence of solutions of (13) can be guaranteed under mild conditions that are typical for optimization problems.

Theorem 2. *Let $\widetilde{\boldsymbol{x}}_i^\Gamma$, $i = 1, \ldots, n_\Omega$, satisfy the constraints (13b) and let $\widetilde{\boldsymbol{x}}_i^\Omega$, $i = 1, \ldots, n_\Omega$, be arbitrary. If the residual function $\boldsymbol{r}$ and the maps $\boldsymbol{g}_i^\Omega$, $\boldsymbol{g}_i^\Gamma$, $i = 1, \ldots, n_\Omega$, are continuous and if the level set*

$$L = \Big\{ (\widehat{\boldsymbol{x}}_1^\Omega, \widehat{\boldsymbol{x}}_1^\Gamma, \ldots, \widehat{\boldsymbol{x}}_{n_\Omega}^\Omega, \widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma) \;:\; \sum_{i=1}^{n_\Omega} \boldsymbol{C}\boldsymbol{A}_i\boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma) = \boldsymbol{0}, \\ \sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i\boldsymbol{r}_i\left(\boldsymbol{g}_i^\Omega\left(\widehat{\boldsymbol{x}}_i^\Omega\right), \boldsymbol{g}_i^\Gamma\left(\widehat{\boldsymbol{x}}_i^\Gamma\right)\right)\right\|_2^2 \le \sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i\boldsymbol{r}_i\left(\boldsymbol{g}_i^\Omega\left(\widetilde{\boldsymbol{x}}_i^\Omega\right), \boldsymbol{g}_i^\Gamma\left(\widetilde{\boldsymbol{x}}_i^\Gamma\right)\right)\right\|_2^2 \Big\}$$

is bounded, then (13) *has a solution.*

Proof. If $(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma)$, $i = 1, \ldots, n_\Omega$, solves (13), then it also solves the minimization problem with the constraint $(\widehat{\boldsymbol{x}}_1^\Omega, \widehat{\boldsymbol{x}}_1^\Gamma, \ldots, \widehat{\boldsymbol{x}}_{n_\Omega}^\Omega, \widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma) \in L$ added. The feasible set of this new minimization problem is compact, the objective function is continuous, and therefore this minimization problem has a solution, which is also a solution of (13). □

Instead of the weak compatibility constraint (13b) one can also construct the maps $\boldsymbol{g}_i^\Gamma$, $i = 1, \ldots, n_\Omega$, such that compatibility is enforced strongly for appropriate components of $\boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)$, $i = 1, \ldots, n_\Omega$. This approach is introduced in the following Section 3.2.

3.2. Strong ROM-port constraints

In the general formulation, the maps (11) are computed separately for each subdomain i. However, since the interface variables $\boldsymbol{x}_i^\Gamma$, $\boldsymbol{x}_\ell^\Gamma$ are identical on each port j associated with the subdomains i, ℓ, i.e., on each port j with $i, \ell \in P(j)$ (see (6)), one can instead reduce the interface variables on each port and then combine the reduced port interface variables to a reduced interface variable. By (6) the interface variables for port j must satisfy

$$\boldsymbol{x}_j^p = \boldsymbol{P}_i^j \boldsymbol{x}_i^\Gamma \in \mathbb{R}^{N_j^p}, \qquad \forall\, i \in P(j). \tag{14}$$

For each port j we reduce the FOM port variables $\boldsymbol{x}_j^p$, i.e., for each port j we compute a single map $\boldsymbol{g}_j^p : \mathbb{R}^{n_j^p} \to \mathbb{R}^{N_j^p}$, where

$$\boldsymbol{g}_j^p\left(\widehat{\boldsymbol{x}}_j^p\right) \approx \boldsymbol{x}_j^p = \boldsymbol{P}_i^j \boldsymbol{x}_i^\Gamma, \qquad \forall\, i \in P(j). \tag{15}$$

The reduced interface variable $\widehat{\boldsymbol{x}}_i^\Gamma$ is now computed by concatenating all port variables $\widehat{\boldsymbol{x}}_j^p$ with $i \in P(j)$. This leads to the ROM port sampling matrices $\widehat{\boldsymbol{P}}_i^j \in \{0,1\}^{n_j^p \times n_i^\Gamma}$ which are defined through

$$\widehat{\boldsymbol{x}}_j^p = \widehat{\boldsymbol{P}}_i^j \widehat{\boldsymbol{x}}_i^\Gamma, \qquad i \in P(j). \tag{16}$$

Equation (16) implies that on the jth port we have

$$\widehat{\boldsymbol{P}}_i^j \widehat{\boldsymbol{x}}_i^\Gamma = \widehat{\boldsymbol{P}}_\ell^j \widehat{\boldsymbol{x}}_\ell^\Gamma, \qquad i, \ell \in P(j). \tag{17}$$

To introduce parallelism, ROM interface variables $\widehat{\boldsymbol{x}}_i^\Gamma$ are introduced for each subdomain, and are coupled by enforcing (17). By construction, the ROM ports are non-overlapping, i.e.

$$\widehat{\boldsymbol{P}}_i^j \left(\widehat{\boldsymbol{P}}_i^\ell\right)^T = \boldsymbol{0}, \qquad \forall\, j, \ell \in Q(i),\ j \neq \ell, \tag{18}$$

and $n_i^\Gamma = \sum_{j \in Q(i)} n_j^p$. As discussed in Section 2, after removing redundant conditions in (17), one can write the ROM port compatibility conditions (17) as

$$\sum_{i=1}^{n_\Omega} \widehat{\boldsymbol{A}}_i \widehat{\boldsymbol{x}}_i^\Gamma = \boldsymbol{0}, \tag{19}$$

where $\widehat{\boldsymbol{A}}_i \in \{-1,0,1\}^{n_A \times n_i^\Gamma}$, $\ n_A = \sum_{j=1}^{n_p} (|P(j)| - 1) n_j^p$, denote the constraint matrices associated with port compatibility conditions (17). The matrix

$$(\widehat{\boldsymbol{A}}_1, \ldots, \widehat{\boldsymbol{A}}_{n\Omega}) \in \mathbb{R}^{n_A \times \sum_{i=1}^{n_\Omega} n_i^\Gamma} \tag{20}$$

has full row rank n_A, and $n_A < \sum_{i=1}^{n_\Omega} n_i^\Gamma$.

The map $\boldsymbol{g}_i^\Gamma : \mathbb{R}^{n_i^\Gamma} \to \mathbb{R}^{N_i^\Gamma}$ that approximates the interface state $\boldsymbol{x}_i^\Gamma$ is implied by the port maps $\boldsymbol{g}_j^p$. To see this, note that the FOM compatibility conditions (6) and the non-overlapping condition (7) allow one to rewrite $\boldsymbol{x}_i^\Gamma$ as

$$\boldsymbol{x}_i^\Gamma = \sum_{j \in Q(i)} (\boldsymbol{P}_i^j)^T \boldsymbol{P}_i^j \boldsymbol{x}_i^\Gamma. \tag{21}$$

Thus, using (15), (16), and (21) the map $\boldsymbol{g}_i^\Gamma : \mathbb{R}^{n_i^\Gamma} \to \mathbb{R}^{N_i^\Gamma}$ that approximates the interface state $\boldsymbol{x}_i^\Gamma$ is given by

$$\boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma) := \sum_{j \in Q(i)} (\boldsymbol{P}_i^j)^T \boldsymbol{g}_j^p \left(\widehat{\boldsymbol{P}}_i^j \widehat{\boldsymbol{x}}_i^\Gamma\right). \tag{22}$$

In particular, the definition (22) of $\boldsymbol{g}_i^\Gamma$ and the ROM compatibility conditions (17) imply that

$$\boldsymbol{P}_i^j \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma) = \boldsymbol{g}_j^p \left(\widehat{\boldsymbol{P}}_i^j \widehat{\boldsymbol{x}}_i^\Gamma\right) = \boldsymbol{g}_j^p \left(\widehat{\boldsymbol{P}}_\ell^j \widehat{\boldsymbol{x}}_\ell^\Gamma\right) = \boldsymbol{P}_\ell^j \boldsymbol{g}_\ell^\Gamma(\widehat{\boldsymbol{x}}_\ell^\Gamma)$$

for all $i, \ell \in P(j)$ and for all ports $P(j)$. This implies that strong compatibility holds for the FOM ports:

$$\sum_{i=1}^{n_\Omega} \boldsymbol{A}_i \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma) = \boldsymbol{0}.$$

In summary, if port maps $\boldsymbol{g}_j^p$ are constructed such that (15) holds and the implied interface maps $\boldsymbol{g}_i^\Gamma$ are (22), then the DD-LSPG-ROM is evaluated by solving

$$\min_{(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma), i=1,\ldots,n_\Omega} \quad \frac{1}{2} \sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i \left(\boldsymbol{g}_i^\Omega \left(\widehat{\boldsymbol{x}}_i^\Omega\right), \boldsymbol{g}_i^\Gamma \left(\widehat{\boldsymbol{x}}_i^\Gamma\right)\right) \right\|_2^2 \tag{23a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_\Omega} \widehat{\boldsymbol{A}}_i \widehat{\boldsymbol{x}}_i^\Gamma = \boldsymbol{0}. \tag{23b}$$

The formulation (23) will be referred to as the strong ROM-port constraint (SRPC) formulation.

In contrast to (13) the constraints in (23) are linear and the set of feasible points for (23) is the null-space of the constraint matrix (23). Thus existence of feasible points for (23) is now trivial. Existence of solutions of (23) can be guaranteed analogously to Theorem 2ii.

Theorem 3. *i. The null-space of the constraint matrix* (23) *has dimension* $(\sum_{i=1}^{n_\Omega} n_i^\Gamma) - n_A \geq 1$.

ii. Let $\widetilde{\boldsymbol{x}}_i^\Gamma$, $i = 1, \ldots, n_\Omega$, *satisfy the constraints (23b) and let* $\widetilde{\boldsymbol{x}}_i^\Omega$, $i = 1, \ldots, n_\Omega$, *be arbitrary. If the residual function* $\boldsymbol{r}$ *and the maps* $\boldsymbol{g}_i^\Omega$, $\boldsymbol{g}_i^\Gamma$, $i = 1, \ldots, n_\Omega$, *are continuous and if the level set*

$$L = \Big\{ (\widehat{\boldsymbol{x}}_1^\Omega, \widehat{\boldsymbol{x}}_1^\Gamma, \ldots, \widehat{\boldsymbol{x}}_{n_\Omega}^\Omega, \widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma) \; : \; \sum_{i=1}^{n_\Omega} \widehat{\boldsymbol{A}}_i \widehat{\boldsymbol{x}}_i^\Gamma = \boldsymbol{0}, \\ \sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i \left(\boldsymbol{g}_i^\Omega \left(\widehat{\boldsymbol{x}}_i^\Omega \right), \boldsymbol{g}_i^\Gamma \left(\widehat{\boldsymbol{x}}_i^\Gamma \right) \right) \right\|_2^2 \leq \sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i \left(\boldsymbol{g}_i^\Omega \left(\widetilde{\boldsymbol{x}}_i^\Omega \right), \boldsymbol{g}_i^\Gamma \left(\widetilde{\boldsymbol{x}}_i^\Gamma \right) \right) \right\|_2^2 \Big\}$$

is bounded, then (23) *has a solution.*

Proof. The first part follows immediately from the properties of the constraint matrix (23). The proof of ii. is analogous to the proof of Theorem 2ii. □

So far, we have specified our DD-LSPG-ROM (13) or (23) given maps $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$ such that (11) holds, or given maps $\boldsymbol{g}_i^\Omega$, $\boldsymbol{g}_j^p$ and implied interface maps (22) such that (11) holds. Next we discuss how these maps can be computed. In the following Section 3.3 we first review traditional approaches based on linear subspaces to compute $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$ (or $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_j^p$). In Section 3.4 we will then introduce the nonlinear-manifold ROM.

3.3. Linear-subspace ROM

First we review linear subspace approximation to construct the maps $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$, or $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_j^p$. We will refer to resulting ROM as LS-ROM. The LS-ROM approach supposes that the state solutions of the FOM are contained in a low-dimensional linear subspace. A basis for the linear subspace is then computed, resulting in a ROM whose state consists of the generalized coordinates of the state solution in the reduced subspace. The use of LS-ROM for the DD problem (13) has already been considered in [16], where the LS-ROM bases are computed using POD, but in principle any choice of basis can be used. The numerics in Section 7 also use POD for consistency with previous works. We briefly review POD here for completeness. A thorough treatment of POD can be found in [23].

As mentioned above, the LS-ROM approach approximates the FOM states $\boldsymbol{x}_i^\Omega$, $\boldsymbol{x}_i^\Gamma$ in a linear subspace. Hence $\boldsymbol{g}_i^\Omega : \mathbb{R}^{n_i^\Omega} \to \mathbb{R}^{N_i^\Omega}$ and $\boldsymbol{g}_i^\Gamma : \mathbb{R}^{n_i^\Gamma} \to \mathbb{R}^{N_i^\Gamma}$ are linear maps,

$$\boldsymbol{g}_i^\Omega : \quad \widehat{\boldsymbol{x}}_i^\Omega \mapsto \boldsymbol{\Phi}_i^\Omega \widehat{\boldsymbol{x}}_i^\Omega, \qquad\qquad \boldsymbol{g}_i^\Gamma : \quad \widehat{\boldsymbol{x}}_i^\Gamma \mapsto \boldsymbol{\Phi}_i^\Gamma \widehat{\boldsymbol{x}}_i^\Gamma,$$

where $\boldsymbol{\Phi}_i^\Omega \in \mathbb{R}^{N_i^\Omega \times n_i^\Omega}$ and $\boldsymbol{\Phi}_i^\Gamma \in \mathbb{R}^{N_i^\Gamma \times n_i^\Gamma}$ are basis matrices corresponding to the reduced linear subspaces. Consequently, the Jacobians $\frac{d}{d\widehat{\boldsymbol{x}}_i^\Omega} \boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega) = \boldsymbol{\Phi}_i^\Omega$ and $\frac{d}{d\widehat{\boldsymbol{x}}_i^\Gamma} \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma) = \boldsymbol{\Phi}_i^\Gamma$ are constant and do not need to be recomputed at each iteration of the SQP solver described in Section 4 that is used to solve (13) or (23).

The POD bases are computed by minimizing the reconstruction error for a set of snapshots. First we focus on constructing POD bases for the WFPC formulation. Recall that the residual functions $\boldsymbol{r}_i$ are parameterized with parameter space $\mathcal{D} \subset \mathbb{R}^{N_\mu}$. Let $\{\boldsymbol{\mu}_\ell^{\text{train}}\}_{\ell=1}^{n_\mu} \subset \mathcal{D}$ be a set of training parameters, and solve the DD FOM (9) for each parameter $\boldsymbol{\mu}_\ell^{\text{train}}$ to obtain FOM solutions $(\boldsymbol{x}_i^\Omega(\boldsymbol{\mu}_\ell^{\text{train}}), \boldsymbol{x}_i^\Gamma(\boldsymbol{\mu}_\ell^{\text{train}}))$, $i = 1, \ldots, n_\Omega$. Of course, one can solve the monolithic, single-domain FOM (1) at $\boldsymbol{\mu} = \boldsymbol{\mu}_\ell^{\text{train}}$, and restrict the solution $\boldsymbol{x}(\boldsymbol{\mu}_\ell^{\text{train}})$ to the subdomain interior and interface states, $\boldsymbol{x}_i^\Omega(\boldsymbol{\mu}_\ell^{\text{train}}) = \boldsymbol{P}_i^\Omega \boldsymbol{x}(\boldsymbol{\mu}_\ell^{\text{train}})$, $\boldsymbol{x}_i^\Gamma(\boldsymbol{\mu}_\ell^{\text{train}}) = \boldsymbol{P}_i^\Gamma \boldsymbol{x}(\boldsymbol{\mu}_\ell^{\text{train}})$. One then computes bases $\boldsymbol{\Phi}_i^\Omega$ and $\boldsymbol{\Phi}_i^\Gamma$ using the SVD applied to snapshot matrices for the interior and interface states

$$\boldsymbol{X}_i^\Omega = \left[\boldsymbol{x}_i^\Omega(\boldsymbol{\mu}_1^{\text{train}}) \quad \ldots \quad \boldsymbol{x}_i^\Omega(\boldsymbol{\mu}_{n_\mu}^{\text{train}}) \right] \in \mathbb{R}^{N_i^\Omega \times n_\mu}, \tag{24a}$$

$$\boldsymbol{X}_i^\Gamma = \left[\boldsymbol{x}_i^\Gamma(\boldsymbol{\mu}_1^{\text{train}}) \quad \ldots \quad \boldsymbol{x}_i^\Gamma(\boldsymbol{\mu}_{n_\mu}^{\text{train}}) \right] \in \mathbb{R}^{N_i^\Gamma \times n_\mu}. \tag{24b}$$

The process is the same for $\boldsymbol{\Phi}_i^\Omega$ and $\boldsymbol{\Phi}_i^\Gamma$ and therefore we drop the superscript Ω or Γ and describe the process to compute a basis $\boldsymbol{\Phi}_i$ from a generic snapshot matrix $\boldsymbol{X}_i \in \mathbb{R}^{N_i \times n_\mu}$.

One computes the 'thin' SVD $\boldsymbol{X}_i = \boldsymbol{U}_i \boldsymbol{\Sigma}_i \boldsymbol{V}_i^T$ of the snapshot matrix, where $\boldsymbol{U}_i \in \mathbb{R}^{N_i \times m_i}$ is the matrix of left singular vectors, $\boldsymbol{\Sigma}_i \in \mathbb{R}^{m_i \times m_i}$ is the diagonal matrix of singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{m_i} \geq 0$, $\boldsymbol{V}_i \in \mathbb{R}^{n_\mu \times m_i}$ is the matrix of right singular vectors, and $m_i = \min\{N_i, n_\mu\}$. Given a tolerance $\nu_i \in (0,1)$ one computes n_i as the smallest integer such that

$$\sum_{j=1}^{n_i} \sigma_j^2 \geq (1-\nu_i) \sum_{j=1}^{m_i} \sigma_j^2, \tag{25}$$

and selects

$$\boldsymbol{\Phi}_i = \boldsymbol{U}_i(:, 1:n_i).$$

The POD basis $\boldsymbol{\Phi}_i$ minimize the snapshot reconstruction error $\left\|\boldsymbol{X}_i - \boldsymbol{\Phi}_i(\boldsymbol{\Phi}_i)^T \boldsymbol{X}_i\right\|_F^2$ among all possible orthogonal basis matrices of sizes $N_i \times n_i$. See, e.g., [23].

POD basis construction for the SRPC formulation from Section 3.2 is similar. In fact, the bases $\boldsymbol{\Phi}_i^\Omega$, $i = 1, \ldots, n_\Omega$, are computed as before, and the bases $\boldsymbol{\Phi}_i^\Gamma$, $i = 1, \ldots, n_\Omega$, are computed from bases $\boldsymbol{\Phi}_j^p$ on the ports. Because $\boldsymbol{x}_j^p(\boldsymbol{\mu}_\ell^{\text{train}}) = \boldsymbol{P}_i^j \boldsymbol{x}_i^\Gamma(\boldsymbol{\mu}_\ell^{\text{train}})$ for any $i \in P(j)$ and all ports $P(j)$, the snapshot matrices restricted to port $P(j)$ are

$$\boldsymbol{X}_j^p = \boldsymbol{P}_i^j \boldsymbol{X}_i^\Gamma \quad \text{for any } i \in P(j). \tag{26}$$

For each port $P(j)$, the POD basis $\boldsymbol{\Phi}_j^p = \boldsymbol{U}_j^p(:, 1:n_j^p)$ is computed from the 'thin' SVD $\boldsymbol{X}_j^p = \boldsymbol{U}_j^p \boldsymbol{\Sigma}_j^p (\boldsymbol{V}_j^p)^T$ as described before. With the port basis matrices $\boldsymbol{\Phi}_j^p$ the linear map $\boldsymbol{g}_i^\Gamma$ is constructed following (22),

$$\boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma) := \sum_{j \in Q(i)} (\boldsymbol{P}_i^j)^T \boldsymbol{\Phi}_j^p \widehat{\boldsymbol{P}}_i^j \widehat{\boldsymbol{x}}_i^\Gamma = \boldsymbol{\Phi}_i^\Gamma \widehat{\boldsymbol{x}}_i^\Gamma, \quad \text{where} \quad \boldsymbol{\Phi}_i^\Gamma = \sum_{j \in Q(i)} (\boldsymbol{P}_i^j)^T \boldsymbol{\Phi}_j^p \widehat{\boldsymbol{P}}_i^j.$$

3.4. Nonlinear-manifold ROM

The ROM approach that we focus on in this work is the nonlinear manifold approach, also referred to as NM-ROM. Rather than supposing that the state solutions of the FOM are contained in a low-dimensional linear subspace as in LS-ROM, one supposes that the FOM state solutions are contained in a low-dimensional nonlinear manifold. To build the NM-ROM, one must compute a suitable mapping from a low-dimensional coordinate space, often referred to as the *latent space*, to the manifold of candidate state solutions, or *trial manifold*. Solving the ROM then yields the generalized coordinates in the latent space for a solution in the trial manifold. The approach considered here for computing the nonlinear trial manifold is similar to the approach in [56, Sec. 3], which uses wide, shallow, and sparse autoencoders to compute suitable mappings from the latent space to the trial manifold. Further information regarding the autoencoder architecture we use is given in Section 5.

To compute a DD ROM using the NM-ROM approach, one must compute continuously differentiable nonlinear mappings from a suitably chosen latent space to the trial manifold. Hence the nonlinear functions $\boldsymbol{g}_i^\Omega : \mathbb{R}^{n_i^\Omega} \to \mathbb{R}^{N_i^\Omega}$ and $\boldsymbol{g}_i^\Gamma : \mathbb{R}^{n_i^\Gamma} \to \mathbb{R}^{N_i^\Gamma}$ defined in (11) are computed as the *decoders* of autoencoders $\boldsymbol{a}_i^\Omega : \mathbb{R}^{N_i^\Omega} \to \mathbb{R}^{N_i^\Omega}$ and $\boldsymbol{a}_i^\Gamma : \mathbb{R}^{N_i^\Gamma} \to \mathbb{R}^{N_i^\Gamma}$. The autoencoders $\boldsymbol{a}_i^\Omega$ and $\boldsymbol{a}_i^\Gamma$ consist of two parts each: encoders $\boldsymbol{h}_i^\Omega : \mathbb{R}^{N_i^\Omega} \to \mathbb{R}^{n_i^\Omega}$ and $\boldsymbol{h}_i^\Gamma : \mathbb{R}^{N_i^\Gamma} \to \mathbb{R}^{n_i^\Gamma}$, and decoders $\boldsymbol{g}_i^\Omega : \mathbb{R}^{n_i^\Omega} \to \mathbb{R}^{N_i^\Omega}$ and $\boldsymbol{g}_i^\Gamma : \mathbb{R}^{n_i^\Gamma} \to \mathbb{R}^{N_i^\Gamma}$. The encoders map inputs from the high-dimensional state space to a low-dimensional latent space, while the decoders map elements from the low-dimensional space to the high-dimensional state space. The autoencoders $\boldsymbol{a}_i^\Omega$ and $\boldsymbol{a}_i^\Gamma$ are then defined via the function compositions

$$\boldsymbol{a}_i^\Omega = \boldsymbol{g}_i^\Omega \circ \boldsymbol{h}_i^\Omega, \qquad \boldsymbol{a}_i^\Gamma = \boldsymbol{g}_i^\Gamma \circ \boldsymbol{h}_i^\Gamma.$$

In this work, the encoders and decoders are neural networks such that the autoencoder approximates its inputs:

$$\boldsymbol{x}_i^\Omega \approx \boldsymbol{a}_i^\Omega(\boldsymbol{x}_i^\Omega) = \boldsymbol{g}_i^\Omega(\boldsymbol{h}_i^\Omega(\boldsymbol{x}_i^\Omega)), \qquad \boldsymbol{x}_i^\Gamma \approx \boldsymbol{a}_i^\Gamma(\boldsymbol{x}_i^\Gamma) = \boldsymbol{g}_i^\Gamma(\boldsymbol{h}_i^\Gamma(\boldsymbol{x}_i^\Gamma)), \qquad i = 1, \ldots, n_\Omega.$$

Further details on the neural network architecture used can be found in Section 5. The decoders $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$ can be interpreted as approximate inverses of the encoders $\boldsymbol{h}_i^\Omega$ and $\boldsymbol{h}_i^\Gamma$. In the SRPC case, the autoencoder $\boldsymbol{a}_i^\Gamma$ is composed of autoencoders $\boldsymbol{a}_j^p : \mathbb{R}^{N_j^p} \to \mathbb{R}^{N_j^p}$ with encoder $\boldsymbol{h}_j^p : \mathbb{R}^{N_j^p} \to \mathbb{R}^{n_j^p}$ and decoder $\boldsymbol{g}_j^p : \mathbb{R}^{n_j^p} \to \mathbb{R}^{N_j^p}$ for each port $P(j)$.

The mean-square-error (MSE) losses for the interior, interface, and port states are defined as

$$\mathcal{L}_i^\Omega := \frac{1}{n_\mu} \sum_{\ell=1}^{n_\mu} \left\| \boldsymbol{x}_i^\Omega(\boldsymbol{\mu}_\ell^{\text{train}}) - \boldsymbol{g}_i^\Omega(\boldsymbol{h}_i^\Omega(\boldsymbol{x}_i^\Omega(\boldsymbol{\mu}_\ell^{\text{train}}))) \right\|_2^2, \qquad i = 1, \ldots, n_\Omega, \tag{27a}$$

$$\mathcal{L}_i^\Gamma := \frac{1}{n_\mu} \sum_{\ell=1}^{n_\mu} \left\| \boldsymbol{x}_i^\Gamma(\boldsymbol{\mu}_\ell^{\text{train}}) - \boldsymbol{g}_i^\Gamma(\boldsymbol{h}_i^\Gamma(\boldsymbol{x}_i^\Gamma(\boldsymbol{\mu}_\ell^{\text{train}}))) \right\|_2^2, \qquad i = 1, \ldots, n_\Omega, \tag{27b}$$

$$\mathcal{L}_j^p := \frac{1}{n_\mu} \sum_{\ell=1}^{n_\mu} \left\| \boldsymbol{x}_j^p(\boldsymbol{\mu}_\ell^{\text{train}}) - \boldsymbol{g}_j^p(\boldsymbol{h}_j^p(\boldsymbol{x}_j^p(\boldsymbol{\mu}_\ell^{\text{train}}))) \right\|_2^2, \qquad j = 1, \ldots, n_p, \tag{27c}$$

where $\boldsymbol{x}_i^\Omega(\boldsymbol{\mu}_\ell^{\text{train}})$ and $\boldsymbol{x}_i^\Gamma(\boldsymbol{\mu}_\ell^{\text{train}})$ are snapshots of the interior and interface states on subdomain i at parameter $\boldsymbol{\mu}_\ell^{\text{train}}$ and $\boldsymbol{x}_j^p(\boldsymbol{\mu}_\ell^{\text{train}})$ is the state on port $P(j)$, as discussed in Section 3.3. In the WPFC case, the interior state and interface state autoencoders $\boldsymbol{a}_i^\Omega$ and $\boldsymbol{a}_i^\Gamma$ are trained by minimizing the interior and interface losses $\mathcal{L}_i^\Omega$ and $\mathcal{L}_i^\Gamma$, respectively. In the SPRC case, the interior state autoencoders $\boldsymbol{a}_i^\Omega$ and the port autoencoders $\boldsymbol{a}_j^p$ are trained by minimizing the interior and interface losses $\mathcal{L}_i^\Omega$ and $\mathcal{L}_j^p$, respectively. The interface state autoencoders $\boldsymbol{a}_i^\Gamma$ are implied by the port autoencoders. Specifically, $\boldsymbol{h}_i^\Gamma$ is

$$\boldsymbol{h}_i^\Gamma(\boldsymbol{x}_i^\Gamma) = \sum_{j \in Q(i)} (\widehat{\boldsymbol{P}}_i^j)^T \boldsymbol{h}_j^p(\boldsymbol{P}_i^j \boldsymbol{x}_i^\Gamma), \tag{28}$$

and $\boldsymbol{g}_i^\Gamma$ is defined using equation (22).

Notice that minimizing the MSE loss is equivalent to minimizing the snapshot reconstruction error, which is exactly how POD bases are constructed, as discussed in Section 3.3. Training the autoencoders can also be interpreted as "learning" the forward and inverse mappings from the latent space of generalized coordinates to the nonlinear trial manifold. After training, the decoders $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$ are used for the DD NM-ROM (13) or (23).

4. Sequential quadratic programming solver

4.1. Lagrange-Gauss-Newton SQP method

The problems (13) and (23) are nonlinear programs (NLPs) with equality constraints, and can be solved using sequential quadratic programming (SQP) [64], [65, Ch. 18]. The SQP solver detailed below amounts to applying a Newton-type method to the Karush-Kuhn-Tucker (KKT) necessary optimality conditions. Note that the SQP solver described in this section can also be applied to the FOM (10) by considering the case $\boldsymbol{B}_i = \boldsymbol{I}$ and $\boldsymbol{g}_i^\Omega, \boldsymbol{g}_i^\Gamma$ equal to the identity mapping.

To develop a solver that is applicable to either the WFPC (13) or the SPRC formulations (23), we define the constraint functions $\widetilde{\boldsymbol{A}}_i : \mathbb{R}^{n_i^\Gamma} \to \mathbb{R}^{n_A}$ and consider the general formulation

$$\min_{(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma), i=1,\ldots,n_\Omega} \quad \frac{1}{2} \sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i \left(\boldsymbol{g}_i^\Omega \left(\widehat{\boldsymbol{x}}_i^\Omega \right), \boldsymbol{g}_i^\Gamma \left(\widehat{\boldsymbol{x}}_i^\Gamma \right) \right) \right\|_2^2 \tag{29a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_\Omega} \widetilde{\boldsymbol{A}}_i(\widehat{\boldsymbol{x}}_i^\Gamma) = \boldsymbol{0}. \tag{29b}$$

In the WFPC case $\widetilde{\boldsymbol{A}}_i(\widehat{\boldsymbol{x}}_i^\Omega) = \boldsymbol{C}\boldsymbol{A}_i \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)$ and the constraints are nonlinear in the case of NM-ROMs. In the SRPC case $\widetilde{\boldsymbol{A}}_i(\widehat{\boldsymbol{x}}_i^\Gamma) = \widehat{\boldsymbol{A}}_i \widehat{\boldsymbol{x}}_i^\Gamma$ and the constraints are always linear.

To apply the SQP solver, one first writes the Lagrangian

$$\widehat{L}(\widehat{\boldsymbol{x}}_1^\Omega, \widehat{\boldsymbol{x}}_1^\Gamma, \ldots, \widehat{\boldsymbol{x}}_{n_\Omega}^\Omega, \widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma, \widehat{\boldsymbol{\lambda}}) = \frac{1}{2}\sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i\left(\boldsymbol{g}_i^\Omega\left(\widehat{\boldsymbol{x}}_i^\Omega\right), \boldsymbol{g}_i^\Gamma\left(\widehat{\boldsymbol{x}}_i^\Gamma\right)\right)\right\|_2^2 + \sum_{i=1}^{n_\Omega} \widehat{\boldsymbol{\lambda}}^T \widetilde{\boldsymbol{A}}_i(\widehat{\boldsymbol{x}}_i^\Gamma) \tag{30}$$

for the NLP (29), where $\widehat{\boldsymbol{\lambda}} \in \mathbb{R}^{n_A}$ are the Lagrange multipliers associated with the DD-ROM constraints (29b).

Let $\nabla_{\boldsymbol{v}}$ and $\frac{\partial}{\partial \boldsymbol{v}}$ denote the partial gradient and partial Jacobian with respect to $\boldsymbol{v}$, respectively, and let $\frac{d}{d\boldsymbol{v}}$ denote the Jacobian. The first order necessary optimality conditions are

$$\nabla_{\widehat{\boldsymbol{x}}_i^\Omega} \widehat{L}(\widehat{\boldsymbol{x}}_1^\Omega, \widehat{\boldsymbol{x}}_1^\Gamma, \ldots, \widehat{\boldsymbol{x}}_{n_\Omega}^\Omega, \widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma, \widehat{\boldsymbol{\lambda}}) = \boldsymbol{\rho}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma) = \boldsymbol{0}, \qquad i = 1, \ldots, n_\Omega, \tag{31a}$$

$$\nabla_{\widehat{\boldsymbol{x}}_i^\Gamma} \widehat{L}(\widehat{\boldsymbol{x}}_1^\Omega, \widehat{\boldsymbol{x}}_1^\Gamma, \ldots, \widehat{\boldsymbol{x}}_{n_\Omega}^\Omega, \widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma, \widehat{\boldsymbol{\lambda}}) = \boldsymbol{\rho}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma, \widehat{\boldsymbol{\lambda}}) = \boldsymbol{0}, \qquad i = 1, \ldots, n_\Omega, \tag{31b}$$

$$\nabla_{\widehat{\boldsymbol{\lambda}}} \widehat{L}(\widehat{\boldsymbol{x}}_1^\Omega, \widehat{\boldsymbol{x}}_1^\Gamma, \ldots, \widehat{\boldsymbol{x}}_{n_\Omega}^\Omega, \widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma, \widehat{\boldsymbol{\lambda}}) = \sum_{i=1}^{n_\Omega} \widetilde{\boldsymbol{A}}_i(\widehat{\boldsymbol{x}}_i^\Gamma) = \boldsymbol{0}, \tag{31c}$$

where

$$\boldsymbol{\rho}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma) = \frac{d\boldsymbol{g}_i^\Omega}{d\widehat{\boldsymbol{x}}_i^\Omega}(\widehat{\boldsymbol{x}}_i^\Omega)^T \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}_i^\Omega}\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big)^T \boldsymbol{B}_i^T \boldsymbol{B}_i \boldsymbol{r}_i\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big), \tag{32a}$$

$$\boldsymbol{\rho}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma, \widehat{\boldsymbol{\lambda}}) = \frac{d\boldsymbol{g}_i^\Gamma}{d\widehat{\boldsymbol{x}}_i^\Gamma}(\widehat{\boldsymbol{x}}_i^\Gamma)^T \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}_i^\Gamma}\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big)^T \boldsymbol{B}_i^T \boldsymbol{B}_i \boldsymbol{r}_i\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big) + \frac{d\widetilde{\boldsymbol{A}}_i}{d\widehat{\boldsymbol{x}}_i^\Gamma}(\widehat{\boldsymbol{x}}_i^\Gamma)^T \widehat{\boldsymbol{\lambda}} \tag{32b}$$

are the gradients of the Lagrangian with respect to the subdomain variables $\widehat{\boldsymbol{x}}_i^\Omega$ and $\widehat{\boldsymbol{x}}_i^\Gamma$, respectively.

A Newton-type method applied to (31) yields the SQP iterations

$$\begin{bmatrix} \boldsymbol{H}_1(\widehat{\boldsymbol{x}}_1^{\Omega(k)}, \widehat{\boldsymbol{x}}_1^{\Gamma(k)}) & & \cdots & \boldsymbol{E}_1(\widehat{\boldsymbol{x}}_1^\Gamma)^T \\ & \ddots & & \vdots \\ & & \boldsymbol{H}_{n_\Omega}(\widehat{\boldsymbol{x}}_{n_\Omega}^{\Omega(k)}, \widehat{\boldsymbol{x}}_{n_\Omega}^{\Gamma(k)}) & \boldsymbol{E}_{n_\Omega}(\widehat{\boldsymbol{x}}_1^\Gamma)^T \\ \boldsymbol{E}_1(\widehat{\boldsymbol{x}}_1^\Gamma) & \cdots & \boldsymbol{E}_{n_\Omega}(\widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma) & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_1^{(k)} \\ \vdots \\ \boldsymbol{s}_{n_\Omega}^{(k)} \\ \boldsymbol{s}^{\widehat{\boldsymbol{\lambda}}(k)} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{\rho}_1(\widehat{\boldsymbol{x}}_1^{\Omega(k)}, \widehat{\boldsymbol{x}}_1^{\Gamma(k)}, \widehat{\boldsymbol{\lambda}}^{(k)}) \\ \vdots \\ \boldsymbol{\rho}_{n_\Omega}(\widehat{\boldsymbol{x}}_{n_\Omega}^{\Omega(k)}, \widehat{\boldsymbol{x}}_{n_\Omega}^{\Gamma(k)}, \widehat{\boldsymbol{\lambda}}^{(k)}) \\ \sum_{i=1}^{n_\Omega} \widetilde{\boldsymbol{A}}_i(\widehat{\boldsymbol{x}}_i^{\Gamma(k)}) \end{bmatrix}, \tag{33}$$

where k is the SQP iteration index, $\boldsymbol{H}_i(\widehat{\boldsymbol{x}}_i^{\Omega(k)}, \widehat{\boldsymbol{x}}_i^{\Gamma(k)})$ is the Hessian of the Lagrangian with respect to the subdomain variables $(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma)$ evaluated at $(\widehat{\boldsymbol{x}}_i^{\Omega(k)}, \widehat{\boldsymbol{x}}_i^{\Gamma(k)})$ or an approximation of this Hessian, and where

$$\boldsymbol{E}_i(\widehat{\boldsymbol{x}}_i^\Gamma) = \begin{bmatrix} \boldsymbol{0} & \frac{d\widetilde{\boldsymbol{A}}_i}{d\widehat{\boldsymbol{x}}_i^\Gamma}(\widehat{\boldsymbol{x}}_i^\Gamma) \end{bmatrix}, \qquad \boldsymbol{s}_i^{(k)} = \begin{bmatrix} \boldsymbol{s}_i^{\Omega(k)} \\ \boldsymbol{s}_i^{\Gamma(k)} \end{bmatrix}, \qquad \boldsymbol{\rho}_i(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma, \widehat{\boldsymbol{\lambda}}) = \begin{bmatrix} \boldsymbol{\rho}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma) \\ \boldsymbol{\rho}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma, \widehat{\boldsymbol{\lambda}}) \end{bmatrix}, \tag{34}$$

for $i = 1, \ldots, n_\Omega$. The next result on the unique solvability of (33) follows from adapting standard results to the block structure of (33).

Lemma 1. *If for $i = 1, \ldots, n_\Omega$ the matrices $\boldsymbol{H}_i(\widehat{\boldsymbol{x}}_i^{\Omega(k)}, \widehat{\boldsymbol{x}}_i^{\Gamma(k)})$ are positive definite on the null-space of $\boldsymbol{E}_i(\widehat{\boldsymbol{x}}_i^\Gamma)$, and if $\big(\boldsymbol{E}_1(\widehat{\boldsymbol{x}}_1^\Gamma), \ldots, \boldsymbol{E}_{n_\Omega}(\widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma)\big)$ has full row rank, then* (33) *has a unique solution.*

For a proof see, e.g., [66, Thm. 3.2], [65, Lemma 16.12].

We use a Gauss-Newton approximation of the Hessian, which is motivated by the following consideration. If the residuals $\boldsymbol{B}_i \boldsymbol{r}_i\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big)$ are small at the solution of (29), then the first order optimality condition (31b) implies that $\widehat{\boldsymbol{\lambda}}$ is small. Thus all second derivative terms in the true Hessians $\boldsymbol{H}_i(\widehat{\boldsymbol{x}}_i^{\Omega(k)}, \widehat{\boldsymbol{x}}_i^{\Gamma(k)})$ are multiplied by small residuals or small Lagrange multipliers. The Gauss-Newton Hessian approximation neglects these terms and approximates the Hessians by

$$\boldsymbol{H}_i(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma) = \boldsymbol{R}_i(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma)^T \boldsymbol{B}_i^T \boldsymbol{B}_i \boldsymbol{R}_i(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma), \tag{35a}$$

where

$$\boldsymbol{R}_i(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma) = \Big[\tfrac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}_i^\Omega}\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big) \tfrac{d\boldsymbol{g}_i^\Omega}{d\widehat{\boldsymbol{x}}_i^\Omega}(\widehat{\boldsymbol{x}}_i^\Omega), \quad \tfrac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}_i^\Gamma}\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big) \tfrac{d\boldsymbol{g}_i^\Gamma}{d\widehat{\boldsymbol{x}}_i^\Gamma}(\widehat{\boldsymbol{x}}_i^\Gamma) \Big] \tag{35b}$$

is the Jacobian of $\boldsymbol{r}_i$ with respect to $(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma)$. The advantage is that (35) only requires first order derivatives. Note that the FOM solution satisfies (9), i.e., the residual in the least squares formulation (10) is zero. Thus, if the ROM well approximates the FOM (9) or, equivalently, its least squares formulation (10), then we expect the residuals $\boldsymbol{B}_i \boldsymbol{r}_i\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big)$ to be small at the solution of (29) and the Gauss-Newton Hessian (35) to be good approximation of the true Hessian of the Lagrangian (30).

Note that with the notation (35), the gradients $\boldsymbol{\rho}_i(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma, \widehat{\boldsymbol{\lambda}})$ in (34) can be written as

$$\boldsymbol{\rho}_i(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma, \widehat{\boldsymbol{\lambda}}) = \boldsymbol{R}_i(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma)^T \boldsymbol{B}_i^T \boldsymbol{B}_i \boldsymbol{r}_i\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)\big) + \boldsymbol{E}_i(\widehat{\boldsymbol{x}}_i^\Gamma)^T \widehat{\boldsymbol{\lambda}}, \quad i = 1, \ldots, n_\Omega. \tag{36}$$

With the Gauss-Newton approximations (35), the SQP system (33) is essentially the optimality system for the quadratic program

$$\min_{\boldsymbol{s}_i = (\boldsymbol{s}_i^\Omega, \boldsymbol{s}_i^\Gamma), i=1,\ldots,n_\Omega} \quad \frac{1}{2} \sum_{i=1}^{n_\Omega} \Big\| \boldsymbol{B}_i \boldsymbol{r}_i\big(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^{\Omega(k)}), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^{\Gamma(k)})\big) + \boldsymbol{B}_i \boldsymbol{R}_i(\widehat{\boldsymbol{x}}_i^{\Omega(k)}, \widehat{\boldsymbol{x}}_i^{\Gamma(k)}) \boldsymbol{s}_i \Big\|_2^2 \tag{37a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_\Omega} \widetilde{\boldsymbol{A}}_i\big(\widehat{\boldsymbol{x}}_i^{\Gamma(k)}\big) + \frac{d}{d\widehat{\boldsymbol{x}}_i^\Gamma} \widetilde{\boldsymbol{A}}_i\big(\widehat{\boldsymbol{x}}_i^{\Gamma(k)}\big) \boldsymbol{s}_i = \boldsymbol{0}. \tag{37b}$$

More precisely, the following result holds.

Lemma 2. *If the assumptions of Lemma 1 hold, then the quadratic program* (37) *has a unique solution* $\boldsymbol{s}_i^{(k)} = (\boldsymbol{s}_i^{\Omega(k)}, \boldsymbol{s}_i^{\Gamma(k)})$, $i = 1, \ldots, n_\Omega$, *given by the solution of* (33). *The associated Lagrange multiplier for* (37) *is* $\widehat{\boldsymbol{\lambda}}^{(k)} + \boldsymbol{s}^{\widehat{\boldsymbol{\lambda}}(k)}$, *where* $\widehat{\boldsymbol{\lambda}}^{(k)}$ *is the Lagrange multiplier estimate in* (33) *and* $\boldsymbol{s}^{\widehat{\boldsymbol{\lambda}}(k)}$ *is the last component in the solution vector of* (33).

The proof of Lemma 2 follows from the necessary and sufficient optimality conditions (e.g., [65, Sec 16.1]) for the quadratic program (37). The necessary and sufficient optimality conditions for (37) are given by (33) with the terms $\boldsymbol{E}_i(\widehat{\boldsymbol{x}}_i^{\Gamma(k)})^T \widehat{\boldsymbol{\lambda}}^{(k)}$ (see (36)) moved from the right to the left hand side.

An advantage of the Gauss-Newton approximation is that no Lagrange multiplier estimate is needed in (37) or the associated optimality system. Of course, quantities like $\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega)$, $\boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)$, $\boldsymbol{B}_i \boldsymbol{r}_i(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma))$, $\boldsymbol{B}_i \boldsymbol{R}_i(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma)$, $\widetilde{\boldsymbol{A}}_i(\widehat{\boldsymbol{x}}_i^\Gamma)$, and $\frac{d\widetilde{\boldsymbol{A}}_i}{d\widehat{\boldsymbol{x}}_i^\Gamma}(\widehat{\boldsymbol{x}}_i^\Gamma)$ can be computed in parallel across the subdomains. Moreover, the block structure of the system (33) lends itself to a parallel solution strategy. However, since (33) corresponds to the ROM its size tends to be small and parallelism in its solution may yield less speedup than it would if applied to the DD formulation (9) of the FOM (1). The parallel implementation of the approach discussed in this paper is left to future work.

Given the solution of the SQP system (33), the new iterate, i.e., the new approximate solution of (29) is computed as

$$\widehat{\boldsymbol{x}}_i^{\Omega(k+1)} = \widehat{\boldsymbol{x}}_i^{\Omega(k)} + \alpha^{(k)} \boldsymbol{s}_i^{\Omega(k)}, \qquad i = 1, \ldots, n_\Omega, \tag{38a}$$

$$\widehat{\boldsymbol{x}}_i^{\Gamma(k+1)} = \widehat{\boldsymbol{x}}_i^{\Gamma(k)} + \alpha^{(k)} \boldsymbol{s}_i^{\Gamma(k)}, \qquad i = 1, \ldots, n_\Omega, \tag{38b}$$

with step size $\alpha^{(k)} \in (0,1]$. If one chooses to keep a Lagrange multiplier estimate, then $\widehat{\boldsymbol{\lambda}}^{(k+1)} = \widehat{\boldsymbol{\lambda}}^{(k)} + \alpha^{(k)} \boldsymbol{s}^{\widehat{\boldsymbol{\lambda}}(k)}$, $i = 1, \ldots, n_\Omega$, where $\boldsymbol{s}^{\widehat{\boldsymbol{\lambda}}(k)}$ is the last component in the solution vector of (33). The step size $\alpha^{(k)}$ is computed via line-search using a merit function that coordinates progress of the iterates (and Lagrange multipliers) towards feasibility and optimality. In this work, we simply use the norm of the gradients (31). This is an appropriate criterion if one starts sufficiently close to a (local) minimizer of (29), and this criterion yielded good results in our examples. In our examples, the step size is computed using a backtracking line search with the Armijo rule.

4.2. Convergence of SQP Solver

Convergence of the Lagrange-Gauss-Newton SQP method can be established using one of two related approaches. The iteration (38) with $\boldsymbol{s}_i^{\Omega(k)}, \boldsymbol{s}_i^{\Gamma(k)}$, $i = 1, \ldots, n_\Omega$, computed as the solution of (33) with Gauss-Newton Hessian approximation (35) can be interpreted and analyzed as a generalized Gauss-Newton iteration. See [67]. Alternatively, this iteration can also be viewed as an inexact Newton method applied to the first-order optimality conditions (31). The local convergence result using either approach requires that the Lagrange-Gauss-Newton SQP method is started sufficiently close to a (local) minimizer of (29). We summarize the convergence theory for inexact Newton methods (see, e.g. [65, Thm. 11.3]) in Theorem 4. First we define the following notation to improve readability. We group the vectors $(\widehat{\boldsymbol{x}}_1^\Omega, \widehat{\boldsymbol{x}}_1^\Gamma, \ldots, \widehat{\boldsymbol{x}}_{n_\Omega}^\Omega, \widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma)$ and $(\boldsymbol{s}_1^{\Omega(k)}, \boldsymbol{s}_1^{\Gamma(k)}, \ldots, \boldsymbol{s}_{n_\Omega}^{\Omega(k)}, \boldsymbol{s}_{n_\Omega}^{\Gamma(k)})$ as

$$\widehat{\boldsymbol{x}} = \begin{bmatrix} \widehat{\boldsymbol{x}}_1^\Omega \\ \widehat{\boldsymbol{x}}_1^\Gamma \\ \vdots \\ \widehat{\boldsymbol{x}}_{n_\Omega}^\Omega \\ \widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma \end{bmatrix} \in \mathbb{R}^{n_D}, \qquad \boldsymbol{s}_x^{(k)} = \begin{bmatrix} \boldsymbol{s}_1^{\Omega(k)} \\ \boldsymbol{s}_1^{\Gamma(k)} \\ \vdots \\ \boldsymbol{s}_{n_\Omega}^{\Omega(k)} \\ \boldsymbol{s}_{n_\Omega}^{\Gamma(k)} \end{bmatrix} \in \mathbb{R}^{n_D}. \tag{39a}$$

where $n_D = \sum_{i=1}^{n_\Omega} (n_i^\Omega + n_i^\Gamma)$. Furthermore let $\widehat{\boldsymbol{F}} : \mathbb{R}^{n_D + n_A} \to \mathbb{R}^{n_D + n_A}$,

$$\widehat{\boldsymbol{F}}(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{\lambda}}) = - \begin{bmatrix} \boldsymbol{\rho}_1(\widehat{\boldsymbol{x}}_1^{\Omega(k)}, \widehat{\boldsymbol{x}}_1^{\Gamma(k)}, \widehat{\boldsymbol{\lambda}}^{(k)}) \\ \vdots \\ \boldsymbol{\rho}_{n_\Omega}(\widehat{\boldsymbol{x}}_{n_\Omega}^{\Omega(k)}, \widehat{\boldsymbol{x}}_{n_\Omega}^{\Gamma(k)}, \widehat{\boldsymbol{\lambda}}^{(k)}) \\ \sum_{i=1}^{n_\Omega} \widetilde{\boldsymbol{A}}_i(\widehat{\boldsymbol{x}}_i^{\Gamma(k)}) \end{bmatrix}. \tag{39b}$$

denote the right hand side of the KKT system. Recall that if $\overline{\boldsymbol{x}} \in \mathbb{R}^{n_D}$ is a local minimizer of (29) with associated Lagrange multiplier $\overline{\boldsymbol{\lambda}} \in \mathbb{R}^{n_A}$, then $\widehat{\boldsymbol{F}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}}) = \boldsymbol{0}$. Next define the Hessian approximation $\boldsymbol{H} : \mathbb{R}^{n_D} \times \mathbb{R}^{n_A} \to \mathbb{R}^{n_D \times n_D}$,

$$\boldsymbol{H}(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{\lambda}}) = \begin{bmatrix} \boldsymbol{H}_1(\widehat{\boldsymbol{x}}_1^\Omega, \widehat{\boldsymbol{x}}_1^\Gamma, \widehat{\boldsymbol{\lambda}}) & & \\ & \ddots & \\ & & \boldsymbol{H}_{n_\Omega}(\widehat{\boldsymbol{x}}_{n_\Omega}^\Omega, \widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma, \widehat{\boldsymbol{\lambda}}) \end{bmatrix}, \tag{39c}$$

and constraint Jacobian $\frac{d}{d\widehat{\boldsymbol{x}}^\Gamma} \widetilde{\boldsymbol{A}} : \mathbb{R}^{\sum_{i=1}^{n_\Omega} n_i^\Gamma} \to \mathbb{R}^{n_A \times n_D}$,

$$\frac{d}{d\widehat{\boldsymbol{x}}^\Gamma} \widetilde{\boldsymbol{A}}(\widehat{\boldsymbol{x}}_1^\Gamma, \ldots, \widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma) = \begin{bmatrix} \boldsymbol{0} & \frac{d}{d\widehat{\boldsymbol{x}}_1^\Gamma} \widetilde{\boldsymbol{A}}_1(\widehat{\boldsymbol{x}}_1^\Gamma) & \ldots & \boldsymbol{0} & \frac{d}{d\widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma} \widetilde{\boldsymbol{A}}_{n_\Omega}(\widehat{\boldsymbol{x}}_{n_\Omega}^\Gamma) \end{bmatrix}. \tag{39d}$$

The convergence result can now be stated as follows.

Theorem 4. *Let $\boldsymbol{r}_i$, $\boldsymbol{g}_i^{\Omega}$, and $\boldsymbol{g}_i^{\Gamma}$ be continuously differentiable for all $i = 1, \dots, n_{\Omega}$. Let $\overline{\boldsymbol{x}}$ be a local minimizer of* (29) *such that the Jacobian $\frac{d}{d\widehat{\boldsymbol{x}}^{\Gamma}}\widetilde{\boldsymbol{A}}(\overline{\boldsymbol{x}}_1^{\Gamma}, \dots, \overline{\boldsymbol{x}}_{n_{\Omega}}^{\Gamma})$ has full row rank, let $\overline{\boldsymbol{\lambda}}$ denote the associated Lagrange multiplier and assume that $\boldsymbol{H}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}})$ is positive definite on the null-space of $\frac{d}{d\widehat{\boldsymbol{x}}^{\Gamma}}\widetilde{\boldsymbol{A}}(\overline{\boldsymbol{x}}_1^{\Gamma}, \dots, \overline{\boldsymbol{x}}_{n_{\Omega}}^{\Gamma})$. Furthermore, assume that $\widehat{\boldsymbol{F}}'(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{\lambda}})$ is Lipschitz continuous with Lipschitz constant K, and that the steps $\boldsymbol{s}_x^{(k)}$ satisfy*

$$\left\| \left(\nabla^2_{\widehat{\boldsymbol{x}}} \widehat{L}(\widehat{\boldsymbol{x}}^{(k)}, \widehat{\boldsymbol{\lambda}}^{(k)}) - \boldsymbol{H}(\widehat{\boldsymbol{x}}^{(k)}, \widehat{\boldsymbol{\lambda}}^{(k)}) \right) \boldsymbol{s}_x^{(k)} \right\|_2 \leq \eta_k \left\| \widehat{\boldsymbol{F}}(\widehat{\boldsymbol{x}}^{(k)}, \widehat{\boldsymbol{\lambda}}^{(k)}) \right\|_2$$

for some sequence of forcing parameters η_k, and where L is the Lagrangian defined in (30).

If $\{\eta_k\}$ satisfies $0 < \eta_k \leq \eta$ where η is such that $4\eta\bar{\kappa} < 1$ with $\bar{\kappa} = \|\widehat{\boldsymbol{F}}'(\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}})^{-1}\|_2 \|\widehat{\boldsymbol{F}}'(\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}})\|_2$, then for all $\sigma \in (4\eta\bar{\kappa}, 1)$ there exists an $\epsilon > 0$ such that for any $(\widehat{\boldsymbol{x}}^{(0)}, \widehat{\boldsymbol{\lambda}}^{(0)})$ with $\|(\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}}) - (\widehat{\boldsymbol{x}}^{(0)}, \widehat{\boldsymbol{\lambda}}^{(0)})\|_2 < \epsilon$, the sequence of iterates $\{(\widehat{\boldsymbol{x}}^{(k)}, \widehat{\boldsymbol{\lambda}}^{(k)})\}$ generated by the SQP solver converges to $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}})$, and the iterates satisfy

$$\begin{aligned}
\|(\widehat{\boldsymbol{x}}^{(k)}, \widehat{\boldsymbol{\lambda}}^{(k)}) - (\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}})\|_2 &\leq K\|\widehat{\boldsymbol{F}}'(\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}})^{-1}\|_2 \|(\widehat{\boldsymbol{x}}^{(k)}, \widehat{\boldsymbol{\lambda}}^{(k)}) - (\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}})\|_2^2 + 4\eta_k\bar{\kappa}\|(\widehat{\boldsymbol{x}}^{(k)}, \widehat{\boldsymbol{\lambda}}^{(k)}) - (\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}})\|_2 \\
&\leq \sigma \|(\widehat{\boldsymbol{x}}^{(k)}, \widehat{\boldsymbol{\lambda}}^{(k)}) - (\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}})\|_2.
\end{aligned}$$

See, e.g., [65, Thm. 11.3] for a proof of this theorem.

Remark 1. *As stated, Theorem 4 only guarantees a solution to the KKT system* (31)*, which are (first order) necessary optimality conditions. However one can give alternative inexactness conditions on Gauss-Newton Hessian approximations that ensure local convergence to a point at which the second order sufficient optimality conditions are satisfied. See [68, L. 2.5] for the unconstrained case and [67, Sec. 3.5] for the constrained case, but with ℓ_1 rather than ℓ_2 (=least squares) objective.*

5. Autoencoder architecture

Following [56], we consider the use of shallow, wide, sparse-masked autoencoders with smooth activation functions for representing the maps, $\boldsymbol{g}_i^{\Omega}$ and $\boldsymbol{g}_i^{\Gamma}$. Shallow networks are used for computational efficiency; fewer layers correspond to fewer repeated matrix-vector multiplications when evaluating the decoders. The shallow depth necessitates a wide network to maintain enough expressiveness for use in NM-ROM. Sparsity is applied at the decoder output layer so that hyper-reduction can be applied. Further details on hyper reduction are addressed in Section 5.3. Smooth activations are used to ensure that $\boldsymbol{g}_i^{\Omega}$ and $\boldsymbol{g}_i^{\Gamma}$ are continuously differentiable. In contrast with [56], we also apply a sparsity mask to the encoder input layer so that the encoders and decoders are symmetric across the latent layer. We found that applying a sparsity mask to the encoder input layer permitted the use of a wider network for the encoder, resulting in improved performance over a dense input layer. See Section 7.4 for further details.

5.1. Weak FOM-port formulation

First we detail the architectures used for the weak FOM-port constraint formulation. We use a single-layer architecture for the encoders and decoders with a smooth, non-polynomial activation function. The encoders, $\boldsymbol{h}_i^{\Omega}$ and $\boldsymbol{h}_i^{\Gamma}$, and decoders, $\boldsymbol{g}_i^{\Omega}$ and $\boldsymbol{g}_i^{\Gamma}$, are of the form

$$\boldsymbol{h}_i^{\Omega} : \mathbb{R}^{N_i^{\Omega}} \to \mathbb{R}^{n_i^{\Omega}}, \qquad \boldsymbol{h}_i^{\Omega}(\boldsymbol{x}_i^{\Omega}) = \boldsymbol{W}_2^{\boldsymbol{h}_i^{\Omega}} \boldsymbol{\sigma}_i^{\Omega}(\boldsymbol{W}_1^{\boldsymbol{h}_i^{\Omega}} \boldsymbol{x}_i^{\Omega} + \boldsymbol{b}_1^{\boldsymbol{h}_i^{\Omega}}), \tag{40a}$$

$$\boldsymbol{g}_i^{\Omega} : \mathbb{R}^{n_i^{\Omega}} \to \mathbb{R}^{N_i^{\Omega}}, \qquad \boldsymbol{g}_i^{\Omega}(\widehat{\boldsymbol{x}}_i^{\Omega}) = \boldsymbol{W}_2^{\boldsymbol{g}_i^{\Omega}} \boldsymbol{\sigma}_i^{\Omega}(\boldsymbol{W}_1^{\boldsymbol{g}_i^{\Omega}} \widehat{\boldsymbol{x}}_i^{\Omega} + \boldsymbol{b}_1^{\boldsymbol{g}_i^{\Omega}}), \tag{40b}$$

$$\boldsymbol{h}_i^{\Gamma} : \mathbb{R}^{N_i^{\Gamma}} \to \mathbb{R}^{n_i^{\Gamma}}, \qquad \boldsymbol{h}_i^{\Gamma}(\boldsymbol{x}_i^{\Gamma}) = \boldsymbol{W}_2^{\boldsymbol{h}_i^{\Gamma}} \boldsymbol{\sigma}_i^{\Gamma}(\boldsymbol{W}_1^{\boldsymbol{h}_i^{\Gamma}} \boldsymbol{x}_i^{\Gamma} + \boldsymbol{b}_1^{\boldsymbol{h}_i^{\Gamma}}), \tag{40c}$$

$$\boldsymbol{g}_i^{\Gamma} : \mathbb{R}^{n_i^{\Gamma}} \to \mathbb{R}^{N_i^{\Gamma}}, \qquad \boldsymbol{g}_i^{\Gamma}(\widehat{\boldsymbol{x}}_i^{\Gamma}) = \boldsymbol{W}_2^{\boldsymbol{g}_i^{\Gamma}} \boldsymbol{\sigma}_i^{\Gamma}(\boldsymbol{W}_1^{\boldsymbol{g}_i^{\Gamma}} \widehat{\boldsymbol{x}}_i^{\Gamma} + \boldsymbol{b}_1^{\boldsymbol{g}_i^{\Gamma}}), \tag{40d}$$

where

$$\boldsymbol{W}_1^{\boldsymbol{g}_i^\Omega}, \left(\boldsymbol{W}_2^{\boldsymbol{h}_i^\Omega}\right)^T \in \mathbb{R}^{w_i^\Omega \times n_i^\Omega}, \qquad \boldsymbol{W}_2^{\boldsymbol{g}_i^\Omega}, \left(\boldsymbol{W}_1^{\boldsymbol{h}_i^\Omega}\right)^T \in \mathbb{R}^{N_i^\Omega \times w_i^\Omega}, \tag{41a}$$

$$\boldsymbol{b}_1^{\boldsymbol{h}_i^\Omega} \in \mathbb{R}^{w_i^\Omega}, \qquad \boldsymbol{b}_1^{\boldsymbol{g}_i^\Omega} \in \mathbb{R}^{w_i^\Omega}, \tag{41b}$$

$$\boldsymbol{W}_1^{\boldsymbol{g}_i^\Gamma}, \left(\boldsymbol{W}_2^{\boldsymbol{h}_i^\Gamma}\right)^T \in \mathbb{R}^{w_i^\Gamma \times n_i^\Gamma}, \qquad \boldsymbol{W}_2^{\boldsymbol{g}_i^\Gamma}, \left(\boldsymbol{W}_1^{\boldsymbol{h}_i^\Gamma}\right)^T \in \mathbb{R}^{N_i^\Gamma \times w_i^\Gamma}, \tag{41c}$$

$$\boldsymbol{b}_1^{\boldsymbol{h}_i^\Gamma} \in \mathbb{R}^{w_i^\Gamma}, \qquad \boldsymbol{b}_1^{\boldsymbol{g}_i^\Gamma} \in \mathbb{R}^{w_i^\Gamma}, \tag{41d}$$

$\boldsymbol{\sigma}_i^\Omega, \boldsymbol{\sigma}_i^\Gamma$ are smooth, non-polynomial activation functions (e.g. Sigmoid or Swish), and where w_i^Ω, w_i^Γ are the network widths for all subdomains $i = 1, \ldots, n_\Omega$. The weight matrices $\boldsymbol{W}_2^{\boldsymbol{g}_i^\Omega}, \boldsymbol{W}_1^{\boldsymbol{h}_i^\Omega}, \boldsymbol{W}_2^{\boldsymbol{g}_i^\Gamma}$ and $\boldsymbol{W}_1^{\boldsymbol{h}_i^\Gamma}$ are all sparse, while the remaining weights and biases are dense.

The widths, w_i^Ω and w_i^Γ, as well as the sparsity patterns of $\boldsymbol{W}_2^{\boldsymbol{g}_i^\Omega}, \boldsymbol{W}_1^{\boldsymbol{h}_i^\Omega}, \boldsymbol{W}_2^{\boldsymbol{g}_i^\Gamma}$ and $\boldsymbol{W}_1^{\boldsymbol{h}_i^\Gamma}$ are hyper-parameters that require tuning. The use of a single-layer architecture of arbitrary width and non-polynomial activation, as defined in (40a-d), is motivated by the well-known universal approximation theorem [69], [70]. Furthermore, the use of a smooth activation function ensures that $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$ are continuously differentiable. This is important because the Jacobians $\frac{d\boldsymbol{g}_i^\Omega}{d\widehat{\boldsymbol{x}}_i^\Omega}$ and $\frac{d\boldsymbol{g}_i^\Gamma}{d\widehat{\boldsymbol{x}}_i^\Gamma}$ are required by the SQP solver discussed in Section 4. The autoencoders are trained by minimizing the MSE loss defined in equation (27).

5.2. *Strong ROM-port formulation*

Next we detail the architectures used for the strong ROM-port constraint formulation. As before, we use a single-layer architecture for the encoders and decoders with a smooth, non-polynomial activation function. The interior state encoders $\boldsymbol{h}_i^\Omega$ and decoders $\boldsymbol{g}_i^\Omega$ have the same architecture as in the weak FOM-port constraint formulation. Thus we focus on the interface encoders $\boldsymbol{h}_i^\Gamma$ and decoders $\boldsymbol{g}_i^\Gamma$. As stated in Section 3.4, in the strong ROM-port case, the interface encoders $\boldsymbol{h}_i^\Gamma$ and decoders $\boldsymbol{g}_i^\Gamma$ are composed of encoders $\boldsymbol{h}_j^p$ and decoders $\boldsymbol{g}_j^p$ for ports $P(j)$. These encoders $\boldsymbol{h}_j^p$ and decoders $\boldsymbol{g}_j^p$ are of the form

$$\boldsymbol{h}_j^p : \mathbb{R}^{N_j^p} \to \mathbb{R}^{n_j^p}, \qquad \boldsymbol{h}_j^p(\boldsymbol{x}_j^p) = \boldsymbol{W}_2^{\boldsymbol{h}_j^p} \boldsymbol{\sigma}_j^p(\boldsymbol{W}_1^{\boldsymbol{h}_j^p} \boldsymbol{x}_j^p + \boldsymbol{b}_1^{\boldsymbol{h}_j^p}), \tag{42a}$$

$$\boldsymbol{g}_j^p : \mathbb{R}^{n_j^p} \to \mathbb{R}^{N_j^p}, \qquad \boldsymbol{g}_j^p(\widehat{\boldsymbol{x}}_j^p) = \boldsymbol{W}_2^{\boldsymbol{g}_j^p} \boldsymbol{\sigma}_j^p(\boldsymbol{W}_1^{\boldsymbol{g}_j^p} \widehat{\boldsymbol{x}}_j^p + \boldsymbol{b}_1^{\boldsymbol{g}_j^p}), \tag{42b}$$

where

$$\boldsymbol{W}_1^{\boldsymbol{g}_j^p}, \left(\boldsymbol{W}_2^{\boldsymbol{h}_j^p}\right)^T \in \mathbb{R}^{w_j^p \times n_j^p}, \qquad \boldsymbol{W}_2^{\boldsymbol{g}_j^p}, \left(\boldsymbol{W}_1^{\boldsymbol{h}_j^p}\right)^T \in \mathbb{R}^{N_j^p \times w_j^p}, \tag{43a}$$

$$\boldsymbol{b}_1^{\boldsymbol{h}_j^p} \in \mathbb{R}^{w_j^p}, \qquad \boldsymbol{b}_1^{\boldsymbol{g}_j^p} \in \mathbb{R}^{w_j^p}, \tag{43b}$$

$\boldsymbol{\sigma}_j^p$ are smooth, non-polynomial activation functions (e.g., Sigmoid or Swish), and where w_j^p are the network widths for all ports $P(j), j = 1, \ldots, n_p$. The weight matrices $\boldsymbol{W}_2^{\boldsymbol{g}_j^p}$ and $\boldsymbol{W}_1^{\boldsymbol{h}_j^p}$ are sparse, while the remaining weights and biases are dense. As in the WFPC case, the width w_j^p and the sparsity patterns of $\boldsymbol{W}_2^{\boldsymbol{g}_j^p}$ and $\boldsymbol{W}_1^{\boldsymbol{h}_j^p}$ are hyper-parameters that require tuning. The autoencoders are trained by minimizing the MSE loss defined in equation (27).

Recall that the interface encoders $\boldsymbol{h}_i^\Gamma$ and $\boldsymbol{g}_i^\Gamma$ are computed using equations (28) and (22), respectively. The encoders $\boldsymbol{h}_i^\Gamma$ and decoders $\boldsymbol{g}_i^\Gamma$ can be written in the form (40c, d) as follows. For a given subdomain i, let $j_1, \ldots, j_{|Q(i)|}$ denote the indices of the subdomains contained in $Q(i)$. The weights and biases of $\boldsymbol{h}_i^\Gamma$ and $\boldsymbol{g}_i^\Gamma$ can then be assembled in block form as

$$\boldsymbol{W}_1^{\boldsymbol{h}_i^\Gamma} = \begin{bmatrix} \boldsymbol{W}_1^{\boldsymbol{h}_{j_1}^p} & & \\ & \ddots & \\ & & \boldsymbol{W}_1^{\boldsymbol{h}_{j_{|Q(i)|}}^p} \end{bmatrix} \begin{bmatrix} \boldsymbol{P}_i^{j_1} \\ \vdots \\ \boldsymbol{P}_i^{j_{|Q(i)|}} \end{bmatrix}, \qquad \boldsymbol{b}_1^{\boldsymbol{h}_i^\Gamma} = \begin{bmatrix} \boldsymbol{b}_1^{\boldsymbol{h}_{j_1}^p} \\ \vdots \\ \boldsymbol{b}_1^{\boldsymbol{h}_{j_{|Q(i)|}}^p} \end{bmatrix}, \tag{44a}$$

$$\boldsymbol{W}_2^{\boldsymbol{h}_i^\Gamma} = \begin{bmatrix} (\widehat{\boldsymbol{P}}_i^{j_1})^T & \dots & (\widehat{\boldsymbol{P}}_i^{j_{|Q(i)|}})^T \end{bmatrix} \begin{bmatrix} \boldsymbol{W}_2^{\boldsymbol{h}_{j_1}^p} & & \\ & \ddots & \\ & & \boldsymbol{W}_2^{\boldsymbol{h}_{j_{|Q(i)|}}^p} \end{bmatrix}, \tag{44b}$$

$$\boldsymbol{W}_1^{\boldsymbol{g}_i^\Gamma} = \begin{bmatrix} \boldsymbol{W}_1^{\boldsymbol{g}_{j_1}^p} & & \\ & \ddots & \\ & & \boldsymbol{W}_1^{\boldsymbol{g}_{j_{|Q(i)|}}^p} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{P}}_i^{j_1} \\ \vdots \\ \widehat{\boldsymbol{P}}_i^{j_{|Q(i)|}} \end{bmatrix}, \qquad \boldsymbol{b}_1^{\boldsymbol{g}_i^\Gamma} = \begin{bmatrix} \boldsymbol{b}_1^{\boldsymbol{g}_{j_1}^p} \\ \vdots \\ \boldsymbol{b}_1^{\boldsymbol{g}_{j_{|Q(i)|}}^p} \end{bmatrix}, \tag{44c}$$

$$\boldsymbol{W}_2^{\boldsymbol{g}_i^\Gamma} = \begin{bmatrix} (\boldsymbol{P}_i^{j_1})^T & \dots & (\boldsymbol{P}_i^{j_{|Q(i)|}})^T \end{bmatrix} \begin{bmatrix} \boldsymbol{W}_2^{\boldsymbol{g}_{j_1}^p} & & \\ & \ddots & \\ & & \boldsymbol{W}_2^{\boldsymbol{g}_{j_{|Q(i)|}}^p} \end{bmatrix}, \tag{44d}$$

with activation

$$\boldsymbol{\sigma}_i^\Gamma(\cdot) = \begin{bmatrix} \boldsymbol{\sigma}_{j_1}^p(\cdot) \\ \vdots \\ \boldsymbol{\sigma}_{j_{|Q(i)|}}^p(\cdot) \end{bmatrix}. \tag{44e}$$

5.3. Hyper-Reduction

If no hyper-reduction (HR) is applied (i.e. $\boldsymbol{B}_i = \boldsymbol{I}$) when solving DD ROM (13), the computational savings from the ROM is limited because the evaluation of the residuals $\boldsymbol{r}_i$ and their Jacobians still scales with the dimension of the FOM. Thus HR is applied to decrease the computational complexity caused by the nonlinearity of $\boldsymbol{r}_i$, and increase the computational speedup. Possible HR approaches include collocation ($\boldsymbol{B}_i = \boldsymbol{Z}_i$) and gappy POD ($\boldsymbol{B}_i = (\boldsymbol{Z}_i\boldsymbol{\Phi}_i^r)^\dagger \boldsymbol{Z}_i$) [61], [16]. In both cases, only a subsample of the residual components and their corresponding Jacobian components are computed. This subsample is determined by the row-sampling matrix $\boldsymbol{Z}_i$, which is typically computed greedily (see Remark 2).

Now for both cases $\boldsymbol{B}_i = \boldsymbol{Z}_i$ and $\boldsymbol{B}_i = (\boldsymbol{Z}_i\boldsymbol{\Phi}_i^r)^\dagger \boldsymbol{Z}_i$, one must compute the products $\boldsymbol{Z}_i\boldsymbol{r}_i$, $\boldsymbol{Z}_i\frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}_i^\Omega}$, and $\boldsymbol{Z}_i\frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}_i^\Gamma}$. In implementation, instead of computing matrix-vector or matrix-matrix products, one only needs to compute the entries of $\boldsymbol{r}_i$ and rows of $\frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}_i^\Omega}$ and $\frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}_i^\Gamma}$ that are sampled by $\boldsymbol{Z}_i$. Hence the application of HR is typically code-intrusive. Moreover, since only a subset of the entries of $\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega)$ and $\boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)$ are needed to evaluate $\boldsymbol{Z}_i\boldsymbol{r}_i$, $\boldsymbol{Z}_i\frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}_i^\Omega}$, and $\boldsymbol{Z}_i\frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}_i^\Gamma}$, only the corresponding outputs of the decoders $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$ need to be kept track of. This motivates the use of a sparsity mask in the last layer of the decoders, which was first introduced in the context of NM-ROM in [56].

Indeed, in the case of a dense linear layer, each node in the hidden layer is needed to compute one node in the output layer, thus limiting the computational savings gained through HR. If instead a sparsity mask is applied to the layer, only a subset of the hidden nodes is required to compute a node in the output layer. This allows for the computation of a *subnet*, which only keeps track of the nodes used to compute the output nodes remaining after HR. We discuss the computation of a subnet in Section 5.4. Figure 3 provides a visualization of a subnet. In this paper, we also apply the transpose of decoder sparsity mask to the input layer of the encoder, resulting in autoencoders whose architectures are

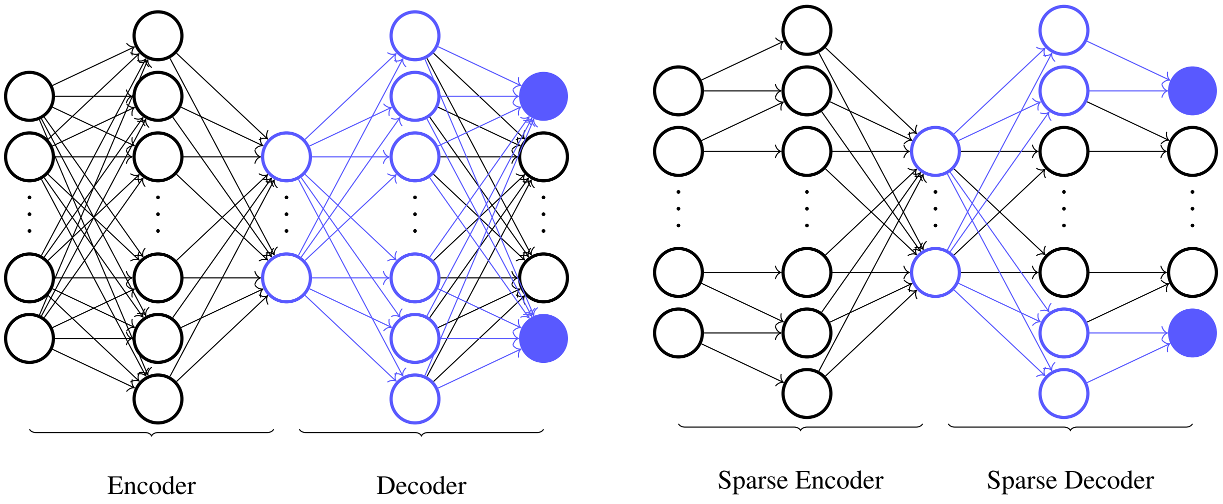


Figure 3: Left: Dense autoencoder. The HR nodes are represented by solid blue neurons, and the nodes required to compute the HR nodes are outlined in blue. Notice that each node in the decoder hidden layer are required to compute the HR nodes in the output layer. Right: Sparse autoencoder. The encoder input layer and decoder output layer are sparsely connected, and only the blue-outlined hidden nodes are required to compute the HR nodes. The sparse output layer allows one to only keep track of the blue connections to evaluate $\boldsymbol{g}_i^\Omega$, $\boldsymbol{g}_i^\Gamma$ and their Jacobians, resulting in computational speedup.

(approximately) symmetric across the latent layer. We found that this choice gave improved performance (i.e. ROM accuracy) over the architectures used in [56], which use dense encoder input layers.

Remark 2. *Following [16] and [56], we use [71, Algo. 3] to greedily compute a row sampling matrix $\boldsymbol{Z}_i$. This approach relies upon the computation of a residual basis $\boldsymbol{\Phi}_i^r$ for each subdomain. In practice, these residual bases are computed by applying POD to residual snapshots, which are collected from the iteration history of Newton's method when computing interior- and interface-state snapshots for ROM training.*

Remark 3. *For the WFPC case, the products $\boldsymbol{C}\boldsymbol{A}_i\boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)$ and $\widehat{\boldsymbol{\lambda}}^T\boldsymbol{C}\boldsymbol{A}_i\frac{d\boldsymbol{g}_i^\Gamma}{d\widehat{\boldsymbol{x}}_i^\Gamma}(\widehat{\boldsymbol{x}}_i^\Gamma)$ coming from the equality constraint appear in the KKT system* (33) *for the SQP solver. For LS-ROM, since the Jacobian of $\boldsymbol{g}_i^\Gamma$ is nothing but the POD basis matrix $\boldsymbol{\Phi}_i^\Gamma$, one can easily precompute $\boldsymbol{C}\boldsymbol{A}_i\boldsymbol{\Phi}_i^\Gamma$, thus making HR unnecessay for these quantities. However for NM-ROM, $d\boldsymbol{g}_i^\Gamma/d\widehat{\boldsymbol{x}}_i^\Gamma$ must be re-evaluated at each iteration of the SQP solver, thus introducing additional computation expense that is not present in LS-ROM. Currently, these quantities do not undergo HR in the NM-ROM case, and hence the decoders $\boldsymbol{g}_i^\Gamma$ for the entire interface states must be kept track of. While this limits the computational savings that can be obtained through HR, in practice the dimension of the FOM interface states N_i^Γ is much smaller than the dimension of the FOM interior states N_i^Ω, thus making the HR of $\boldsymbol{g}_i^\Gamma$ less critical than the HR of $\boldsymbol{g}_i^\Omega$. This issue is not present in the SRPC case because the constraints are purely linear.*

Remark 4. *In practice, the pattern of the sparsity mask is determined by a number of hyper parameters to be tuned by the user. Further details on the sparsity pattern used for our numerical results is discussed in Section 7.*

5.4. Construction of a Subnet

The authors in [56, Sec. 4.4.1] discuss the construction of a subnet in terms of gradients of the loss function with respect to the weights and biases of the sparse decoder. In this section, we present an alternative method for constructing the subnet solely by keeping track of indices of HR nodes. For simplicity, we consider a generic decoder $\boldsymbol{g}:\mathbb{R}^n\to\mathbb{R}^N$ of the form (40b, d) with width w. Computing subnets for each decoder, e.g., $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$, follows the same procedure. Recall that in the architecture considered in this paper, $\boldsymbol{W}_2 \in \mathbb{R}^{N\times w}$ is sparse and $\boldsymbol{W}_1 \in \mathbb{R}^{w\times n}$ is dense.

Let $\mathcal{I}_o \subset \{1,\dots,N\}$ denote the indices of the outputs of $\boldsymbol{g}$ that are selected through HR. To find the indices of the hidden nodes required to compute the HR nodes, find the index set $\mathcal{I}_h$ where

$$\mathcal{I}_h = \{j \in \{1,\dots,w\} \mid \exists\, i \in \mathcal{I}_o \text{ s.t. } (\boldsymbol{W}_2)_{ij} \neq 0\}.$$

The index sets $\mathcal{I}_o$ and $\mathcal{I}_h$ contain the indices of all nonzero elements in $\boldsymbol{W}_2$. Now let $i_1,\dots,i_{|\mathcal{I}_o|}$ and $j_1,\dots,j_{|\mathcal{I}_h|}$ denote the elements of $\mathcal{I}_o$ and $\mathcal{I}_h$ in ascending order, respectively. Define the matrix $\widetilde{\boldsymbol{W}}_2 \in \mathbb{R}^{|\mathcal{I}_o|\times|\mathcal{I}_h|}$ as

$$(\widetilde{\boldsymbol{W}}_2)_{\ell,k} = (\boldsymbol{W}_2)_{i_\ell,j_k}, \quad \forall\, \ell = 1,\dots,|\mathcal{I}_o|,\ k = 1,\dots,|\mathcal{I}_h|.$$

The matrix $\widetilde{\boldsymbol{W}}_2$ precisely consists of the connections in the subnet that remain after HR. Next, since the activation $\boldsymbol{\sigma}$ acts element-wise, the connections that remain in the first layer of the subnet can be represented by $\widetilde{\boldsymbol{W}}_1 \in \mathbb{R}^{|\mathcal{I}_h|\times n}$, which consists of nothing but the rows in $\boldsymbol{W}_1$ corresponding to the index set $\mathcal{I}_h$:

$$(\widetilde{\boldsymbol{W}}_1)_{k,:} = \boldsymbol{W}_{j_k,:}, \quad \forall\, k = 1,\dots,|\mathcal{I}_h|.$$

Lastly, the subnet bias $\widetilde{\boldsymbol{b}}_1 \in \mathbb{R}^{|\mathcal{I}_h|}$ is similarly defined as $(\widetilde{\boldsymbol{b}}_1)_k = (\boldsymbol{b}_1)_{j_k}$ for all $k = 1,\dots,|\mathcal{I}_h|$. The subnet $\widetilde{\boldsymbol{g}} : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{I}_o|}$ is then defined as

$$\widetilde{\boldsymbol{g}}(\widehat{\boldsymbol{x}}) = \widetilde{\boldsymbol{W}}_2\boldsymbol{\sigma}(\widetilde{\boldsymbol{W}}_1\widehat{\boldsymbol{x}} + \widetilde{\boldsymbol{b}}_1). \tag{45}$$

Remark 5. *This framework for computing a subnet can easily be extended to neural networks with arbitrarily many sparse linear layers provided that the sparsity patterns for each layer's weight matrix is known. Thus one could construct deep sparse autoencoders with narrower width than the architectures considered here. However, for this paper we only consider single-layer, wide, sparse decoders.*

6. Error Analysis

We present *a priori* and *a posteriori* error bounds analogous to those found in [16]. To simplify notation, analogous to the notation in Section 4.2, we denote the optimal solutions to the FOM (10), to the ROM (29), and the ROM solution lifted to the FOM state space as

$$\boldsymbol{x}^* = \begin{bmatrix} \boldsymbol{x}_1^{\Omega*} \\ \boldsymbol{x}_1^{\Gamma*} \\ \vdots \\ \boldsymbol{x}_{n_\Omega}^{\Omega*} \\ \boldsymbol{x}_{n_\Omega}^{\Gamma*} \end{bmatrix} \in \mathbb{R}^{N_D}, \qquad \widehat{\boldsymbol{x}}^* = \begin{bmatrix} \widehat{\boldsymbol{x}}_1^{\Omega*} \\ \widehat{\boldsymbol{x}}_1^{\Gamma*} \\ \vdots \\ \widehat{\boldsymbol{x}}_{n_\Omega}^{\Omega*} \\ \widehat{\boldsymbol{x}}_{n_\Omega}^{\Gamma*} \end{bmatrix} \in \mathbb{R}^{n_D}, \qquad \boldsymbol{g}(\widehat{\boldsymbol{x}}^*) = \begin{bmatrix} \boldsymbol{g}_1^{\Omega}(\widehat{\boldsymbol{x}}_1^{\Omega*}) \\ \boldsymbol{g}_1^{\Gamma}(\widehat{\boldsymbol{x}}_1^{\Gamma*}) \\ \vdots \\ \boldsymbol{g}_{n_\Omega}^{\Omega}(\widehat{\boldsymbol{x}}_{n_\Omega}^{\Omega*}) \\ \boldsymbol{g}_{n_\Omega}^{\Gamma}(\widehat{\boldsymbol{x}}_{n_\Omega}^{\Gamma*}) \end{bmatrix} \in \mathbb{R}^{N_D}, \tag{46}$$

respectively, where $N_D = \sum_{i=1}^{n_\Omega}(N_i^\Omega + N_i^\Gamma)$ and, as before, $n_D = \sum_{i=1}^{n_\Omega}(n_i^\Omega + n_i^\Gamma)$. We also define the FOM constraint matrix

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{A}_1 & \dots & \boldsymbol{0} & \boldsymbol{A}_{n_\Omega} \end{bmatrix} \in \mathbb{R}^{N_A\times N_D} \tag{47}$$

so that the constraints (10b) can be written as $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$. As in Section 4, we define the constraint functions $\widetilde{\boldsymbol{A}}_i : \mathbb{R}^{n_i^\Gamma} \to \mathbb{R}^{n_A}$, where $\widetilde{\boldsymbol{A}}_i(\widehat{\boldsymbol{x}}_i^\Omega) = \boldsymbol{C}\boldsymbol{A}_i\boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma)$ in the WFPC case (13) and $\widetilde{\boldsymbol{A}}_i(\widehat{\boldsymbol{x}}_i^\Gamma) = \widehat{\boldsymbol{A}}_i\widehat{\boldsymbol{x}}_i^\Gamma$ in the SRPC case (23), so that the DD-LSPG-ROMs (13) and (23) can be written as (29). We define the ROM constraint function $\widetilde{\boldsymbol{A}} : \mathbb{R}^{n_D} \to \mathbb{R}^{n_A}$ as

$$\widetilde{\boldsymbol{A}}(\widehat{\boldsymbol{x}}) = \sum_{i=1}^{n_\Omega} \widetilde{\boldsymbol{A}}_i(\widehat{\boldsymbol{x}}_i^\Gamma), \tag{48}$$

so that the constraints (29b) can be written as $\widetilde{\boldsymbol{A}}(\widehat{\boldsymbol{x}}) = \mathbf{0}$. Lastly we define the feasible set

$$\mathcal{S}_{\mathrm{ROM}} = \left\{ \widehat{\boldsymbol{x}} \in \mathbb{R}^{n_D} \ : \ \widetilde{\boldsymbol{A}}(\widehat{\boldsymbol{x}}) = \mathbf{0} \right\} \tag{49}$$

for (29).

The next two results provide basic error bounds between a solution to the FOM (9) and solutions to the DD-LSPG-ROM (13) or (23).

Theorem 5 (*A posteriori* error bound)**.** *Let $\boldsymbol{x}^* \in \mathbb{R}^{N_D}$ be a solution to the FOM (9) and let $\widehat{\boldsymbol{x}}^* \in \mathbb{R}^{n_D}$ be a (local) solution to the DD-LSPG-ROM* (13) *or* (23)*. If the residual is inverse Lipschitz continuous, that is, if there exists $\kappa_\ell > 0$ such that*

$$\left(\sum_{i=1}^{n_\Omega} \left\| \boldsymbol{r}_i(\boldsymbol{y}_i^\Omega, \boldsymbol{y}_i^\Gamma) - \boldsymbol{r}_i(\boldsymbol{z}_i^\Omega, \boldsymbol{z}_i^\Gamma) \right\|_2^2 \right)^{1/2} \geq \kappa_\ell \left\| \boldsymbol{y} - \boldsymbol{z} \right\|_2 \qquad \forall\, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^{N_D}, \tag{50a}$$

and if there exists $P > 0$ such that

$$\left(\sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i(\boldsymbol{y}_i^\Omega, \boldsymbol{y}_i^\Gamma) \right\|_2^2 \right)^{1/2} \geq P \left(\sum_{i=1}^{n_\Omega} \left\| \boldsymbol{r}_i(\boldsymbol{y}_i^\Omega, \boldsymbol{y}_i^\Gamma) \right\|_2^2 \right)^{1/2} \qquad \forall\, \boldsymbol{y} \in \boldsymbol{g}(\mathcal{S}_{\mathrm{ROM}}), \tag{50b}$$

then

$$\left\| \boldsymbol{x}^* - \boldsymbol{g}(\widehat{\boldsymbol{x}}^*) \right\|_2 \leq \frac{1}{P\kappa_\ell} \left(\sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}^{\Omega*}), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}^{\Gamma*})) \right\|_2^2 \right)^{1/2}. \tag{51}$$

Proof. Using (50a) and the fact that $\boldsymbol{x}^* \in \mathbb{R}^{N_D}$ solves the FOM (9) gives

$$\left\| \boldsymbol{x}^* - \boldsymbol{g}(\widehat{\boldsymbol{x}}^*) \right\|_2^2 \leq \frac{1}{\kappa_\ell^2} \sum_{i=1}^{n_\Omega} \left\| \boldsymbol{r}_i(\boldsymbol{x}_i^{\Omega*}, \boldsymbol{x}_i^{\Gamma*}) - \boldsymbol{r}_i(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^{\Omega*}), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^{\Gamma*})) \right\|_2^2 \leq \frac{1}{\kappa_\ell^2} \sum_{i=1}^{n_\Omega} \left\| \boldsymbol{r}_i(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^{\Omega*}), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^{\Gamma*})) \right\|_2^2.$$

Applying (50b) with $(\boldsymbol{y}_i^\Omega, \boldsymbol{y}_i^\Gamma) = (\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^{\Omega*}), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^{\Gamma*}))$ gives the desired result. □

Theorem 6 (*A priori* error bound)**.** *Let $\boldsymbol{x}^* \in \mathbb{R}^{N_D}$ be a solution to the FOM (9) and let $\widehat{\boldsymbol{x}}^* \in \mathbb{R}^{n_D}$ be a solution to the DD-LSPG-ROM* (13) *or* (23)*. If the inequalities (50a, b) hold and the HR residual is Lipschitz continuous, i.e., there exists $\kappa_u > 0$ such that*

$$\left(\sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i(\boldsymbol{y}_i^\Omega, \boldsymbol{y}_i^\Gamma) - \boldsymbol{B}_i \boldsymbol{r}_i(\boldsymbol{z}_i^\Omega, \boldsymbol{z}_i^\Gamma) \right\|_2^2 \right)^{1/2} \leq \kappa_u \left\| \boldsymbol{y} - \boldsymbol{z} \right\|_2 \qquad \forall\, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^{N_D}, \tag{52}$$

then

$$\left\| \boldsymbol{x}^* - \boldsymbol{g}(\widehat{\boldsymbol{x}}^*) \right\|_2 \leq \frac{\kappa_u}{P\kappa_\ell} \inf_{\widehat{\boldsymbol{w}} \in \mathcal{S}_{\mathrm{ROM}}} \left\| \boldsymbol{x}^* - \boldsymbol{g}(\widehat{\boldsymbol{w}}) \right\|_2. \tag{53}$$

Proof. Since $\widehat{\boldsymbol{x}}^* \in \mathbb{R}^{n_D}$ is a solution to the DD-LSPG-ROM (13) or (23), any feasible $\widehat{\boldsymbol{w}}$, i.e., any $\widehat{\boldsymbol{w}} \in \mathcal{S}_{\mathrm{ROM}}$ satisfies

$$\sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^{\Omega*}), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^{\Gamma*})) \right\|_2^2 \leq \sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{w}}_i^{\Omega*}), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{w}}_i^{\Gamma*})) \right\|_2^2. \tag{54}$$

Moreover, since $\boldsymbol{x}^* \in \mathbb{R}^{N_D}$ solves the FOM (9), $\boldsymbol{r}_i(\boldsymbol{x}_i^{\Omega *}, \boldsymbol{x}_i^{\Gamma *}) = \boldsymbol{0}$, for all $i = 1, \ldots, n_\Omega$, (52) and (54) imply

$$\sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^{\Omega *}), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^{\Gamma *})) \right\|_2^2 \leq \sum_{i=1}^{n_\Omega} \left\| \boldsymbol{B}_i \boldsymbol{r}_i(\boldsymbol{x}_i^{\Omega *}, \boldsymbol{x}_i^{\Gamma *}) - \boldsymbol{B}_i \boldsymbol{r}_i(\boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{w}}_i^\Omega), \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{w}}_i^\Gamma)) \right\|_2^2$$
$$\leq \kappa_u \left\| \boldsymbol{x}^* - \boldsymbol{g}(\widehat{\boldsymbol{w}}) \right\|_2 \qquad \text{for all } \widehat{\boldsymbol{w}} \in \mathcal{S}_{\mathrm{ROM}}.$$

Combining this result with the a-posterior bound (51) in Theorem 5 yields $\left\| \boldsymbol{x}^* - \boldsymbol{g}(\widehat{\boldsymbol{x}}^*) \right\|_2 \leq \frac{\kappa_u}{P\kappa_\ell} \left\| \boldsymbol{x}^* - \boldsymbol{g}(\widehat{\boldsymbol{w}}) \right\|_2$ for all $\widehat{\boldsymbol{w}} \in \mathcal{S}_{\mathrm{ROM}}$, which implies (53). $\square$

Remark 6. *As a consequence of Theorem 6, if (50a, b) and* (52) *hold, and if $\boldsymbol{x}^*$ is in the image of the $\boldsymbol{g}$ over the feasible set $\mathcal{S}_{\mathrm{ROM}}$ of (13), i.e. if $\boldsymbol{x}^* \in \boldsymbol{g}(\mathcal{S}_{\mathrm{ROM}})$, then $\boldsymbol{x}^* = \boldsymbol{g}(\widehat{\boldsymbol{x}}^*)$.*

The error bounds in Theorems 5 and 6 only involve the FOM and ROM states, but not the Lagrange multipliers. However, the present error bounds require stronger assumptions such as (50a). Alternatively, one could try to extend the error analysis for ROMs applied to nonlinear systems. such as those in [22, Sec. 11.5], [72]. In the context of the FOM (9) and the DD-LSPG-ROM (13) or (23) the role of the nonlinear residual in [22, Sec. 11.5], [72] would now be played by the system of first order necessary optimality conditions, given by (31) for the general ROM formulation (29) and correspondingly for the FOM (9). However, these residuals involve the FOM states and Lagrange multipliers associated with (9b), and ROM states and Lagrange multipliers associated with (29b). Moreover, this analysis requires to relate the ROM states with the FOM states and the ROM Lagrange multipliers with the FOM Lagrange multipliers. The former is done via $\boldsymbol{g}(\widehat{\boldsymbol{x}}) \approx \boldsymbol{x}^*$. However, the connection between the Lagrange multipliers in the general case is still open. For linear PDEs and a PDE-based (as opposed to our algebraic) DD formulation [10] construct appropriate, so called trace-compatible reduced bases for the Lagrange multipliers (see e.g., [10, Eq. (14)]). In the context of [10], the reduced bases for the Lagrange multipliers determine the ROM constraints (29b). In our setting we first derive ROM constraints (29b), which yield ROM Lagrange multipliers, but no explicit construction of a reduced bases for these ROM Lagrange multipliers. This is subject of future research.

7. Numerics

We apply LS-ROM and NM-ROM with and without HR to the DD ROM with WFPC (13) and with SRPC (23) for the 2D steady-state Burgers equation. We use the following formula for computing the relative error between the FOM and ROM solutions:

$$e = \left(\frac{1}{n_\Omega} \sum_{i=1}^{n_\Omega} \frac{\left\| \boldsymbol{x}_i^\Omega - \boldsymbol{g}_i^\Omega(\widehat{\boldsymbol{x}}_i^\Omega) \right\|_2^2 + \left\| \boldsymbol{x}_i^\Gamma - \boldsymbol{g}_i^\Gamma(\widehat{\boldsymbol{x}}_i^\Gamma) \right\|_2^2}{\left\| \boldsymbol{x}_i^\Omega \right\|_2^2 + \left\| \boldsymbol{x}_i^\Gamma \right\|_2^2} \right)^{1/2}. \tag{55}$$

The autoencoder training and subsequent computations in this section were performed on the Lassen machine at Lawrence Livermore National Laboratory, which consists of an IBM Power9 processor with NVIDIA V100 (Volta) GPUs, clock speed between 2.3-3.8 GHz, and 256 GB DDR4 memory.

The implementation of the DD FOM, DD LS-ROM, and DD NM-ROM is done sequentially. However, to highlight potential advantages of a parallel implementation, the recorded wall clock time for the computation of the subdomain-specific quantities required by the SQP solver is taken to be the largest wall clock time incurred among all subdomains. The wall clock time for the remaining steps of the SQP solver (e.g. assembling and solving the KKT system (33), updating the interior- and interface-states and Lagrange multipliers (38), etc.) is set to the overall wall clock time to execute the steps.

7.1. 2D Burgers' Equation

In this experiment, we consider the 2D steady-state Burgers' equation on the domain $[-1,1]\times[0,0.05]$

$$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right),\qquad (x,y)\in[-1,1]\times[0,0.05], \tag{56a}$$

$$u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right),\qquad (x,y)\in[-1,1]\times[0,0.05], \tag{56b}$$

where $\nu>0$ is the viscosity. As in [16], we consider the following exact solution, and its restriction to the boundary as Dirichlet boundary conditions:

$$u_{ex}(x,y;a,\lambda)=-2\nu\left[a+\lambda\left(e^{\lambda(x-1)}-e^{-\lambda(x-1)}\right)\cos(\lambda y)\right]/\psi(x,y;a,\lambda), \tag{57a}$$

$$v_{ex}(x,y;a,\lambda)=2\nu\left[\lambda\left(e^{\lambda(x-1)}+e^{-\lambda(x-1)}\right)\sin(\lambda y)\right]/\psi(x,y;a,\lambda), \tag{57b}$$

where

$$\psi(x,y;a,\lambda)=a(1+x)+\left(e^{\lambda(x-1)}+e^{-\lambda(x-1)}\right)\cos(\lambda y), \tag{57c}$$

and where (a,λ) are parameters. The PDE is discretized using centered finite differences with n_x+2 uniformly spaced grid points in the x-direction and n_y+2 uniformly spaced grid points in the y-direction, resulting in grid points (x_i,y_j) where

$$\begin{aligned} x_i&=-1+ih_x, & i&=0,\dots,n_x+1,\\ y_j&=jh_y, & j&=0,\dots,n_y+1,\end{aligned}$$

where $h_x=2/(n_x+1)$ and $h_y=0.05/(n_y+1)$. The solutions u,v on the grid points are denoted $u_{ij}\approx u(x_i,y_j)$ and $v_{ij}\approx v(x_i,y_j)$. The PDE is then discretized using centered finite differences for the first and second derivative terms. The fully discretized system is given by

$$\mathbf{0}=\boldsymbol{r}_u(\boldsymbol{u},\boldsymbol{v})=\boldsymbol{u}\odot(\boldsymbol{B}_x\boldsymbol{u}-\boldsymbol{b}_{u,x})+\boldsymbol{v}\odot(\boldsymbol{B}_y\boldsymbol{u}-\boldsymbol{b}_{u,y})+\boldsymbol{C}\boldsymbol{u}+\boldsymbol{c}_u, \tag{58a}$$

$$\mathbf{0}=\boldsymbol{r}_v(\boldsymbol{u},\boldsymbol{v})=\boldsymbol{u}\odot(\boldsymbol{B}_x\boldsymbol{v}-\boldsymbol{b}_{v,x})+\boldsymbol{v}\odot(\boldsymbol{B}_y\boldsymbol{v}-\boldsymbol{b}_{v,y})+\boldsymbol{C}\boldsymbol{v}+\boldsymbol{c}_v, \tag{58b}$$

where $\odot$ represents the Hadamard product, and where

$$\boldsymbol{u}=\begin{bmatrix}\boldsymbol{u}^{[1]}\\ \vdots\\ \boldsymbol{u}^{[n_y]}\end{bmatrix}\in\mathbb{R}^{n_xn_y},\qquad \boldsymbol{u}^{[j]}=\begin{bmatrix}u_{1,j}\\ \vdots\\ u_{n_x,j}\end{bmatrix}\in\mathbb{R}^{n_x},\quad j=1,\dots,n_y, \tag{59a}$$

$$\boldsymbol{v}=\begin{bmatrix}\boldsymbol{v}^{[1]}\\ \vdots\\ \boldsymbol{v}^{[n_y]}\end{bmatrix}\in\mathbb{R}^{n_xn_y},\qquad \boldsymbol{v}^{[j]}=\begin{bmatrix}v_{1,j}\\ \vdots\\ v_{n_x,j}\end{bmatrix}\in\mathbb{R}^{n_x},\quad j=1,\dots,n_y, \tag{59b}$$

$$\boldsymbol{B}_x = -\frac{1}{2h_x}\left(\boldsymbol{I}_{n_y} \otimes \widetilde{\boldsymbol{B}}_x\right) \in \mathbb{R}^{n_x n_y \times n_x n_y}, \qquad \widetilde{\boldsymbol{B}}_x = \begin{bmatrix} 0 & 1 & \\ -1 & \ddots & 1 \\ & -1 & 0 \end{bmatrix} \in \mathbb{R}^{n_x \times n_x}, \tag{59c}$$

$$\boldsymbol{B}_y = -\frac{1}{2h_y}\left(\widetilde{\boldsymbol{B}}_y \otimes \boldsymbol{I}_{n_x}\right) \in \mathbb{R}^{n_x n_y \times n_x n_y} \qquad \widetilde{\boldsymbol{B}}_y = \begin{bmatrix} 0 & 1 & \\ -1 & \ddots & 1 \\ & -1 & 0 \end{bmatrix} \in \mathbb{R}^{n_y \times n_y}, \tag{59d}$$

$$\boldsymbol{C} = \frac{\nu}{h_x^2}\left(\boldsymbol{I}_{n_y} \otimes \widetilde{\boldsymbol{C}}_x\right) + \frac{\nu}{h_y^2}\left(\widetilde{\boldsymbol{C}}_y \otimes \boldsymbol{I}_{n_x}\right) \in \mathbb{R}^{n_x n_y \times n_x n_y}, \tag{59e}$$

$$\widetilde{\boldsymbol{C}}_x = \begin{bmatrix} -2 & 1 & \\ 1 & \ddots & \\ & 1 & -2 \end{bmatrix} \in \mathbb{R}^{n_x \times n_x}, \qquad \widetilde{\boldsymbol{C}}_y = \begin{bmatrix} -2 & 1 & \\ 1 & \ddots & \\ & 1 & -2 \end{bmatrix} \in \mathbb{R}^{n_y \times n_y}, \tag{59f}$$

$$\boldsymbol{b}_{u,x} = -\frac{1}{2h_x}(\boldsymbol{b}_{ux\ell} - \boldsymbol{b}_{uxr}), \qquad \boldsymbol{b}_{u,y} = -\frac{1}{2h_y}(\boldsymbol{b}_{uy\ell} - \boldsymbol{b}_{uyr}), \tag{59g}$$

$$\boldsymbol{c}_u = \frac{\nu}{h_x^2}(\boldsymbol{b}_{ux\ell} + \boldsymbol{b}_{uxr}) + \frac{\nu}{h_y^2}(\boldsymbol{b}_{uy\ell} + \boldsymbol{b}_{uyr}) \tag{59h}$$

$$\boldsymbol{b}_{v,x} = -\frac{1}{2h_x}(\boldsymbol{b}_{vx\ell} - \boldsymbol{b}_{vxr}), \qquad \boldsymbol{b}_{v,y} = -\frac{1}{2h_y}(\boldsymbol{b}_{vy\ell} - \boldsymbol{b}_{vyr}), \tag{59i}$$

$$\boldsymbol{c}_v = \frac{\nu}{h_x^2}(\boldsymbol{b}_{vx\ell} + \boldsymbol{b}_{vxr}) + \frac{\nu}{h_y^2}(\boldsymbol{b}_{vy\ell} + \boldsymbol{b}_{vyr}) \tag{59j}$$

$$\boldsymbol{b}_{ux\ell} = \begin{bmatrix} u_{ex}(x_0, y_1) \\ \vdots \\ u_{ex}(x_0, y_{n_y}) \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n_x \times 1} \in \mathbb{R}^{n_x n_y}, \qquad \boldsymbol{b}_{uxr} = \begin{bmatrix} u_{ex}(x_{n_x+1}, y_1) \\ \vdots \\ u_{ex}(x_{n_x+1}, y_{n_y}) \end{bmatrix} \otimes \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}_{n_x \times 1} \in \mathbb{R}^{n_x n_y}, \tag{59k}$$

$$\boldsymbol{b}_{uyb} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n_y \times 1} \otimes \begin{bmatrix} u_{ex}(x_1, y_0) \\ \vdots \\ u_{ex}(x_{n_x}, y_0) \end{bmatrix} \in \mathbb{R}^{n_x n_y} \qquad \boldsymbol{b}_{uyt} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}_{n_y \times 1} \otimes \begin{bmatrix} u_{ex}(x_1, y_{n_y+1}) \\ \vdots \\ u_{ex}(x_{n_x}, y_{n_y+1}) \end{bmatrix} \in \mathbb{R}^{n_x n_y} \tag{59l}$$

$$\boldsymbol{b}_{vx\ell} = \begin{bmatrix} v_{ex}(x_0, y_1) \\ \vdots \\ v_{ex}(x_0, y_{n_y}) \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n_x \times 1} \in \mathbb{R}^{n_x n_y}, \qquad \boldsymbol{b}_{vxr} = \begin{bmatrix} v_{ex}(x_{n_x+1}, y_1) \\ \vdots \\ v_{ex}(x_{n_x+1}, y_{n_y}) \end{bmatrix} \otimes \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}_{n_x \times 1} \in \mathbb{R}^{n_x n_y}, \tag{59m}$$

$$\boldsymbol{b}_{vyb} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n_y \times 1} \otimes \begin{bmatrix} v_{ex}(x_1, y_0) \\ \vdots \\ v_{ex}(x_{n_x}, y_0) \end{bmatrix} \in \mathbb{R}^{n_x n_y} \qquad \boldsymbol{b}_{vyt} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}_{n_y \times 1} \otimes \begin{bmatrix} v_{ex}(x_1, y_{n_y+1}) \\ \vdots \\ v_{ex}(x_{n_x}, y_{n_y+1}) \end{bmatrix} \in \mathbb{R}^{n_x n_y}. \tag{59n}$$

For the monolithic (single domain) FOM, we take $n_x = 480$, $n_y = 24$, viscosity $\nu = 0.1$, and parameters $(a, \lambda) \in \mathcal{D} = [1, 10^4] \times [5, 25]$. The parameter a corresponds to the distance of the shock from the left boundary, whereas λ corresponds to the steepness of the shock, as illustrated in Fig. 4. We use the ROMs to predict the case where $(a, \lambda) = (7692.5384, 21.9230)$. The SQP solver for the DD FOM, DD LS-ROM, and DD NM-ROM terminates when the 2-norm of the right hand side of (33) is less than 10^{-4}, or after 15 iterations.

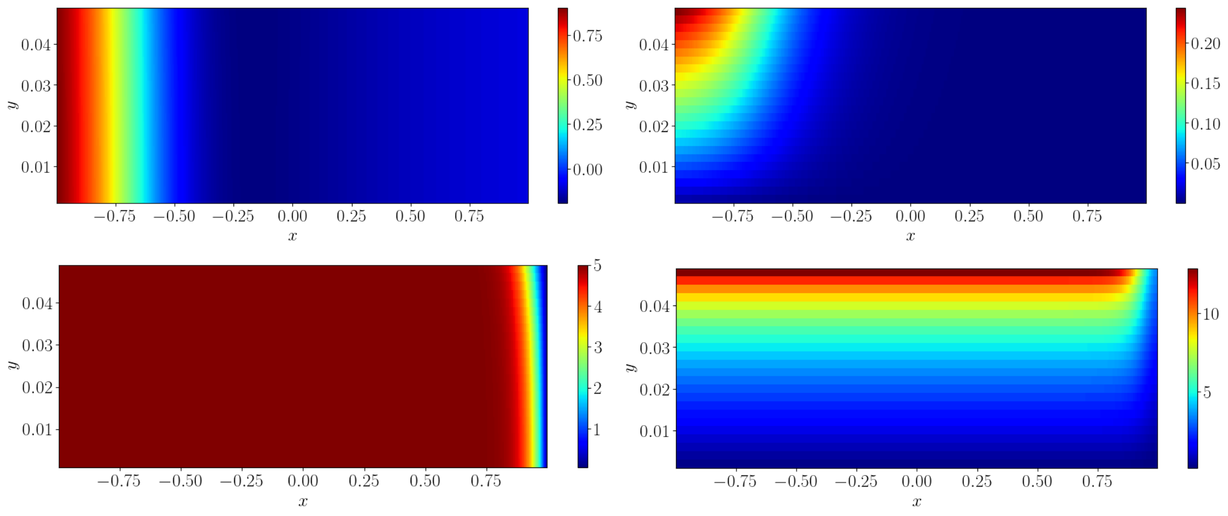


Figure 4: Top left: u-component with $(a, \lambda) = (10^4, 5)$; Top right: v-component with $(a, \lambda) = (10^4, 5)$; Bottom left: u-component with $(a, \lambda) = (1, 25)$; Bottom right: v-component with $(a, \lambda) = (1, 25)$

7.2. *Snapshot data collection*

To compute ROMs, we first collect 6400 snapshots for training with parameters (a, λ) uniformly sampled in a 80×80 grid for the full-domain problem. These full-domain snapshots are then restricted to the interior, interface, and port states, which are then used for training. This is the so-called "top-down" approach. The residual bases $\boldsymbol{\Phi}_i^r$ for each subdomain are computed by taking the Newton iteration history for 400 state snapshots sampled on a 20×20 (a, λ) grid, and computing a POD basis with energy criterion $\nu = 10^{-10}$. These 400 state snapshots are then used to train RBF interpolator models (using Scipy's `RBFInterpolator` function) for each subdomain's interior and interface states, which are then used to compute an initial iterate for $(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma)$ for the SQP solver. The $(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma)$ initial iterates are then used to compute an initial iterate for the Lagrange multipliers $\boldsymbol{\lambda}$ by applying a least-squares solver to equation (32b). The wall clock time to compute the initial iterates $(\widehat{\boldsymbol{x}}_i^\Omega, \widehat{\boldsymbol{x}}_i^\Gamma)$ is taken to be the largest wall clock time incurred among all subdomains, while the wall clock time to compute the initial iterate for $\boldsymbol{\lambda}$ is the time required to sequentially solve the least-squares problem (32b). The wall clock time to compute the initial iterate for NM-ROM using the RBF interpolator is included in the computation times and speedups reported.

7.3. *Autoencoder training*

For NM-ROM, we randomly split the state snapshots into 5760 training snapshots and 640 testing snapshots. For training the autoencoders, we use the MSE loss, the Adam optimizer over 2000 epochs, and a batch size of 32. We also normalize the snapshots so that all snapshot components are in $[-1, 1]$, and apply a de-normalization layer to the output of the autoencoder. We apply early stopping with a stopping patience of 300, and reduce the learning rate on plateau with an initial learning rate of 10^{-3} and a patience of 50. The implementation was done using PyTorch, as well as the Pytorch Sparse and SparseLinear packages.

The sparsity masks used for the output layers of the decoders have a banded structure inspired by 2D finite difference stencils. Each row has three bands, where each band consists of contiguous nonzero entries, and where the band shifts to the right a specified amount from one row to the next. The number of nonzero entries per band and the number of columns the band shifts over are hyper-parameters The separation between the bands in each row is equal to the product of the number of nonzeros per band and the column-shift per row. These parameters, as well as the dimension of the interior and interface states, determine the width of the decoders. Figure 5 provides a visualization for the decoder mask used. The transpose of these masks is used at input layer of the encoders.

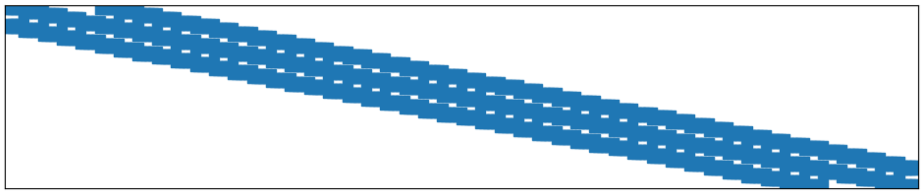


Figure 5: Three-banded sparsity mask for decoder

7.4. Dense vs Sparse Encoder

We first briefly compare the performance of a dense encoder with a sparse encoder. We train two autoencoders, one with a fully dense encoder and one with a sparse encoder, on a coarse, single domain problem with $n_x = 240$ and $n_y = 12$. Both decoders share the same sparse architecture. The data collection, training, and relevant sparsity masks are identical to the procedures discussed in Section 7.3. The discretization results in a FOM size of 5760 and we used a ROM of size 4. Table 1 summarizes the key performance differences between the two encoders. Importantly, the sparse encoder architecture achieves losses 4 orders of magnitude smaller than the dense encoder with 2 orders of magnitude fewer parameters.

	Train loss	Test loss	Width	# encoder parameters
Dense encoder	5.74×10^{-2}	1.56×10^{-1}	17280	9.96×10^{7}
Sparse encoder	7.21×10^{-5}	7.48×10^{-5}	28800	2.3×10^{5}

Table 1: Comparison of dense and sparse encoder architectures.

7.5. Single-domain NM-ROM vs DD NM-ROM

Next we examine the per-subdomain reduction in the required number of autoencoder parameters for different DD configurations compared to the monolithic single-domain NM-ROM. See Table 2. In the single domain case, we solve the LSPG problem

$$\min_{\widehat{\boldsymbol{x}}} \; \|\boldsymbol{B}\boldsymbol{r}\left(\boldsymbol{g}\left(\widehat{\boldsymbol{x}}\right)\right)\|_2^2 \tag{60}$$

using the Gauss-Newton method. The function $\boldsymbol{g} : \mathbb{R}^{n_x} \to \mathbb{R}^{N_x}$ is the decoder of an autoencoder trained on snapshots of the monolithic single-domain FOM as discussed in Sections 3.4, 5, and 7.3, and $\boldsymbol{B} \in \{0,1\}^{N_B \times N_x}$ is a collocation HR matrix, as discussed in Section 5.3.

Subdomains	Max # subdomain params.	Reduction	Total # params.	Error
1×1	2.995×10^{6}	0.0 %	2.995×10^{6}	1.08×10^{-3}
2×1	1.147×10^{6}	61.7 %	2.307×10^{6}	1.27×10^{-3}
2×2	5.257×10^{5}	82.4 %	2.384×10^{6}	2.42×10^{-3}
4×2	2.617×10^{5}	91.3 %	2.391×10^{6}	4.26×10^{-3}
8×2	1.297×10^{5}	95.7 %	2.406×10^{6}	4.58×10^{-2}

Table 2: Max number of NN parameters per subdomain, the per-subdomain reduction in number of NN parameters, the total number of parameters, and the corresponding error for different subdomain configurations. For the single-domain case, an NM-ROM of dimension $n = 9$ is used. For the DD cases, $(n_i^{\Omega}, n_i^{\Gamma}) = (6, 3)$, resulting in 9 DoF per subdomain. HR was not used to evaluate the NM-ROMs in these examples.

We use the notation 2×1 subdomains to indicate 2 subdomains in the x-direction and 1 subdomain in the y-direction. As expected, from Table 2, we see that the maximum number of NN parameters per subdomain decreases significantly as more subdomains are used. Furthermore, the total number of NN parameters in the DD cases also decreases relative to the single-domain case. We also note that the error increases as more subdomains are used. We kept $(n_i^\Omega, n_i^\Gamma) = (6, 3)$ constant for each subdomain configuration to isolate the effect of DD on the number of NN parameters, but this may cause overfitting in the 16 subdomain case. More careful hyper-parameter tuning is necessary to mitigate increases in error as the number of subdomains is increased.

7.6. LS-ROM vs NM-ROM comparison

Next we compare the DD LS-ROM and DD NM-ROM. We first focus on a DD configuration with 2 uniformly sized subdomains in the x-direction and 2 uniformly sized subdomains in the y-direction (4 subdomains total) using the WFPC formulation (13). The interior and interface states for the FOM were of dimension $N_i^\Omega = 5238$ and $N_i^\Gamma = 1006$, respectively, resulting in 25056 degrees of freedom (DoF) aggregated across all subdomains. For both the LS-ROM and NM-ROM, we use reduced state dimensions of $n_i^\Omega = 8$ for the interior states $\widehat{\boldsymbol{x}}_i^\Omega$ and $n_i^\Gamma = 4$ for the interface states $\widehat{\boldsymbol{x}}_i^\Gamma$ for each subdomain, resulting in 48 DoF aggregated across all subdomains. In the HR case, $N_i^B = 100$ HR nodes are used for each subdomain, resulting in 400 total HR nodes aggregated all subdomains.

Each interior-state autoencoder has input/output dimension $N_i^\Omega = 5238$, width $w_i^\Omega = 26290$, latent dimension $n_i^\Omega = 8$, and Swish activation $\boldsymbol{\sigma}_i^\Omega(z) = z/(1 + e^{-z})$. Each interface-state autoencoder has input/output dimension $N_i^\Gamma = 1006$, width $w_i^\Gamma = 5030$, latent dimension $n_i^\Gamma = 4$, and Swish activation. The number of nonzeros per row and column-shift were both set to 5 for both the interior and interface state decoders. The number of nonzeros for the interior-states masks is 78820, resulting in 99.94% sparsity, while the number of nonzeros for the interface-states masks is 15040, resulting in 99.70% sparsity.

Figure 6 shows the FOM, LS-ROM, and NM-ROM solutions without HR, and Figure 8 shows the solutions with collocation HR using 48 DoF in both cases. In both the HR and non-HR cases with the same DoF, NM-ROM achieves an order of magnitude lower relative error than LS-ROM, as evidenced in Figures 7 and 9 and Table 3. Without HR, NM-ROM achieves a relative error of 1.28×10^{-3} while LS-ROM achieves a relative error 1.98×10^{-2} using the same number of DoF. LS-ROM also achieves a speedup of 30.0, whereas NM-ROM achieves a 21.7 times speedup. With HR, NM-ROM achieves a relative error of 1.64×10^{-3} while LS-ROM achieves a relative error 1.44×10^{-2} using the same number of DoF. In the HR case, LS-ROM achieves a speedup of 347.6, whereas NM-ROM achieves a 43.9 speedup.

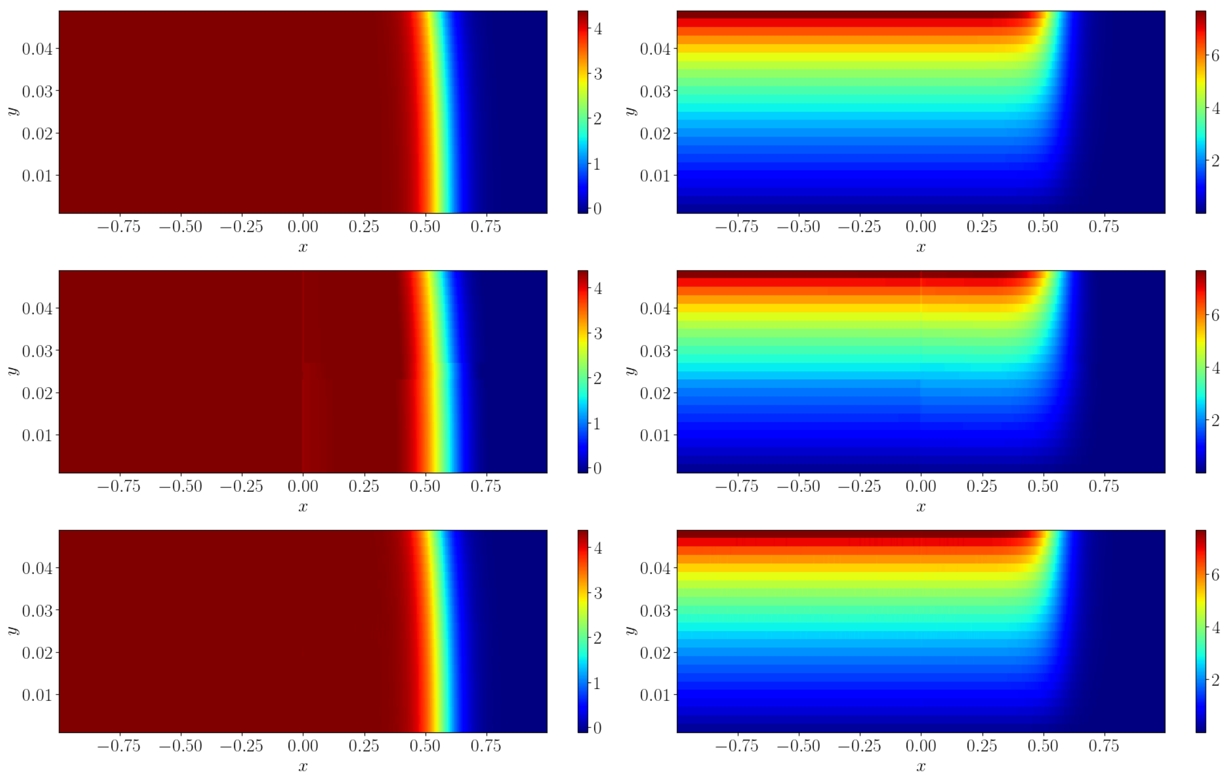


Figure 6: Top left: u-component of FOM; Top right: v-component of FOM; Middle left: u-component of LS-ROM without HR, WFPC, 48 DoF; Middle right: v-component of LS-ROM without HR, WFPC, 48 DoF; Bottom left: u-component of NM-ROM without HR, WFPC, 48 DoF; Bottom right: v-component of NM-ROM without HR, WFPC, 48 DoF.

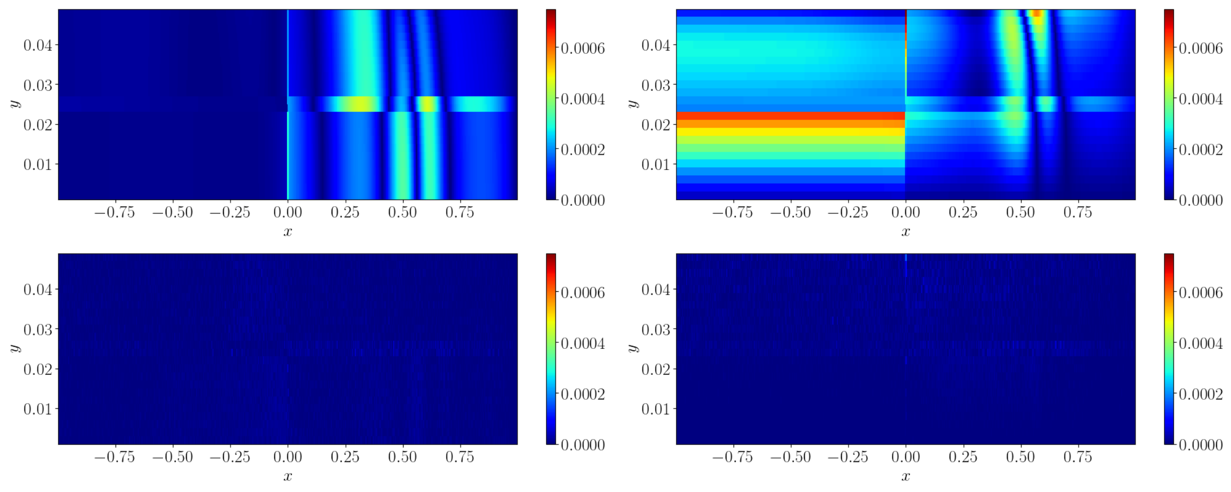


Figure 7: Top left: u relative error for LS-ROM without HR, WFPC, 48 DoF; Top right: v relative error for LS-ROM without HR, WFPC, 48 DoF; Bottom left: u relative error for RM-ROM without HR, WFPC, 48 DoF; Bottom right: v relative error for NM-ROM without HR, WFPC, 48 DoF.

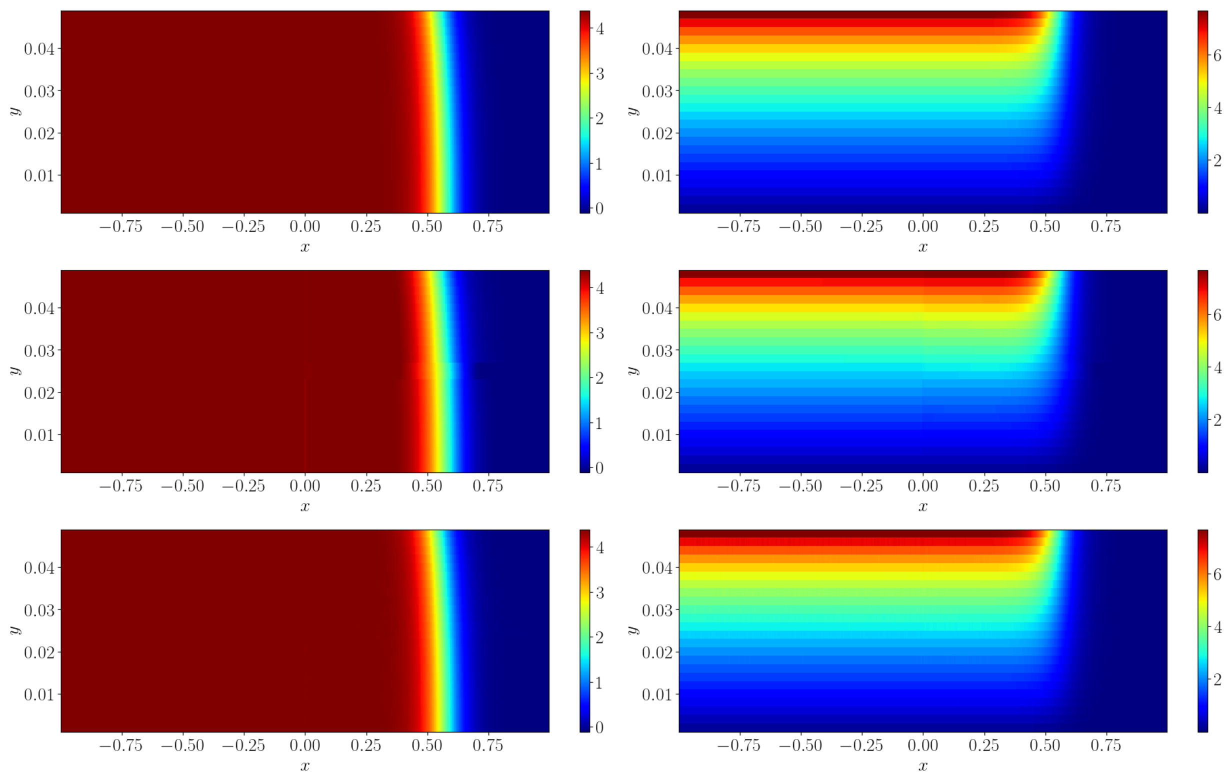


Figure 8: Top left: u-component of FOM; Top right: v-component of FOM; Middle left: u-component of LS-ROM with collocation HR, WFPC, 48 DoF; Middle right: v-component of LS-ROM with collocation HR, WFPC, 48 DoF; Bottom left: u-component of NM-ROM with collocation HR, WFPC, 48 DoF; Bottom right: v-component of NM-ROM with collocation HR, WFPC, 48 DoF.

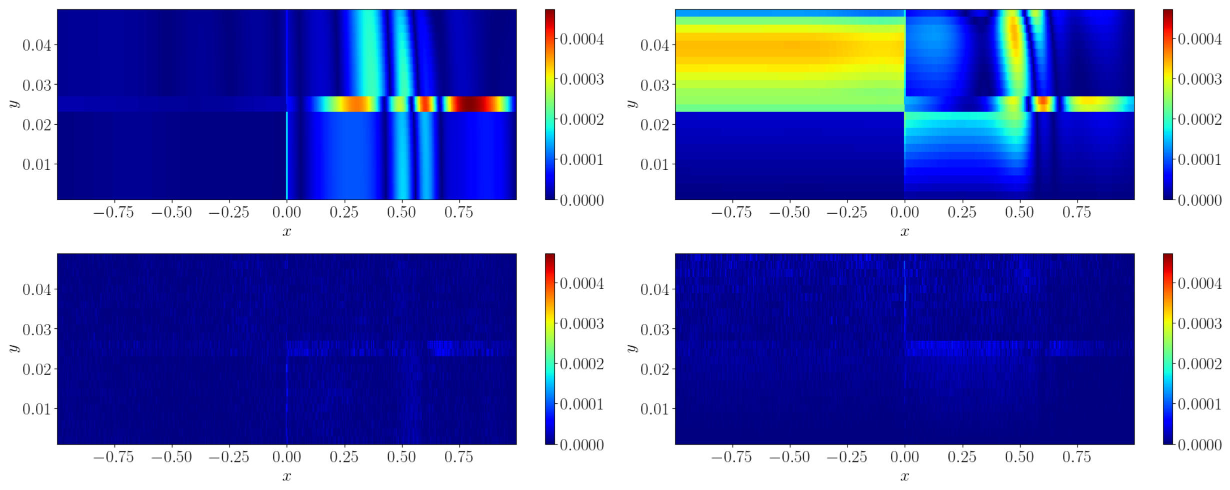


Figure 9: Top left: u relative error for LS-ROM with collocation HR, WFPC, 48 DoF; Top right: v relative error for LS-ROM with collocation HR, WFPC, 48 DoF; Bottom left: u relative error for NM-ROM with collocation HR, WFPC, 48 DoF; Bottom right: v relative error for NM-ROM with collocation HR, WFPC, 48 DoF.

Now we compare the performance of LS-ROM and NM-ROM for both the WFPC and SRPC formulations while varying the interior and interface ROM state dimensions n_i^Ω and n_i^Γ. For WFPC, the decoders $\boldsymbol{g}_i^\Omega$ and $\boldsymbol{g}_i^\Gamma$ use Swish activations, and their sparsity masks have 5 nonzeros per band and a column-shift of 5 for each test. For SRPC, the interior-state decoders $\boldsymbol{g}_i^\Omega$ are reused from the WFPC formulation, and the port-state decoders $\boldsymbol{g}_j^p$ were chosen to have Sigmoid activation and sparsity masks with 3 nonzero entries per band with column-shift of 3. We also define $\widetilde{n}_j^p$, which determines the port latent dimensions n_j^p via the relation

$$n_j^p = \max\left\{\min\left\{N_j^p - 1,\ \widetilde{n}_j^p\right\}, 1\right\}, \qquad \forall\, j = 1, \ldots, n_p.$$

This rule was chosen to ensure that $1 \leq n_j^p < N_j^p$ because some DD configurations have ports with a very small number of nodes (e.g., $N_j^p = 2$). Recall that for SRPC, the ROM interface-state dimension is $n_i^\Gamma = \sum_{j \in Q(i)} n_j^p$. Table 3 shows the relative error and speedup for LS-ROM and NM-ROM for both WFPC and SRPC on the 2×2 subdomain problem with and without HR while varying n_i^Ω, $\widetilde{n}_j^p$, and n_i^Γ.

From Table 3, we see that NM-ROM consistently achieves an order of magnitude lower error than LS-ROM both with and without HR when comparing ROMs of the same dimensions and constraint formulations. In the non-HR case with WFPC, LS-ROM only achieves an order 10^{-3} error for a ROM with 96 total DoF (rel. error = 2.66×10^{-3}), while NM-ROM can achieve a similar error with only 36 DoF (rel. error = 2.42×10^{-3}) and a higher speedup (speedup = 26.2) compared to LS-ROM with similar accuracy (speedup = 18.3). For SRPC, LS-ROM was only able to achieve order 10^{-2} accuracy for all cases tested. We also see that LS-ROM achieves a much higher speedup in the HR cases while retaining similar errors from the non-HR cases. NM-ROM also retains high accuracy after HR, and gains an extra 15-20$\times$ speedup after applying HR.

	Constraints	n_i^Ω	$\widetilde{n}_j^p$	n_i^Γ	Total DoF	Error	Speedup	Error (HR)	Speedup (HR)
LS-ROM	WFPC	4	-	2	24	4.12×10^{-2}	32.1	3.45×10^{-2}	352.7
		6	-	3	36	2.06×10^{-2}	48.7	1.78×10^{-2}	340.0
		8	-	4	48	1.98×10^{-2}	30.0	1.44×10^{-2}	347.6
		10	-	5	60	1.50×10^{-2}	16.3	1.16×10^{-2}	329.6
		16	-	8	96	2.66×10^{-3}	18.3	3.23×10^{-3}	280.4
	SRPC	6	1	5	44	5.17×10^{-2}	24.5	5.12×10^{-2}	315.6
		8	2	7	60	3.75×10^{-2}	22.0	4.22×10^{-2}	313.9
		10	3	9	76	1.87×10^{-2}	19.4	2.94×10^{-2}	262.6
		16	4	11	108	2.00×10^{-2}	17.8	3.37×10^{-2}	253.8
NM-ROM	WFPC	4	-	2	24	6.94×10^{-3}	22.7	7.04×10^{-3}	37.4
		6	-	3	36	2.42×10^{-3}	26.2	2.60×10^{-3}	44.7
		8	-	4	48	1.28×10^{-3}	21.7	1.64×10^{-3}	43.9
		10	-	5	60	1.09×10^{-3}	15.0	1.19×10^{-3}	43.6
		16	-	8	96	7.87×10^{-4}	13.9	9.80×10^{-4}	37.5
	SRPC	6	1	5	44	2.75×10^{-2}	27.6	3.17×10^{-2}	41.1
		8	2	7	60	1.19×10^{-3}	28.8	1.70×10^{-3}	42.6
		10	3	9	76	1.46×10^{-3}	17.0	2.54×10^{-3}	43.1
		16	4	11	108	1.11×10^{-3}	16.8	2.45×10^{-3}	47.1

Table 3: Relative error and speedup for LS-ROM and NM-ROM with and without HR applied to the WFPC and SRPC formulations. We use $N_i^B = 100$ HR nodes per subdomain in the HR case.

Next we examine the effect of subdomain configuration on the accuracy of LS-ROM and NM-ROM. Again let n_Ω^x and n_Ω^y denote the number of subdomains in the x- and y-directions, respectively. In each case, the subdomains are of uniform size. For the WFPC cases, we used $(n_i^\Omega, n_i^\Gamma) = (8, 4)$ for each subdomain, and for the SRPC cases, we used $(n_i^\Omega, \widetilde{n}_j^p) = (8, 2)$ for each subdomain and port, respectively. For the HR cases, we used $N_i^B = 100$ HR nodes. The results for the non-HR and HR cases are reported in Table 4.

Table 4 shows that LS-ROM is more sensitive to subdomain configuration than NM-ROM in the WFPC cases. Indeed, when using 2 subdomains in the y-direction, the relative error for LS-ROM increases to the order of 10^{-2}, compared to errors on the order of 10^{-3} when only 1 subdomain is used in the y-direction. In contrast, relative error for NM-ROM with WFPC is more stable with respect to subdomain configuration. The relative errors are on the order of 10^{-3} for all configuration considered. For SRPC, LS-ROM only achieves order 10^{-3} relative error for the $(n_\Omega^x, n_\Omega^y) = (2,1)$ configuration, while the remaining configurations have order 10^{-2} relative error. For NM-ROM with SRPC, all configurations have order 10^{-3} relative error except for $(n_\Omega^x, n_\Omega^y) = (4,2)$, which attains order 10^{-2} relative error. For both WFPC and SRPC, the NM-ROM error increases slightly as more subdomains are used, but we expect this error can be decreased by adjusting hyper-parameters during ROM training. Hyper-parameter tuning was only done for the 2×2 subdomain configuration.

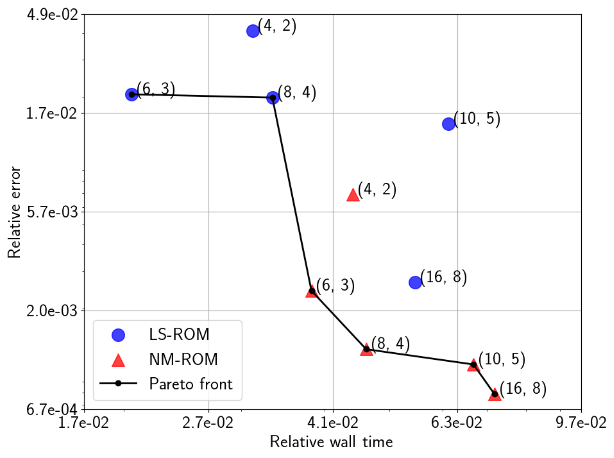
	Constraints	n_Ω^x	n_Ω^y	# subdomains	Error	Speedup	Error (HR)	Speedup (HR)
LS-ROM	WFPC	2	1	2	6.36×10^{-3}	25.1	6.64×10^{-3}	285.7
		2	2	4	1.98×10^{-2}	30.0	1.44×10^{-2}	347.6
		4	1	4	7.34×10^{-3}	37.1	7.47×10^{-3}	373.1
		4	2	8	2.29×10^{-2}	35.8	4.21×10^{-2}	259.2
	SRPC	2	1	2	6.85×10^{-3}	30.5	9.49×10^{-3}	293.0
		2	2	4	3.75×10^{-2}	22.0	4.22×10^{-2}	313.9
		4	1	4	1.04×10^{-2}	24.6	5.94×10^{-2}	287.5
		4	2	8	4.96×10^{-2}	12.2	5.19×10^{-2}	181.6
NM-ROM	WFPC	2	1	2	1.34×10^{-3}	16.8	1.36×10^{-3}	30.5
		2	2	4	1.28×10^{-3}	21.7	1.64×10^{-3}	43.9
		4	1	4	3.14×10^{-3}	27.8	4.98×10^{-3}	38.6
		4	2	8	4.82×10^{-3}	26.3	5.98×10^{-3}	40.4
	SRPC	2	1	2	1.00×10^{-3}	17.0	1.37×10^{-3}	35.9
		2	2	4	1.19×10^{-3}	28.8	1.70×10^{-3}	42.6
		4	1	4	1.68×10^{-3}	27.4	2.12×10^{-3}	39.1
		4	2	8	1.67×10^{-2}	24.2	2.39×10^{-2}	32.5

Table 4: Relative error and speedup for LS-ROM and NM-ROM with and without HR and different subdomain configurations for the WFPC and SRPC formulations. We use $n_i^\Omega = 8$ for all cases, $n_i^\Gamma = 4$ for the WFPC cases, $\widetilde{n}_j^p = 2$ for the SRPC cases, and $N_i^B = 100$ for the HR cases.

Tables 3 and 4 show that SRPC has slightly worse performance than WFPC. In particular, Table 3 shows that the relative errors for both LS-ROM and NM-ROM with SRPC do not decrease monotonically as n_i^Ω and $\widetilde{n}_j^p$ increase. In contrast, the relative errors do decrease monotonically as n_i^Ω and n_i^Γ increase for both LS-ROM and NM-ROM with WFPC. Furthermore, Table 4 shows that LS-ROM with SRPC consistently has larger errors than with WFPC for each subdomain configuration. For NM-ROM, the relative errors and speedups are similar between WFPC and SRPC for each subdomain configuration except for $(n_\Omega^x, n_\Omega^y) = (4,2)$, which has an order of magnitude higher error than the other configurations. Since SRPC performs as well or worse compared to WFPC for the cases tested, we only consider WFPC in the remainder of this section.

7.7. Pareto fronts

Next we compute Pareto fronts to compare LS-ROM and NM-ROM with WFPC while varying different parameters. The relative error reported is the same as defined in equation (55). Figure 10 shows the Pareto fronts for varying (n_i^Ω, n_i^Γ) for both the non-HR and HR cases. In the HR case, we use $N_i^B = 100$ for each subdomain. We see that LS-ROM wins in terms of relative wall time, while NM-ROM attains an order of magnitude lower error in comparison to LS-ROM in each case.



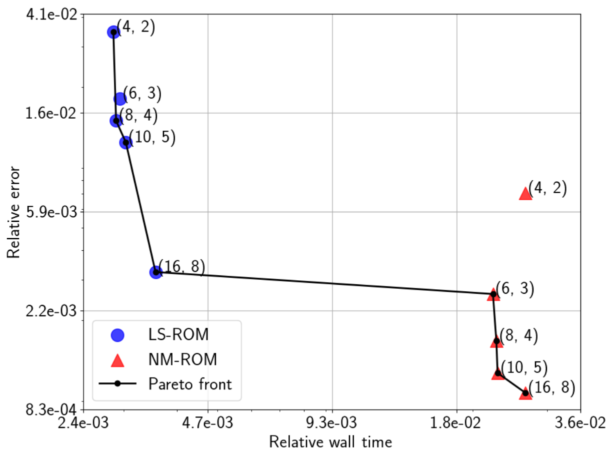


Figure 10: Left: Pareto front for LS-ROM and NM-ROM without HR with varying (n_i^Ω, n_i^Γ) for WFPC formulation; Right: Pareto front for LS-ROM and NM-ROM with varying (n_i^Ω, n_i^Γ) and $N_i^B = 100$ HR nodes per subdomain for WFPC formulation

Figure 11 shows the Pareto front for varying number of HR nodes per subdomain, N_i^B. In this case, $(n_i^\Omega, n_i^\Gamma) = (8, 4)$ for each experiment. As in the case for varying (n_i^Ω, n_i^Γ), NM-ROM attains an order of magnitude lower relative error. Moreover, both the relative error and relative wall time for NM-ROM remains small for each value of N_i^B, whereas the relative error and relative wall time for LS-ROM has more variability with respect to number of HR nodes.

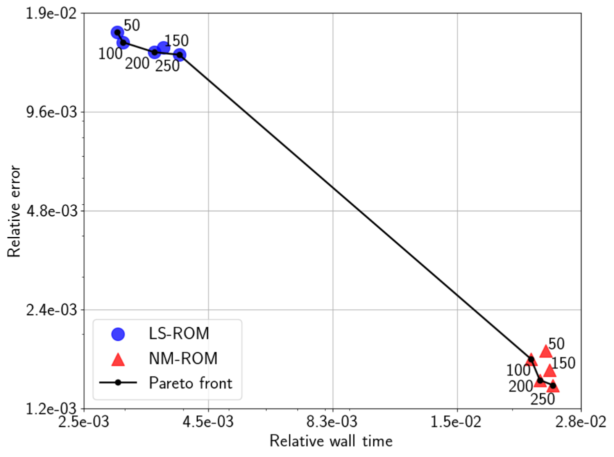
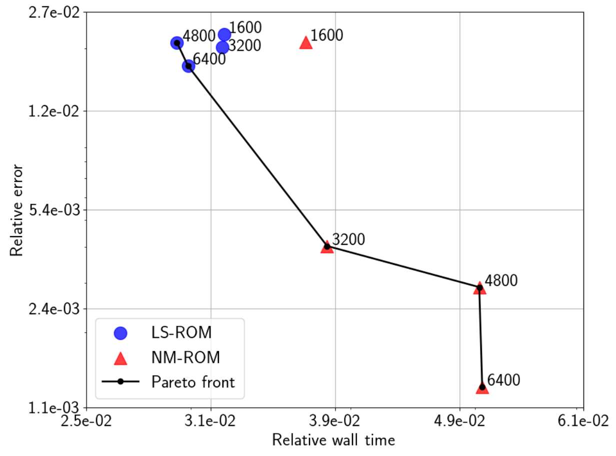


Figure 11: Pareto front for LS-ROM and NM-ROM with $(n_i^\Omega, n_i^\Gamma) = (8, 4)$ and varying number of HR nodes per subdomain N_i^B for WFPC formulation

Figure 12 shows the Pareto fronts for varying number of training snapshots in the non-HR and HR cases. We used $(n_i^\Omega, n_i^\Gamma) = (8, 4)$ for each experiment, and used $N_i^B = 100$ HR nodes in the HR case. For LS-ROM, the whole training set was used to compute the POD bases, whereas for NM-ROM, the displayed number of snapshots underwent a random 90-10 split for training and validation, respectively. In the case of LS-ROM, the relative error remained constant for the number of training snapshots used, whereas the relative error for NM-ROM decreased as more training snapshots were used.



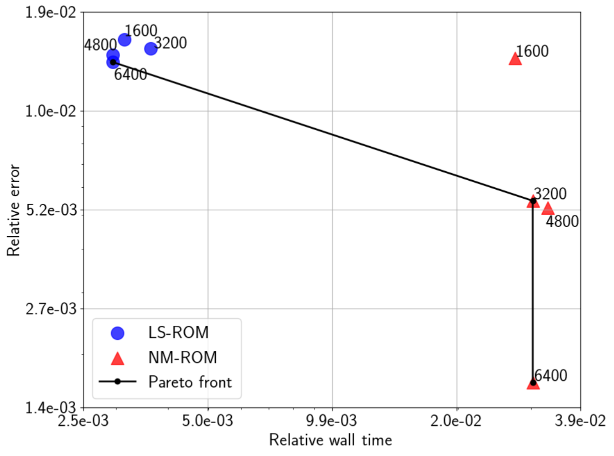


Figure 12: Left: Pareto front for LS-ROM and NM-ROM with $(n_i^\Omega, n_i^\Gamma) = (8, 4)$ and varying number of training snapshots for WFPC formulation; Right: Pareto front for LS-ROM and NM-ROM with $(n_i^\Omega, n_i^\Gamma) = (8, 4)$, $N_i^B = 100$ HR nodes per subdomain, and varying number of training snapshots for WFPC formulation

8. Conclusion

In this work, we detail the first application of NM-ROM with HR to a DD problem. We extend the DD framework of [16] and compute ROMs using the NM-ROM [55] approach on each subdomain. Furthermore, we apply HR to NM-ROM on each subdomain, which informs the use of shallow, sparse autoencoders, as in [56]. We detail how to implement an inexact Lagrange-Newton SQP method to solve the constrained least-squares formulation of the ROM, where the inexactness comes from using a Gauss-Newton approximation of the residual terms, and from neglecting second-order decoder derivative terms coming from the compatibility constraints. We then provide a convergence result for the SQP solver used using standard theory of inexact Newton's method. We also provide *a priori* and *a posteriori* error bounds between the FOM and ROM solutions.

From our numerical experiments on the 2D steady-state Burgers' equation, we showed that using the DD NM-ROM approach significantly decreases the number of required NN parameters per subdomain compared to the monolithic single-domain NM-ROM. We also showed that DD NM-ROM achieves an order of magnitude lower relative error than DD LS-ROM in nearly all cases tested. Furthermore, in the non-HR case, NM-ROM is faster than LS-ROM in some instances. We also saw that NM-ROM is more robust than LS-ROM with respect to changes in subdomain configuration. In some cases, the subdomain configuration increased the LS-ROM relative error by an order of magnitude. While LS-ROM with HR achieves much higher speedup than NM-ROM with HR, NM-ROM is still the clear winner in terms of ROM accuracy for a given ROM size. Moreover, HR allows NM-ROM to gain an extra 15-$20\times$ speedup compared to the non-HR cases. While the speedup is not as drastic as for LS-ROM, these speedup gains for NM-ROM are the highest that have been achieved for NM-ROM to our knowledge. Our results indicate that NM-ROM should be the preferred approach for problems where ROM accuracy for a given ROM size is more important than speedup.

Although NM-ROM performs better than LS-ROM in our experiments, LS-ROM still attains a relatively low relative error. This indicates that the model problem considered may still be too benign to expose the advantages that NM-ROM has over LS-ROM, particularly when applied to problems with slowly decaying Kolmogorov n-width. Therefore, in future work, we plan to apply DD NM-ROM to more complicated problems, including those with slowly decaying Kolmogorov n-width, as well as to time-dependent problems. Furthermore, the speedup of the DD NM-ROM is highly dependent on the SQP solver used. Thus, it will be important to investigate the use of other optimization algorithms for the solution of (13) and examine their effects on computational speedup. Other directions for future research include a greedy sampling strategy for the parameter space $\mathcal{D}$ when choosing which FOM snapshots to compute for NM-ROM training, implementing a "bottom-up" training strategy that uses subdomain snapshots rather than full-domain snapshots for training, applying the DD NM-ROM framework to decomposable or component-based systems, and applying NM-ROM to other DD approaches such as the Schwarz method. Finally error estimates based on the first order necessary optimality conditions, as outlined in the last paragraph of Section 6 is also part of future research.

Acknowledgements

This work was performed at Lawrence Livermore National Laboratory. A. N. Diaz was supported for this work by a Defense Science and Technology Internship (DSTI) at Lawrence Livermore National Laboratory and a 2021 National Defense Science and Engineering Graduate Fellowship. Y. Choi was supported for this work by the US Department of Energy under the Mathematical Multifaceted Integrated Capability Centers – DoE Grant DE – SC0023164; The Center for Hierarchical and Robust Modeling of Non-Equilibrium Transport (CHaRMNET) and partially by LDRD (21-SI-006). M. Heinkenschloss was supported by AFOSR Grant FA9550-22-1-0004 at Rice University. Lawrence Livermore National Laboratory is operated by Lawrence Livermore National Security, LLC, for the U.S. Department of Energy, National Nuclear Security Administration under Contract DE-AC52-07NA27344; IM release number: LLNL-JRNL-849457.

References

[1] Y. Maday, E. M. Rønquist, A reduced-basis element method, J. Sci. Comput. 17 (1-4) (2002) 447–459. doi:10.1023/A:1015197908587.
URL https://doi.org/10.1023/A:1015197908587

[2] Y. Maday, E. M. Rønquist, The reduced basis element method: application to a thermal fin problem, SIAM J. Sci. Comput. 26 (1) (2004) 240–258. doi:10.1137/S1064827502419932.
URL https://doi.org/10.1137/S1064827502419932

[3] L. Iapichino, A. Quarteroni, G. Rozza, A reduced basis hybrid method for the coupling of parametrized domains represented by fluidic networks, Comput. Methods Appl. Mech. Engrg. 221/222 (2012) 63–82. doi:10.1016/j.cma.2012.02.005.
URL https://doi.org/10.1016/j.cma.2012.02.005

[4] P. F. Antonietti, P. Pacciarini, A. Quarteroni, A discontinuous Galerkin reduced basis element method for elliptic problems, ESAIM Math. Model. Numer. Anal. 50 (2) (2016) 337–360. doi:10.1051/m2an/2015045.
URL https://doi.org/10.1051/m2an/2015045

[5] J. L. Eftang, D. B. P. Huynh, D. J. Knezevic, E. M. Ronquist, A. T. Patera, Adaptive port reduction in static condensation, IFAC Proceedings Volumes 45 (2) (2012) 695–699, 7th Vienna International Conference on Mathematical Modelling. doi:10.3182/20120215-3-AT-3016.00123.
URL https://doi.org/10.3182/20120215-3-AT-3016.00123

[6] D. B. P. Huynh, D. J. Knezevic, A. T. Patera, A static condensation reduced basis element method: approximation and *a posteriori* error estimation, ESAIM Math. Model. Numer. Anal. 47 (1) (2013) 213–251. doi:10.1051/m2an/2012022.
URL https://doi.org/10.1051/m2an/2012022

[7] J. L. Eftang, A. T. Patera, Port reduction in parametrized component static condensation: approximation and *a posteriori* error estimation, Internat. J. Numer. Methods Engrg. 96 (5) (2013) 269–302. doi:10.1002/nme.4543.
URL https://doi.org/10.1002/nme.4543

[8] M. Buffoni, H. Telib, A. Iollo, Iterative methods for model reduction by domain decomposition, Comput. & Fluids 38 (6) (2009) 1160–1167. `doi:10.1016/j.compfluid.2008.11.008`.
URL `https://doi.org/10.1016/j.compfluid.2008.11.008`

[9] J. Barnett, I. Tezaur, A. Mota, The Schwarz alternating method for the seamless coupling of nonlinear reduced order models and full order models, arXiv:2210.12551 (2022). `doi:10.48550/ARXIV.2210.12551`.
URL `https://doi.org/10.48550/ARXIV.2210.12551`

[10] A. de Castro, P. Bochev, P. Kuberry, I. Tezaur, Explicit synchronous partitioned scheme for coupled reduced order models based on composite reduced bases, Comput. Methods Appl. Mech. Engrg. 417 (2023) Paper No. 116398. `doi:10.1016/j.cma.2023.116398`.
URL `https://doi.org/10.1016/j.cma.2023.116398`

[11] A. Iollo, G. Sambataro, T. Taddei, A one-shot overlapping Schwarz method for component-based model reduction: application to nonlinear elasticity, Comput. Methods Appl. Mech. Engrg. 404 (2023) Paper No. 115786, 32. `doi:10.1016/j.cma.2022.115786`.
URL `https://doi.org/10.1016/j.cma.2022.115786`

[12] K. Smetana, T. Taddei, Localized model reduction for nonlinear elliptic partial differential equations: localized training, partition of unity, and adaptive enrichment, SIAM J. Sci. Comput. 45 (3) (2023) A1300–A1331. `doi:10.1137/22M148402X`.
URL `https://doi.org/10.1137/22M148402X`

[13] K. Sun, R. Glowinski, M. Heinkenschloss, D. C. Sorensen, Domain decomposition and model reduction of systems with local nonlinearities, in: K. Kunisch, G. Of, O. Steinbach (Eds.), Numerical Mathematics And Advanced Applications. ENUMATH 2007, Springer-Verlag, Heidelberg, 2008, pp. 389–396. `doi:10.1007/978-3-540-69777-0_46`.
URL `http://dx.doi.org/10.1007/978-3-540-69777-0_46`

[14] S. McBane, Y. Choi, Component-wise reduced order model lattice-type structure design, Comput. Methods Appl. Mech. Engrg. 381 (2021) Paper No. 113813, 28. `doi:10.1016/j.cma.2021.113813`.
URL `https://doi.org/10.1016/j.cma.2021.113813`

[15] S. McBane, Y. Choi, K. Willcox, Stress-constrained topology optimization of lattice-like structures using component-wise reduced order models, Comput. Methods Appl. Mech. Engrg. 400 (2022) Paper No. 115525, 25. `doi:10.1016/j.cma.2022.115525`.
URL `https://doi.org/10.1016/j.cma.2022.115525`

[16] C. Hoang, Y. Choi, K. Carlberg, Domain-decomposition least-squares Petrov-Galerkin (DD-LSPG) nonlinear model reduction, Comput. Methods Appl. Mech. Engrg. 384 (2021) Paper No. 113997, 41. `doi:10.1016/j.cma.2021.113997`.
URL `https://doi.org/10.1016/j.cma.2021.113997`

[17] K. Li, K. Tang, T. Wu, Q. Liao, D3M: A deep domain decomposition method for partial differential equations, IEEE Access 8 (2020) 5283–5294. `doi:10.1109/ACCESS.2019.2957200`.
URL `https://doi.org/10.1109/ACCESS.2019.2957200`

[18] W. Li, X. Xiang, Y. Xu, Deep domain decomposition method: Elliptic problems, in: J. Lu, R. Ward (Eds.), Proceedings of The First Mathematical and Scientific Machine Learning Conference, Vol. 107 of Proceedings of Machine Learning Research, PMLR, 2020, pp. 269–286.
URL `https://proceedings.mlr.press/v107/li20a.html`

[19] Q. Sun, X. Xu, H. Yi, Domain decomposition learning methods for solving elliptic problems, arXiv preprint arXiv:2207.10358 (2022). doi:10.48550/arXiv.2207.10358.
URL https://doi.org/10.48550/arXiv.2207.10358

[20] S. Li, Y. Xia, Y. Liu, Q. Liao, A deep domain decomposition method based on fourier features, Journal of Computational and Applied Mathematics 423 (2023) 114963. doi:10.1016/j.cam.2022.114963.
URL https://doi.org/10.1016/j.cam.2022.114963

[21] B. Haasdonk, Chapter 2: Reduced basis methods for parametrized PDEs - a tutorial introduction for stationary and instationary problems, in: P. Benner, A. Cohen, M. Ohlberger, K. Willcox (Eds.), Model Reduction and Approximation: Theory and Algorithms, Computational Science and Engineering, SIAM, Philadelphia, 2017, pp. 65–136. doi:10.1137/1.9781611974829.ch2.
URL https://doi.org/10.1137/1.9781611974829.ch2

[22] A. Quarteroni, A. Manzoni, F. Negri, Reduced Basis Methods for Partial Differential Equations. An Introduction, Vol. 92 of Unitext, Springer, Cham, 2016. doi:10.1007/978-3-319-15431-2.
URL https://doi.org/10.1007/978-3-319-15431-2

[23] M. Hinze, S. Volkwein, Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: Error estimates and suboptimal control, in: P. Benner, V. Mehrmann, D. C. Sorensen (Eds.), Dimension Reduction of Large-Scale Systems, Lecture Notes in Computational Science and Engineering, Vol. 45, Springer-Verlag, Heidelberg, 2005, pp. 261–306. doi:10.1007/3-540-27909-1_10.
URL http://doi.org/10.1007/3-540-27909-1_10

[24] M. Gubisch, S. Volkwein, Chapter 1: Proper Orthogonal Decomposition for linear-quadratic optimal control, in: P. Benner, A. Cohen, M. Ohlberger, K. Willcox (Eds.), Model Reduction and Approximation: Theory and Algorithms, Computational Science and Engineering, SIAM, Philadelphia, 2017, pp. 3–64. doi:10.1137/1.9781611974829.ch1.
URL https://doi.org/10.1137/1.9781611974829.ch1

[25] S. W. Cheung, Y. Choi, D. M. Copeland, K. Huynh, Local lagrangian reduced-order modeling for the rayleigh-taylor instability by solution manifold decomposition, Journal of Computational Physics 472 (2023) 111655. doi:10.1016/j.jcp.2022.111655.
URL https://doi.org/10.1016/j.jcp.2022.111655

[26] D. M. Copeland, S. W. Cheung, K. Huynh, Y. Choi, Reduced order models for lagrangian hydrodynamics, Computer Methods in Applied Mechanics and Engineering 388 (2022) 114259. doi:10.1016/j.cma.2021.114259.
URL https://doi.org/10.1016/j.cma.2021.114259

[27] K. Carlberg, Y. Choi, S. Sargsyan, Conservative model reduction for finite-volume models, Journal of Computational Physics 371 (2018) 280–314. doi:10.1016/j.jcp.2018.05.019.
URL https://doi.org/10.1016/j.jcp.2018.05.019

[28] A. C. Antoulas, Approximation of Large-Scale Dynamical Systems, Vol. 6 of Advances in Design and Control, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005. doi:10.1137/1.9780898718713.
URL https://doi.org/10.1137/1.9780898718713

[29] P. Benner, T. Breiten, Chapter 6: Model order reduction based on system balancing, in: P. Benner, A. Cohen, M. Ohlberger, K. Willcox (Eds.), Model Reduction and Approximation: Theory and Algorithms, Computational Science and Engineering, SIAM, Philadelphia, 2017, pp. 261–295. doi:10.1137/1.9781611974829.

ch6.
URL https://doi.org/10.1137/1.9781611974829.ch6

[30] A. C. Antoulas, C. A. Beattie, S. Gugercin, Interpolatory Model Reduction, Vol. 21 of Computational Science & Engineering, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2020. doi:10.1137/1.9781611976083.
URL https://doi.org/10.1137/1.9781611976083

[31] C. Gu, QLMOR: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 30 (9) (2011) 1307–1320. doi:10.1109/TCAD.2011.2142184.
URL https://doi.org/10.1109/TCAD.2011.2142184

[32] P. Benner, T. Breiten, Two-sided projection methods for nonlinear model order reduction, SIAM J. Sci. Comput. 37 (2) (2015) B239–B260. doi:10.1137/14097255X.
URL http://dx.doi.org/10.1137/14097255X

[33] A. J. Mayo, A. C. Antoulas, A framework for the solution of the generalized realization problem, Linear Algebra Appl. 425 (2-3) (2007) 634–662. doi:10.1016/j.laa.2007.03.008.
URL https://doi.org/10.1016/j.laa.2007.03.008

[34] A. C. Antoulas, I. V. Gosea, A. C. Ionita, Model reduction of bilinear systems in the Loewner framework, SIAM J. Sci. Comput. 38 (5) (2016) B889–B916.
URL https://doi.org/10.1137/15M1041432

[35] I. V. Gosea, A. C. Antoulas, Data-driven model order reduction of quadratic-bilinear systems, Numer. Linear Algebra Appl. 25 (6) (2018) e2200. doi:10.1002/nla.2200.
URL http://dx.doi.org/10.1002/nla.2200

[36] Y. Choi, P. Brown, W. Arrighi, R. Anderson, K. Huynh, Space–time reduced order model for large-scale linear dynamical systems with application to boltzmann transport problems, Journal of Computational Physics 424 (2021) 109845. doi:10.1016/j.jcp.2020.109845.
URL https://doi.org/10.1016/j.jcp.2020.109845

[37] Y. Kim, K. Wang, Y. Choi, Efficient space–time reduced order model for linear dynamical systems in python using less than 120 lines of code, Mathematics 9 (14) (2021) 1690. doi:10.3390/math9141690.
URL https://doi.org/10.3390/math9141690

[38] Y. Choi, K. Carlberg, Space–time least-squares petrov–galerkin projection for nonlinear model reduction, SIAM Journal on Scientific Computing 41 (1) (2019) A26–A58. doi:10.1137/17M1120531.
URL https://doi.org/10.1137/17M1120531

[39] M. Ohlberger, S. Rave, Reduced basis methods: Success, limitations and future challenges, Proceedings of the Conference Algoritmy (2016) 1–12.
URL http://www.iam.fmph.uniba.sk/amuc/ojs/index.php/algoritmy/article/view/389

[40] N. J. Nair, M. Balajewicz, Transported snapshot model order reduction approach for parametric, steady-state fluid flows containing parameter-dependent shocks, Internat. J. Numer. Methods Engrg. 117 (12) (2019) 1234–1262. doi:10.1002/nme.5998.
URL https://doi.org/10.1002/nme.5998

[41] A. Iollo, D. Lombardi, Advection modes by optimal mass transfer, Phys. Rev. E 89 (2014) 022923. `doi:10.1103/PhysRevE.89.022923`.
URL `https://doi.org/10.1103/PhysRevE.89.022923`

[42] N. Cagniart, Y. Maday, B. Stamm, Model order reduction for problems with large convection effects, in: B. N. Chetverushkin, W. Fitzgibbon, Y. A. Kuznetsov, P.Neittaanmäki, J.Periaux, O. Pironneau (Eds.), Contributions to partial differential equations and applications, Vol. 47 of Comput. Methods Appl. Sci., Springer, Cham, 2019, pp. 131–150. `doi:10.1007/978-3-319-78325-3_10`.
URL `https://doi.org/10.1007/978-3-319-78325-3_10`

[43] J. Reiss, P. Schulze, J. Sesterhenn, V. Mehrmann, The shifted proper orthogonal decomposition: a mode decomposition for multiple transport phenomena, SIAM J. Sci. Comput. 40 (3) (2018) A1322–A1344. `doi:10.1137/17M1140571`.
URL `https://doi.org/10.1137/17M1140571`

[44] G. Welper, Interpolation of functions with parameter dependent jumps by transformed snapshots, SIAM J. Sci. Comput. 39 (4) (2017) A1225–A1250. `doi:10.1137/16M1059904`.
URL `https://doi.org/10.1137/16M1059904`

[45] R. Mojgani, M. Balajewicz, Lagrangian basis method for dimensionality reduction of convection dominated nonlinear flows, arXiv:1701.04343v1 (2017). `doi:10.48550/arXiv.1701.04343`.
URL `https://doi.org/10.48550/arXiv.1701.04343`

[46] M. Dihlmann, M. Drohmann, B. Haasdonk, Model reduction of parametrized evolution problems using the reduced basis method with adaptive time-partitioning, in: International Conference on Adaptive Modeling and Simulation ADMOS 2011, 2011, p. 64.

[47] M. Drohmann, B. Haasdonk, M. Ohlberger, Adaptive reduced basis methods for nonlinear convection–diffusion equations, in: J. Fort, J. Fürst, J. Halama, R. Herbin, F. Hubert (Eds.), Finite Volumes for Complex Applications VI Problems & Perspectives: FVCA 6, International Symposium, Prague, June 6-10, 2011, Springer, Berlin, Heidelberg, 2011, pp. 369–377. `doi:10.1007/978-3-642-20671-9_39`.
URL `https://doi.org/10.1007/978-3-642-20671-9_39`

[48] T. Taddei, S. Perotto, A. Quarteroni, Reduced basis techniques for nonlinear conservation laws, ESAIM Math. Model. Numer. Anal. 49 (3) (2015) 787–814. `doi:10.1051/m2an/2014054`.
URL `https://doi.org/10.1051/m2an/2014054`

[49] B. Peherstorfer, D. Butnaru, K. Willcox, Online Adaptive Model Reduction for Nonlinear Systems via Low-Rank Updates, SIAM J. Sci. Comput. 37 (4) (2015) A2123–A2150. `doi:10.1137/140989169`.
URL `http://dx.doi.org/10.1137/140989169`

[50] D. Amsallem, M. J. Zahr, C. Farhat, Nonlinear model order reduction based on local reduced-order bases, Internat. J. Numer. Methods Engrg. 92 (10) (2012) 891–916. `doi:10.1002/nme.4371`.
URL `http://dx.doi.org/10.1002/nme.4371`

[51] J. Barnett, C. Farhat, Quadratic approximation manifold for mitigating the Kolmogorov barrier in nonlinear projection-based model order reduction, J. Comput. Phys. 464 (2022) Paper No. 111348. `doi:10.1016/j.jcp.2022.111348`.
URL `https://doi.org/10.1016/j.jcp.2022.111348`

[52] R. Geelen, S. Wright, K. Willcox, Operator inference for non-intrusive model reduction with nonlinear manifolds, arXiv:2205.02304v1 (2022).
URL `http://arxiv.org/abs/2205.02304v1`

[53] K. Kashima, Nonlinear model reduction by deep autoencoder of noise response data, in: 2016 IEEE 55th Conference on Decision and Control (CDC), 2016, pp. 5750–5755. `doi:10.1109/CDC.2016.7799153`.
URL `https://doi.org/10.1109/CDC.2016.7799153`

[54] D. Hartman, L. K. Mestha, A deep learning framework for model reduction of dynamical systems, in: 2017 IEEE Conference on Control Technology and Applications (CCTA), 2017, pp. 1917–1922. `doi:10.1109/CCTA.2017.8062736`.
URL `https://doi.org/10.1109/CCTA.2017.8062736`

[55] K. Lee, K. T. Carlberg, Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, J. Comput. Phys. 404 (2020) 108973, 32. `doi:10.1016/j.jcp.2019.108973`.
URL `https://doi.org/10.1016/j.jcp.2019.108973`

[56] Y. Kim, Y. Choi, D. Widemann, T. Zohdi, A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder, J. Comput. Phys. 451 (2022) Paper No. 110841, 29. `doi:10.1016/j.jcp.2021.110841`.
URL `https://doi.org/10.1016/j.jcp.2021.110841`

[57] Y. Kim, Y. Choi, D. Widemann, T. Zohdi, Efficient nonlinear manifold reduced order model, arXiv preprint arXiv:2011.07727 (2020). `doi:10.48550/arXiv.2011.07727`.
URL `https://doi.org/10.48550/arXiv.2011.07727`

[58] F. Romor, G. Stabile, G. Rozza, Non-linear Manifold Reduced-Order Models with Convolutional Autoencoders and Reduced Over-Collocation Method, J. Sci. Comput. 94 (3) (2023) Paper No. 74. `doi:10.1007/s10915-023-02128-2`.
URL `https://doi.org/10.1007/s10915-023-02128-2`

[59] T. Taddei, X. Xu, L. Zhang, A non-overlapping optimization-based domain decomposition approach to component-based model reduction of incompressible flows, arXiv (2023). `doi:10.48550/arXiv.2310.20267`.
URL `https://doi.org/10.48550/arXiv.2310.20267`

[60] C. Farhat, S. Grimberg, A. Manzoni, A. Quarteroni, Computational bottlenecks for PROMs: precomputation and hyperreduction, in: P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, L. M. Silveira (Eds.), Model Order Reduction. Volume 2: Snapshot-Based Methods and Algorithms, Walter de Gruyter & Co., Berlin, 2021, pp. 181–243. `doi:10.1515/9783110671490-005`.
URL `https://doi.org/10.1515/9783110671490-005`

[61] R. Everson, L. Sirovich, The Karhunen-Loéve procedure for gappy data, Journal of the Optical Society of America 12 (8) (1995) 1657–1664.

[62] Y. Choi, D. Coombs, R. Anderson, SNS: A solution-based nonlinear subspace method for time-dependent model order reduction, SIAM Journal on Scientific Computing 42 (2) (2020) A1116–A1146.

[63] J. T. Lauzon, S. W. Cheung, Y. Shin, Y. Choi, D. M. Copeland, K. Huynh, S-OPT: A points selection algorithm for hyper-reduction in reduced order models, arXiv preprint arXiv:2203.16494 (2022).

[64] P. T. Boggs, J. W. Tolle, Sequential quadratic programming, in: Acta Numerica, 1995, Cambridge Univ. Press, Cambridge, 1995, pp. 1–51. `doi:10.1017/s0962492900002518`.
URL `https://doi.org/10.1017/s0962492900002518`

[65] J. Nocedal, S. J. Wright, Numerical Optimization, 2nd Edition, Springer Verlag, Berlin, Heidelberg, New York, 2006. `doi:10.1007/978-0-387-40065-5`.
URL `https://doi.org/10.1007/978-0-387-40065-5`

[66] M. Benzi, G. H. Golub, J. Liesen, Numerical solution of saddle point problems, Acta Numer. 14 (2005) 1–137. `doi:10.1017/S0962492904000212`.
URL `https://doi.org/10.1017/S0962492904000212`

[67] H. G. Bock, E. Kostina, J. P. Schlöder, Numerical methods for parameter estimation in nonlinear differential algebraic equations, GAMM-Mitteilungen 30 (2007) 376–408. `doi:10.1002/gamm.200790024`.
URL `http://dx.doi.org/10.1002/gamm.200790024`

[68] M. Heinkenschloss, Mesh independence for nonlinear least squares problems with norm constraints, SIAM J. Optimization 3 (1993) 81–117. `doi:10.1137/0803005`.
URL `http://dx.doi.org/10.1137/0803005`

[69] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (4) (1989) 303–314. `doi:10.1007/BF02551274`.
URL `https://doi.org/10.1007/BF02551274`

[70] A. Pinkus, Approximation theory of the MLP model in neural networks, in: Acta Numerica, 1999, Vol. 8 of Acta Numer., Cambridge Univ. Press, Cambridge, 1999, pp. 143–195. `doi:10.1017/S0962492900002919`.
URL `https://doi.org/10.1017/S0962492900002919`

[71] K. T. Carlberg, C. Farhat, J. Cortial, D. Amsallem, The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows, Journal of Computational Physics 242 (2013) 623 – 647. `doi:10.1016/j.jcp.2013.02.028`.
URL `http://dx.doi.org/10.1016/j.jcp.2013.02.028`

[72] A. Schmidt, D. Wittwar, B. Haasdonk, Rigorous and effective a-posteriori error bounds for nonlinear problems—application to RB methods, Adv. Comput. Math. 46 (2) (2020) Paper No. 32, 30. `doi:10.1007/s10444-020-09741-x`.
URL `https://doi.org/10.1007/s10444-020-09741-x`