Partial Differential Equations and Applications (2024) 5:3
https://doi.org/10.1007/s42985-023-00270-y

ORIGINAL PAPER

Hyper-reduction for parametrized transport dominated problems via adaptive reduced meshes

Sara Grundel[1] · **Neeraj Sarna**[1]

Received: 3 November 2022 / Accepted: 20 December 2023 / Published online: 26 February 2024

Abstract
We propose an efficient residual minimization technique for the nonlinear model-order reduction of parameterized hyperbolic partial differential equations. Our nonlinear approximation space is spanned by snapshots functions over spatial transformations, and we compute our reduced approximation via residual minimization. To speedup the residual minimization, we compute and minimize the residual on a (preferably small) subset of the mesh, the so-called reduced mesh. We show that, similar to the solution, the residual also exhibits transport-type behaviour. To account for this behaviour, we introduce adaptivity in the reduced mesh by "moving" it along the spatial domain depending on the parameter value. Numerical experiments showcase the effectiveness of our method and the inaccuracies resulting from a non-adaptive reduced mesh.

Mathematics Subject Classification 65D99

1 Introduction

For some solution $u(x, t, \mu) \in \mathbb{R}$, consider a parametrized partial differential equation (PDE) given as

$$\partial_t u(x, t, \mu) + \mathcal{L}(u(x, t, \mu), t, \mu) = 0, \quad \forall (x, t, \mu) \in \Omega \times D \times \mathcal{P}. \tag{1}$$

Here, $x \in \Omega \subset \mathbb{R}^d$ is a point in the space domain Ω, $\mathcal{P} \subset \mathbb{R}^p$ is some parameter domain, that can encode, for instance, the change in the material properties, variation in length scales, changes in the background temperature, $t \in D$ is a point in the finite-dimensional time-domain and $\mathcal{L}$ is some spatial differential operator. For the rest of the paper, we include the time-domain $D := [0, T]$ in $\mathcal{Z}$ and express $\mathcal{Z}$ as $\mathcal{Z} = D \times \mathcal{P} \subset \mathbb{R}^{p+1}$ as the full parameter

This article is part of the section "Computational Approaches" edited by Siddhartha Mishra.

✉ Sara Grundel
grundel@mpi-magdeburg.mpg.de

Neeraj Sarna
sarna@mpi-magdeburg.mpg.de

[1] Max-Planck-Institut für Dynamik komplexer technischer Systeme, Sandtorstr. 1, 39106 Magdeburg, Germany

domain. The later sections of our article further elaborate on the relevance of $\mathcal{P}$, we refer to [19] for additional examples.

An exact solution to the above equation is often unavailable and one seeks a numerical approximation

$$u(\cdot, z) \approx u_N(\cdot, z) \in \mathcal{X}_N,$$

with $\mathcal{X}_N$ being a N-dimensional finite-volume/element/difference-type space. We refer to u_N as the full-order model (FOM). In a multi-query setting where solutions at several different parameter instances are needed, due to the high-dimensionality of $\mathcal{X}_N$, computing a FOM is unaffordable. This motivates one to consider a reduced-order model (ROM).

A ROM splits the solution algorithm into an offline and an online phase and performs most of the expensive computations offline, thus making the online phase efficient. For some finite number of training parameters $\{z^{(i)}\}_{i=1,\dots,m} \subset \mathcal{Z}$, the offline phase computes the set of solution snapshots $\{u_N(\cdot, z^{(i)})\}_i$ that are used by the online phase to efficiently approximate the FOM of a different parameter. If the number of snapshots required to reasonably approximate the FOM are sufficiently small, then one can expect a ROM to be more efficient than the FOM. We refer to the review article [4] and the later sections of our article for further details of the offline and the online phase.

This work focuses on parametrized hyperbolic PDEs. For such equations, there is ample numerical and analytical evidence indicating that a ROM based on a linear approximation space is inefficient/inaccurate, meaning a large number of solution snapshots are required to reasonably approximate the FOM. The inefficiency arises from the poor approximability of the solution set $\{u(\cdot, z) \ : \ z \in \mathcal{Z}\}$ (or the so-called solution manifold) in a linear space—we refer to [5, 7, 11, 17, 18, 27, 31] for proofs related to the slow m-width decay of these solution sets and the related numerical experiments. It is basically clear that a nonlinear approximation space is needed and several ideas for such nonlinear spaces are presented in the literature [6, 12, 16, 21, 28]. All these papers show the good approximation quality of the defined space but lack in finding efficient algorithms to compute reduced solutions within this space. We do not consider methods that interpolate between different reduced spaces as our parameter includes time and we are interested in solving a reduced time dependent differential equation with a time derivative.

Given a linear approximation space different projection methods such as Galerkin projection exist to reduced the original equation to a low-dimensional one, for a non-linear search space we can not use a projection method that is equivalent to the residual minimization as is done in the linear setting. Performing residual minimization directly in the online phase is often of the same complexity as computing the FOM [6, 8, 17]. This is the core methodology presented in this paper, namely to reduce the computational cost of the residual minimization by using an adaptive reduced mesh to reduced the minimization.

We have organized the rest of the article as follows. Section 2 presents our FOM. The discussion is a brief recall of the standard first-order finite-volume methods. Section 3 presents our ROM. We discuss the sampling of the parameter domain, we concretely define the transformed snapshot based approximation space, we present the residual minimization technique and explain the reason behind its complexity scaling with the dimension of the FOM. Section 4 presents a adaptive reduced mesh technique to speedup residual minimization. This technique attempts to make our ROM more efficient than the FOM. Section 5 present numerical examples showcasing the accuracy of our method.

2 Full-order model (FOM)

To compute our FOM, we consider an explicit Euler time-stepping scheme and a first-order finite-volume (FV) spatial discretization. The details are as follows. For a parametrized hyperbolic conservation law, the differential operator $\mathcal{L}$ appearing in (1) is given as

$$\mathcal{L}(\cdot, z) = \nabla \cdot f(\cdot, z) \quad \forall z \in \mathcal{Z}. \tag{2}$$

The flux-function $f(\cdot, z) : \mathbb{R} \to \mathbb{R}^d$ is assumed to be convex and at least twice-differentiable. For all $(x, \mu) \in \mathbb{R}^d \times \mathcal{P}$, the initial conditions read $u(x, t = 0, \mu) = u_0(x, \mu)$. We assume that $u_0(\cdot, \mu)$ is compactly supported. As a result, due to a finite speed of propagation, for any finite final time T, the solution $u(\cdot, z)$ is also compactly supported. Therefore, we consider a bounded and connected spatial domain $\Omega \subset \mathbb{R}^d$, which, for all $z \in \mathcal{Z}$, contains the support of $u(\cdot, z)$. Along the boundary $\partial\Omega$ and for all $z \in \mathcal{Z}$, we prescribe $u(\cdot, z) = 0$.

A discretization of the space-time domain is as follows. We consider a two-dimensional square spatial domain i.e., $d = 2$ and $\Omega = [0, 1]^2$. We consider $N_x \in \mathbb{N}$ number of mesh elements in each spatial direction, resulting in a grid-size of $\Delta x := 1/N_x$. To discretize the time-domain D, we consider the discrete time-steps $\{t_k\}_{k=0,\ldots,K}$ ordered such that

$$0 = t_0 < t_1 \cdots < t_K = T. \tag{3}$$

For simplicity, we consider a constant time-step of Δt.

An explicit Euler time stepping scheme and a FV spatial discretization of the evolution equation (1) provides

$$u_N(\cdot, t_{k+1}, \mu) = u_N(\cdot, t_k, \mu) + \Delta t \times \mathcal{L}_N(u_N(\cdot, t_k, \mu), t_k, \mu), \tag{4a}$$

$$\forall k \in \{0, \ldots, K-1\}, \mu \in \mathcal{P}. \tag{4b}$$

The full order solution is approximated in a finite dimensional subspace $\mathcal{X}_N$ which is equivalent to $\mathbb{R}^N$ with $N = N_x^2$ for a 2-dimensional spatial variable and a vector representation of the solution therefore exists. The operator $\mathcal{L}_N : \mathbb{R}^N \to \mathbb{R}^N$ is a result of a FV discretization and its explicit expression can be found in any standard textbook (for instance, [15]) on FV methods. As the numerical flux, we choose the Local-Lax–Friedrich (LLF) flux—a different choice does not change the forthcoming discussion. If the time-step Δt does not satisfy the CFL-condition, we compute time steps in between that are not used later on and therefore ignored. However one can easily include all solutions in the snapshot samples later one without changing the later algorithms.

3 Reduced-order model (ROM)

We base the construction of the reduced order model on the idea of snapshots, meaning for certain parameter values $\{z^{(i)}\}$ we have already computed solutions of the FOM $u_N(\cdot, z^{(i)})$. In a classical snapshot approach one looks for a solution of the reduced system in the space spanned by all the snapshot namely $\text{span}\{u_N(\cdot, z^{(i)})\}_i$. This does not work well for solutions that differ in the parameter by being transported in the spatial domain. Interpolating function over a linear space results in the typical staircase solutions. Therefore we want the solution to be approximated in a space that is spanned by snapshot solutions over a transformed spatial domain. For now, we introduce a spatial transform $\varphi(\cdot, z, z^{(i)}) : \Omega \to \mathbb{R}^d$ without specifying the properties and how to compute it, and approximate the solution $u_N(\cdot, z)$ in the space

$$\mathcal{X}_m(z) := \Pi(\text{span}\{u_N(\varphi(\cdot, z, z^{(i)}), z^{(i)})\}_{i \in \Lambda(z)}). \tag{5}$$

taking the span of transformed snapshot function. As this span may not be a subspace of our large discretization space $\mathcal{X}_N$ we need Π, a projection operator from $L^2(\Omega)$ or the used surrounding space to $\mathcal{X}_N$ to ensure that $\mathcal{X}_m$ is a subspace of $\mathcal{X}_N$. The set $\Lambda(z) \subseteq \{1, \dots, m\}$ is a z-dependent index set, selecting a subset of the precomputed solutions for a given parameter value z. One could take all snapshot solutions but for computational efficiency taking only some is preferred and for accuracy it turns out that a few snapshots are often enough. The index set is picked as those indices, which are close to the given parameter z. In the numerical experiments we often only take four indices for the four nearest neighbours in the two dimensional parameter space. However, depending on the application one could take any numbers of nearest neighbours in a given appropriate norm [1, 17].

The choice of φ the spatial transformation function and its theoretical foundation is rather complicated, we refer to [31, 32] for more detail, where some uniqueness under certain circumstances can be shown. However the computation is still a nontrivial task then. Theoretically we could take an approximation space in which snapshots are collected evaluated at several spatial transformations. This will be investigated in future work.

We also restrict ourselves to a rather simple construction of the spatial transformation. We use a numerical construction of φ that is accurate for the numerical experiments considered later in a data-driven manner while computing the solution by using the original PDE. It is also possible to compute φ via a PDE based approach [7, 24]. This is typically done by computing characteristic curves of the PDE, which can result in a highly accurate φ but for non-linear problems, is limited to classical solutions. On the opposite one could compute not only φ but also the solution in a fully data-driven approach [31]. We however suspect that by discarding the PDE in the online phase, one might miss out on some physics of the problem, resulting in inaccuracies.

3.1 The approximation space

So far the definition of the spatial transformation function φ as well as the index set $\Lambda(z)$ are kept a bit vague. In order to fully define (5) those need to be specified more clearly. Construct the spatial transform from the full order solutions by assuming that $\varphi(\cdot, z, \hat{z})$ is a spatial shift function i.e.,

$$\varphi(\cdot, z, \hat{z}) := x - c(z, \hat{z}), \quad \forall z, \hat{z} \in \mathcal{Z}, \tag{6}$$

reducing to problem to computing the spatial shift $c(z, \hat{z}) \in \mathbb{R}^d$. This is done via L_2-minimization [25]. We acknowledge that the above ansatz has limited applicability. As the numerical experiments and the discussion in [31] indicate, its applicability is limited to problems with a single large discontinuity or to multiple discontinuities moving with the same velocity. To cater to a broader class of problems, we will require a more sophisticated φ than that considered above—see [17, 22, 31, 32]. We first compute $\{c(z^{(j)}, z^{(i)})\}_{i,j}$ for all pairs of parameter samples and interpolate the spatial transform via a Lagrange interpolation.

In order to have some regularity in $u(\varphi(x, z, \cdot), \cdot)$ we want to satisfy approximately the matching condition

$$\varphi(\mathcal{D}(z^{(i)}), z^{(j)}, z^{(i)}) = \mathcal{D}(z^{(j)}), \tag{7}$$

where $\mathcal{D}(z^{(i)}) \subset \Omega$ and $\mathcal{D}(z^{(j)}) \subset \Omega$ represent the point/curve/surface of discontinuity in $u(\cdot, z^{(i)})$ and $u(\cdot, z^{(j)})$, respectively. With our spatial shift ansatz for φ given in (6), the matching condition transforms to

$$\mathcal{D}(z^{(i)}) = \mathcal{D}(z^{(j)}) + c(z^{(i)}, z^{(j)}). \tag{8}$$

To find an approximate $\mathcal{D}(z^{(j)})$ (and $\mathcal{D}(z^{(i)})$), we apply to $u_N(\cdot, z^{(i)})$ the multi-resolution-analysis (MRA) based troubled cell indicator proposed in [29] (any other shock-detection technique [13, 20] also suffices). For a 1D spatial domain and for a finite-volume scheme, $\mathcal{D}(z^{(j)})$ is a collection of grid points at which the first-order derivative (approximated via central differences) overshoots a user-specified tolerance. Equivalently, $\mathcal{D}(z^{(j)}) := \{x_i : |u_N(x_{i+1}, z^{(j)}) - u_N(x_{i-1}, z^{(j)})| > \mathcal{K}\Delta x\}$, where x_i and $u_N(x_i, z^{(j)})$ represent the center of the i-th cell and the finite-volume approximation in the i-th cell, respectively. In our experiments, $\mathcal{K} = 5$ provides reasonable results. The general definition of $\mathcal{D}(z^{(j)})$ for a higher-order finite element scheme in multiD can be found in [29].

Remark 3.1 In general, condition (8) is (very) restrictive. For instance, in a 1D spatial domain, the condition holds only if the solution has a single shock or multiple shocks that move with the same velocity. However, the condition is violated for two shocks moving with different velocities. Furthermore, in a multi-dimensional setting, even a single shock that changes in length violates the above condition.

Despite the restrictions of the above condition, empirically, we observe that a spatial shift provides accurate results for problems that approximately satisfy the above relation. For instance, in a multi-dimensional setting, $\mathcal{D}(z^{(i)})$ might be a translation of $\mathcal{D}(z^{(j)})$ but with an elongation. If the elongation is not significant then, we recover a reasonable snapshot transformation via shifting. Similarly, when $\mathcal{D}(z^{(i)})$ has the same length as $\mathcal{D}(z^{(j)})$ but is both a translation and a rotation of $\mathcal{D}(z^{(j)})$, we expect a spatial shift to provide reasonable results if the rotation is not significant.

In order to find c we define first the set $B(z^{(j)}, z^{(i)}) \subset \mathbb{R}^d$

$$\mathcal{B}(z^{(j)}, z^{(i)}) := \{c^* \ : \exists x_i^* \in \mathcal{D}(z^{(i)}) \text{ and } x_j^* \in \mathcal{D}(z^{(j)}) \text{ s.t. } c^* = x_i^* - x_j^*, \ \}. \tag{9}$$

Out of all the possible shifts in $\mathcal{B}(z^{(j)}, z^{(i)})$, we select the one that solves the minimization problem

$$c(z^{(j)}, z^{(i)}) = \underset{c^* \in \mathcal{B}(z^{(j)}, z^{(i)})}{\arg\min} \|u_N(\cdot - c^*, z^{(i)}) - u_N(\cdot, z^{(j)})\|_{L^2(\Omega)} \tag{10}$$

an L2-minimization technique proposed in [17, 23, 25, 26, 31]. We solve the above problem via enumeration. In our numerical experiments, the set $\mathcal{B}(z^{(j)}, z^{(i)})$ is not too large and a solution via enumeration is affordable. Given $c(z^{(j)}, z^{(i)})$ the general function $c(z, z^{(i)})$ is defined via Lagrange interpolation. Then that defines the spatial transform φ as in (6).

Remark 3.2 (Computing the projection operator Π) On a Cartesian mesh, we define Π (given in (5)) by approximating the shifts by an integer multiple of Δx. This provides

$$\Pi c(t_k, \mu, z^{(i)}) = \left\lfloor \frac{c(t_k, \mu, z^{(i)})}{\Delta x} \right\rfloor \Delta x, \tag{11}$$

where $\lfloor n \rfloor$ represents the greatest integer less than equal to n and acts component-wise if n is more than one dimensional.

The set $\Lambda(z)$ can be generally defined as $\Lambda_\epsilon(z)$ for a certain ϵ.

$$\Lambda(z) := \{i : \|z - z^{(i)}\| \leq \epsilon\}$$

If the parameter is sampled in a grid we can pick epsilon in such a way that generically we end up with the corners of the cell the parameter value z lies in.

3.2 Sampling the parameter domain

We consider a two-dimensional parameter domain $\mathcal{Z}$—an extension to higher dimensions is straightforward. We assume that $\mathcal{Z}$ is (or can be mapped via a bijection to) a rectangle. We take N_t and N_μ uniformly placed samples from D and $\mathcal{P}$, respectively. The vertices of $\mathcal{Z}$ are included in the samples, and the samples from D are a subset of the time-instances $\{t_k\}_{k=0,\dots,K}$ used to compute the FOM. To collect the samples from $\mathcal{Z}$, we take a tensor-product of the samples in D and $\mathcal{P}$.

Remark 3.3 (Scaling with p) Uniform sampling can make snapshot computation unaffordable for large values of p, the dimension of the parameter domain $\mathcal{P}$. In that case, one can consider a greedy/sparse sampling technique [10, 19] or Smolyak quadrature rules [14, 30].

Remark 3.4 (The locality of the approximation space $\mathcal{X}_m(z)$) The space $\mathcal{X}_m(z)$ (and also the one considered in [17]) is local—it only uses parameter samples that lie in a neighbourhood of the target parameter.

3.3 Residual minimization

For a target parameter $z = (t, \mu) \notin \{z^{(i)}\}_i$, the following steps of the online phase compute a reduced approximation $u_m(\cdot, z)$. First we introduce the previous discrete time steps $0 = t_0 < \cdots < t_{K^*} = t$, assuming that the given t is one of the timesteps of the full order model. We also keep Δt as in the FOM. Different time-stepping is possible see [7] for details. For each previous time step we compute $\{c(t_k, \mu, z^{(i)})\}_i$ interpolating the snapshots of the shifts $\{c(z^{(j)}, z^{(i)})\}_{i,j}$ via Lagrange polynomial interpolation. Any regression or interpolation technique can be used. Compute the reduced approximation $u_m(\cdot, t_k, \mu)$ in $\mathcal{X}_m(t_k, \mu)$ via residual minimization given as [1, 7, 17]

$$\begin{aligned} u_m(\cdot, t_k, \mu) = \underset{v \in \mathcal{X}_m(t_k,\mu)}{\arg\min} \; & \| v - u_m(\cdot, t_{k-1}, \mu) \\ & + \Delta t \times \mathcal{L}_N(u_m(\cdot, t_{k-1}, \mu), t_{k-1}, \mu) \|_{L^2(\Omega)}. \end{aligned} \tag{12}$$

We initialize with $u_m(t_0, \mu) = \arg\min_{v \in \mathcal{X}_m(t_0,\mu)} \|v - u_0(\cdot, \mu)\|_{L^2(\Omega)}$. The approximation space $\mathcal{X}_m(t_k, \mu)$ is given in (5). The operator $\mathcal{L}_N$ approximates the evolution operator $\mathcal{L}$. Its precise form is not important here and depends on the details of the full order model. The complexity of solving the above problem scales with the dimension of the FOM therefore, we later equip it with hyper-reduction.

The approximation space $\mathcal{X}_m(z)$ is isomorphic to range$(A(z))$ where the matrix $A(z)$ is of size $N \times \ell$ where ℓ is the number of chosen neighbours for a given value of z. Each column of the matrix represents the discrete FOM solution of the shifted snapshot.

Similarly $u_m(\cdot, t_{k-1}, \mu) + \Delta t \times \mathcal{L}_N(u_m(\cdot, t_{k-1}, \mu), t_{k-1}, \mu)$ can be represented in $\mathbb{R}^N$ by $b(t_k, \mu)$ and we can write the residual minimization in matrix vector form as

$$\alpha(t_{k+1}, \mu) = \underset{y \in \mathbb{R}^\ell}{\arg\min} \, \| \underline{A(t_{k+1}, \mu) y - b(t_k, \mu)} \|_{l^2}^2, \tag{13}$$

Consider the underlined vector in the above problem (13). The i-th element of this vector represents the residual in the i-th mesh element. Since we take the l^2-norm of the entire vector, we minimize the residual over the entire mesh. This can become a computational bottle neck. We propose to collect element ids in the possibly z-dependent set $\mathcal{E}_z \subseteq \mathcal{E}_{full}$,

where

$$\mathcal{E}_{full} := \{1, \ldots, N\},$$

such that doing the residual minimization only over that set leads to a good approximation quality of the reduced system. The computational cost of the least square minimization of (13) over only $\mathcal{E}_z$ has $\mathcal{O}(\ell^2 n)$ operations, where $n = \#\mathcal{E}_z$.

3.4 Summary

The reduced model is a fast algorithm to compute the solution of a given equation at a certain time and parameter value $z^* = (t^*, \mu^*)$. That means given that we are interested in an approximation of $u_N(\cdot; t^*, \mu^*)$. The steps to get there are

1. Find interpolation points $z_j = (t_j, \mu_j)$ close for which full order solutions exists.
2. Compute $c(z^*, z_j)$ (The function c is a simple Lagrange interpolation over the known values $c(z_i, z_J)$ of all pairs of sample points)
3. Solution is then found through residual minimization over the ansatz functions $u_N(\cdot - c(z^*, z_j), z_j)$.

4 Hyper-reduction—finding $\mathcal{E}_z$

With $\mathcal{E}_z = \mathcal{E}_{full}$, our ROM is (at least) as expensive as the FOM, which is undesirable. While maintaining the accuracy of the ROM, we want to choose $\mathcal{E}_z$ such that $n \ll N$. This way, one can expect the ROM to be more efficient than the FOM. We pursue two approaches to compute such an $\mathcal{E}_z$.

1. *The non-adaptive technique*, where $\mathcal{E}_z$ is independent of the parameter z.
2. *The adaptive technique* where $\mathcal{E}_z$ changes with the parameter z i.e., the index set $\mathcal{E}_z$ is adaptive.

First, we present the non-adaptive technique and its shortcomings.

4.1 Non-adaptive technique

This technique consists only of an offline phase. At $\{\bar{\mu}_j\}_{j=1,\ldots,m_{hyp}} \in \mathcal{P}$ parameter samples, we solve the non-hyper-reduced least-squares problem in (13) and collect snapshots of the residuals for each time step generating a residual snapshot matrix S of size $N \times (N_t \times m_{hyp})$. The columns of this matrix are given by

$$S_{col} := A(t_k, \bar{\mu}_i)\alpha(t_k, \bar{\mu}_i) - b(t_{k-1}, \bar{\mu}_i). \text{ for } k = 0, \ldots N_t, i = 1, \ldots, m_{hyp} \tag{14}$$

The relevant indices are those in which the residual is large and are picked via a point-selection algorithm as described in Algorithm 1. (Any other point-selection algorithm from [2, 3, 8, 17] could also be used here).

Note that as compared to [2], we apply the algorithm directly to S and not to its POD modes. Numerical experiments suggest that both the strategies provide similar results.

The selection of the relevant mesh points is parameter value independent and our numerical experiments suggest that the non-linearity of the approximation space $\mathcal{X}_m(z)$ induces a transport-type behaviour in the residual. As a result, only a large reduced mesh computed

Algorithm 1 Summary of the reduced mesh selection algorithm from [2]

Input $\mathcal{S}, n$
Output $\mathcal{E}_z$
for $i \in \{1, \ldots, N\}$ **do**
  $r(i) = \|\mathcal{S}(i,:)\|_2$ {$\mathcal{S}(i,:)$ denotes the i-th row of $\mathcal{S}$. }
end for
[r-sorted,idx] = sort(r) {Sort in decreasing order and r-sorted=r(idx).}
$\mathcal{E}_z \leftarrow idx(1:n)$

using Algorithm 1 can provide a reasonable accuracy - a similar observation holds for the other point-selection techniques outlined in [3, 8]. This is undesirable because, at least ideally, for some error tolerance of practical interest ($\|u_N(\cdot, z) - u_m(\cdot, z)\|_{L^2} \leq$ TOL, for instance), the size of the reduced mesh should be as small as possible.

4.2 Adaptive technique

To account for the transport-type behaviour of the residual, we introduce adaptivity in the reduced mesh. For this we precompute different index sets $\mathcal{E}_{\hat{z}}$. Instead of using one large matrix S for all the residual snapshots we divide the set of training parameters into subsets and create several residual snapshot matrices for each subset. The resulting index set is then used for all parameters within the subset. The more subsets we define the better the approximation. The maximal number of subset we can pick is the number of snapshots in the parameter domain. This could be a reasonable choice. For each parameter value we pick a reduced mesh meaning the index set $\mathcal{E}_{z_i}$. During the online phase in which we compute the ROM we pick the index set given by the parameter value closest to the one we are evaluating, or we pick a combination of index sets collected by the 2 nearest neighbours.

5 Numerical experiments

The goal of our numerical experiments is to study the accuracy of the adaptive reduced mesh technique compared to the non-adaptive technique and the FOM as well as the computation time improvement in particular through the two different hyperreduction methods. We abbreviate the different ROMs that we compare via numerical experiments as in Table 1. The space $\tilde{\mathcal{X}}_m$ is given as

$$\tilde{\mathcal{X}}_m := \text{span}\{u_N(\cdot, z^{(i)})\}, \tag{15}$$

which is just the space spanned by all snapshots. To compute a solution in $\tilde{\mathcal{X}}_m$, we use the residual minimization technique from Sect. 3.3. We are only interested in the accuracy of the L-ROM and do not equip it with any hyper-reduction technique.

Recall that N_μ and N_t represent the number of parameter samples along the domains $\mathcal{P}$ and D, respectively—see Sect. 3.2 for details. We quantify the error in our ROMs via the relative error

$$E(N_t, N_\mu) := \|e\|_{L^\infty(\mathcal{Z})} \quad \text{where} \quad e(z) := \frac{\|u_N(\cdot, z) - u_m(\cdot, z)\|_{L^2(\Omega)}}{\|u_N(\cdot, z)\|_{L^2(\Omega)}}. \tag{16}$$

Table 1 Abbreviations for the different ROMs compared via numerical experiments

Abbreviation	Approximation space	Hyper-reduction
Adp-S-ROM	$\mathcal{X}_m(z)$	Adaptive (see Sect. 4.2)
N-Adp-S-ROM	$\mathcal{X}_m(z)$	Non-adaptive (see Sect. 4.1)
S-ROM	$\mathcal{X}_m(z)$	None
L-ROM	$\tilde{\mathcal{X}}_m$	None

See (5) and (15) for a definition of $\mathcal{X}_m(z)$ and $\tilde{\mathcal{X}}_m$, respectively. *S* shifted, *L* linear

The reduced solution u_m can result from either of the ROMs listed in Table 1. We approximate the $L^\infty(\mathcal{Z})$ norm via

$$\|e\|_{L^\infty(\mathcal{Z})} \approx \max_{z \in \mathcal{Z}_{target}} |e(z)|, \tag{17}$$

where $\mathcal{Z}_{target} \subset \mathcal{Z}$ is a sufficiently dense, problem dependent and finite set of target parameters given later.

Remark 5.1 (Software and hardware details) All the simulations are run using matlab, in serial, and on a computer with two Intel Xeon Silver 4110 processors, 16 cores each and 92GB of RAM.

5.1 1D Linear advection

We consider a linear one-dimensional advection equation with a parameterised advection speed and a reaction term

$$\partial_t u(x,t,\mu) + \mu \partial_x u(x,t,\mu) = -r \times u(x,t,\mu), \quad \forall (x,t,\mu) \in \Omega \times D \times \mathcal{P}. \tag{18}$$

We choose $r = 10^{-4}$, $\Omega = [0, 3]$, $D = [0, 0.5]$, and $\mathcal{P} = [1, 3]$. The initial data reads

$$u_0(x,\mu) = \begin{cases} \mu, & x \in [0.5, 1] \\ 0, & \text{else} \end{cases}, \quad \forall \mu \in \mathcal{P}. \tag{19}$$

We choose a constant time-step of $\Delta t = 1/N_x$, which satisfies the CFL-condition everywhere. We choose N_μ, $N_t = 2$ and $N_x = 10^3$. Furthermore, as a set of target parameters, we choose $\mathcal{Z}_{target} = \{(t_i, \tilde{\mu}_j)\}_{i,j}$, where t_j are the time-instances at which we compute the ROM, and $\{\tilde{\mu}_j\}_j$ are 40 different parameter samples uniformly placed inside $\mathcal{P}$. For the offline phase of the hyper-reduction, we consider five uniformly placed samples inside $\mathcal{P}$ i.e., $m_{hyp} = 5$. For the different ROMs outlined in Table 1, Table 2 compares the error $E(N_\mu, N_t)$. The size of the reduced mesh for the hyperreduction is choosen to be $n = N_x \times 5 \times 10^{-3}$, which is 0.5% of the total mesh size. A few observations are in order. Firstly, with a relative error of 1.05, the L-ROM performs poorly. It results in an error that is almost five and ten times larger than that resulting from the Adp-S-ROM and the S-ROM, respectively. Secondly, the error resulting from the Adp-S-ROM is twice of that resulting from the S-ROM. Given the speedup offered by the Adp-S-ROM (see the results below), we insist that this loss in accuracy is reasonable. Lastly, the N-Adp-S-ROM showed large oscillations and appeared to be unstable, which resulted in extremely large error values. Increasing the size of the reduced mesh (as discussed next) makes N-Adp-S-ROM stable and provides acceptable accuracy.

Table 2 Results for 1D linear advection

	N-Adp-S-ROM	Adp-S-ROM	S-ROM	ROM
$E(N_\mu, N_t)$	1.95×10^{30}	0.21	0.10	1.05

Error comparison between the different ROMs listed in Table 1. Computations performed with $N_\mu, N_t = 2$, $N_x = 10^3$, and $n = N_x \times 5 \times 10^{-3}$. The N-Adp-S-ROM showed large oscillations and appeared to be unstable, hence the extremely large error values

Table 3 Results for test 1D linear advection

n (%N_x)	$E(N_\mu, N_t)$ Adp-S-ROM	N-Adp-S-ROM
100 (10)	0.12	5.6×10^{27}
200 (20)	0.10	1.1×10^{26}
400 (40)	0.10	1.01×10^{13}
800 (80)	0.10	0.10

Error versus the size of the reduced mesh n. For the underlined values, the N-Adp-S-ROM showed large oscillations and appeared to be unstable, hence the large error values

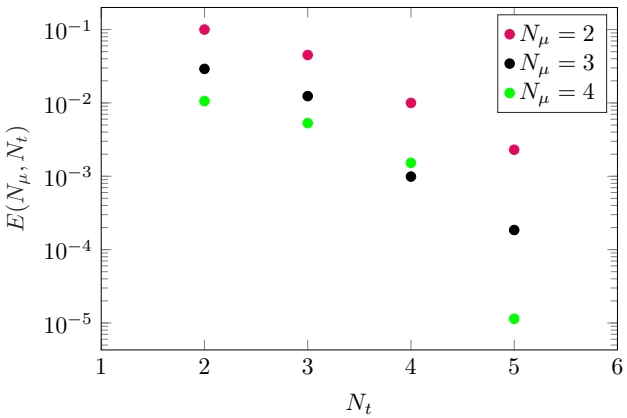
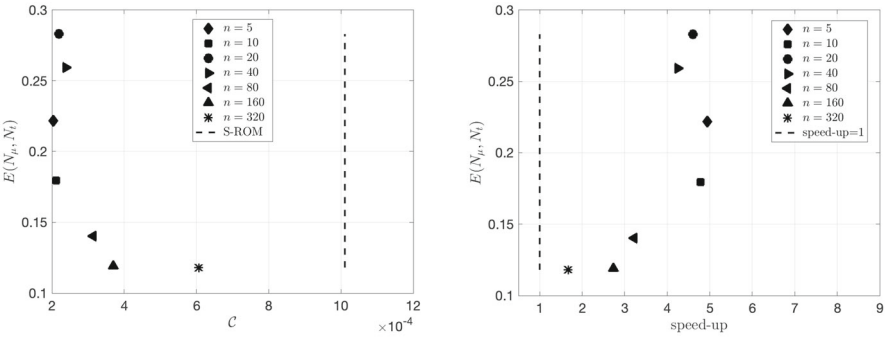


Fig. 1 Error of the shifted ROM without any hyperreduction for different sizes of the sampling set

Remark 5.2 (Treatment of the reaction term) The reaction term in (18) is treated implicitly. This results in the system matrix $A(z)$ being multiplied by $(1+r\Delta t)$. The rest of the framework remains the same.

By increasing the number of points used in the hyperreduction the N-Adp-S-ROM will be able to achieve the error of the S-ROM without hyperreduction. Table 3 compares the error values with varying but large n.

Since the accuracy of our ROM is limited by our choice of N_t and N_μ, and increasing n can only offer so much accuracy, the error from Adp-S-ROM stagnates after a value of 0.11. In Fig. 1 we see the error value of the S-ROM for increased snapshot sizes.



(a) Error vs runtime (b) Error vs speedup w.r.t the S-ROM

Fig. 2 Results for 1D linear advection computed with the Adp-S-ROM. See (16) and (20) for a definition of the error $E(N_\mu, N_t)$ and the runtime $\mathcal{C}$, respectively. The dashed line represents (a) the time taken by the S-ROM, and (b) a speedup of one

In order to show the necessity of the adaptive hyper reduction we consider the average runtime given as

$$\mathcal{C} := \sum_{z \in \mathcal{Z}_{\text{target}}} \mathcal{C}_z / (\#\mathcal{Z}_{\text{target}}), \tag{20}$$

where $\mathcal{C}_z$ represents the cpu-time (measured with the tic-toc function of matlab) required by the online stage of the ROMs (or by the FOM) to compute the solution at the parameter z.

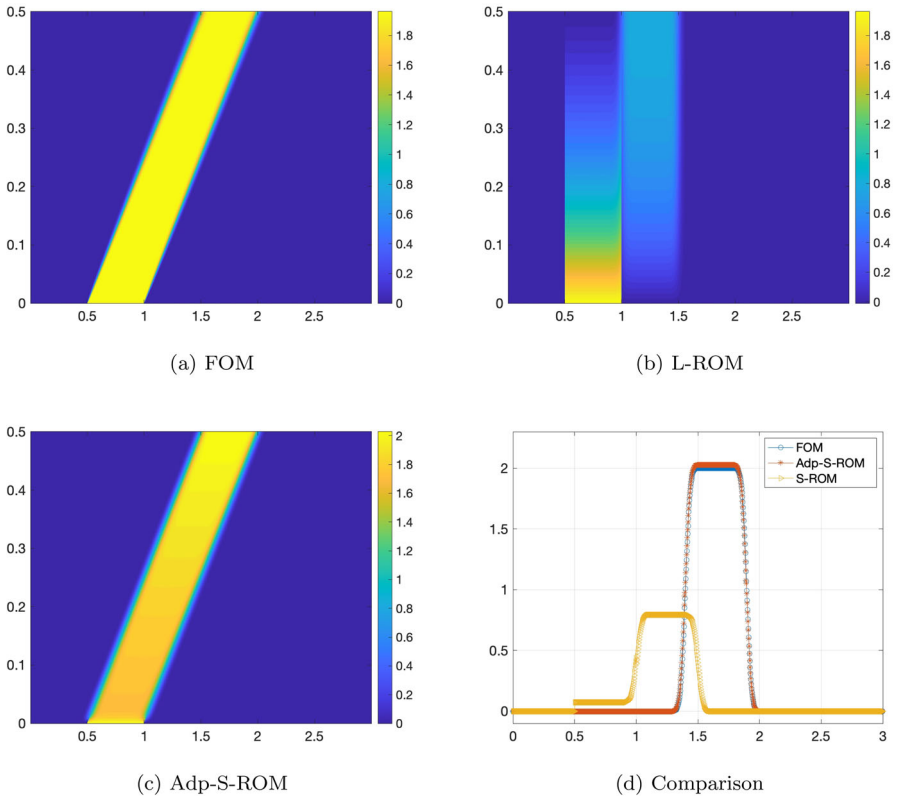
Figure 2 plots the runtime $\mathcal{C}$ and the speedup against the error $E(N_\mu, N_t)$. Here we are only interested in the speedup of the hyperreduction technique meaning comparing the runtime of the S-ROM with the Adp-S-ROM. We make the following observations. (i) Although not monotonically, the error converges with n. The non-monotonic convergence of the error can be an artefact of the point selection algorithm given in Algorithm 1. (ii) Increasing n increases the runtime. This is consistent with the fact that the cost of the Adp-S-ROM scales with n. (iii) At worst, for $n = 320$, the Adp-S-ROM is 1.8 times faster than the S-ROM, and at best, for $n = 5$, it is five times faster than the S-ROM. (iv) The problem is one-dimensional therefore, the explicit time-stepping based FOM is already very efficient. As a result, none of the ROMs provide any speedup. We refer to the last test case for a 2D problem where, as compared to the FOM, our hyper-reduction technique offers a significant speedup.

Figure 3 compares the solution between the FOM, Adp-S-ROM and L-ROM. Notice that, due to the linearity of the approximation, L-ROM entirely misrepresents the solution. It fails to capture the transport nature of the solution. In contrast, owing to the shifting of the reduced basis, Adp-S-ROM provides an accurate approximation. Furthermore, at least for all our test cases, it does not exhibit any Gibbs-like phenomenon.

5.2 1D Burgers equation

As an example of a nonlinear equation we tested the one dimensional Burgers equation

$$\partial_t u(t, x) + \frac{1}{2} \partial_x u^2(t, x) = 0$$



(a) FOM (b) L-ROM

(c) Adp-S-ROM (d) Comparison

Fig. 3 Solution comparison for 1D linear advection. The `L-ROM` and `Adp-S-ROM` both use 4 snapshots and for the hyperreduction $n = 5$

with initial condition as in (21).

$$u_0(x) = \begin{cases} \mu \exp\left(-1/\left(1-\left(\frac{x-\delta_1}{\delta_2}\right)^2\right)\right), & \left|\frac{x-\delta_1}{\delta_2}\right| < 1 \\ -\exp\left(-1/\left(1-\left(\frac{x+\delta_1}{\delta_2}\right)^2\right)\right), & \left|\frac{x+\delta_1}{\delta_2}\right| < 1 \\ 0, & \text{else} \end{cases} \tag{21}$$

where we set $\delta_1 = 0.5$ and $\delta_2 = 0.2$. The solution as shown in Fig. 4 presents two features moving in opposite directions. We see that the `S-ROM` can capture only one of the features accurately which we expected due to the limitation of the spatial transformation as just a simple shift. The qualitative results are shown in Fig. 4. The speedup of the hyperreduction step shows to be about a factor of 500.

5.3 A moving box function

In this example we discuss approximation quality without using a partial differential equation. We construct a reduced approximation to the set $\{u_N(\cdot, z) \ : \ z \in \mathcal{Z}\}$, where $u_N(\cdot, z)$ is a FV approximation to a function $u(\cdot, z)$ that shifts in Ω and changes its "shape" with the

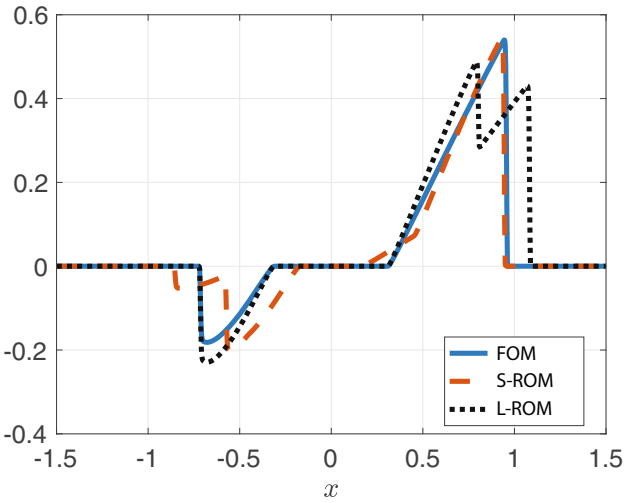


Fig. 4 Solution quality of the S-ROM and the L-ROM of the one dimensional Burgers equation compared to the FOM at $\mu = 2$ and $t = 1$

parameter. For all $(t, \mu) \in D \times \mathcal{P}$, the function $u(\cdot, z)$ is given as

$$u(\cdot, t, \mu) = \begin{cases} \exp(-\mu t), & |x_1 - (\mu + t)| \leq 0.3, \ |x_2 - t| \leq 0.3 \\ 0, & \text{else} \end{cases}. \tag{22}$$

We choose $D, \mathcal{P} = [0, 1]$, and $\Omega = [-0.5, 2.5]^2$. Note that increasing μ shifts $u(\cdot, t, \mu)$ along the x_1-direction, and increasing t shifts $u(\cdot, t, \mu)$ along the vector $(t, t)^T$. To compute the FV approximation $u_N(\cdot, z)$, we project $u(\cdot, z)$ onto the FV approximation space. The details related to the projection and the reduced approximation are discussed later. We set $N_t = N_\mu = 3$, and $N_x = 600$. To compute the S-ROM, we consider the minimization problem

$$\alpha(z) = \underset{y \in \mathbb{R}^4}{\arg\min} \, \| A[\mathcal{E}_z](z) y - b[\mathcal{E}_z](z) \|_{l^2}. \tag{23}$$

Here, $\mathcal{E}_z$ is as defined earlier and $A[\mathcal{E}_z](z)$ is built from A by choosing the corresponding rows as is the vector $b[\mathcal{E}_z](z)$, a sub-vector of the vector $b(z)$. To compute $b(z)$, we project $u(\cdot, z)$ onto the FV approximation space. To perform the projection, we consider tensorized 5×5 Gauss-Legendre quadrature points in each mesh element. The reduced approximation is given by $U_m = A(z)\alpha(z)$. Note that for the current test case, computing the FOM is equivalent to projecting the exact solution onto the FV approximation space. To collect snapshots of the residual, we set $m_{hyp} = 4$. We choose $\mathcal{Z}_{\text{target}}$ as 100×100 uniformly placed and tensorised points inside $\mathcal{Z}$.

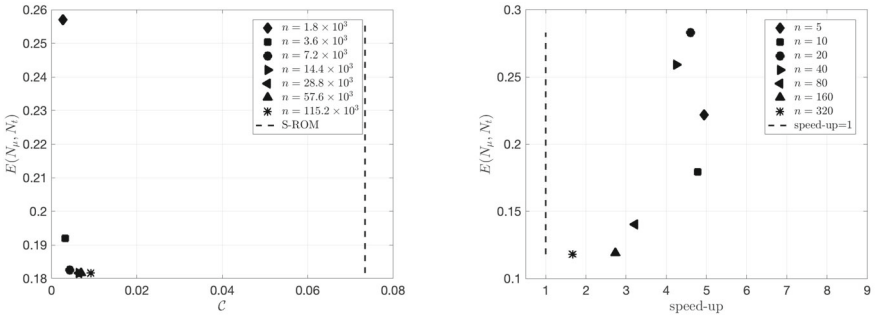
Remark 5.3 (No time-stepping) The above minimization problem does not involve a time-stepping scheme. This allows us to study the errors resulting from the reduced approximation, residual minimization and hyper-reduction without the errors introduced from the time-stepping scheme.

Table 4 presents the error values resulting from the different ROMs listed in Table 1. As the size of the reduced mesh, we choose $n = N_x^2 \times 10^{-2}$, which is 1% of the total mesh size.

Table 4 Results for the moving box

	N-Adp-S-ROM	Adp-S-ROM	S-ROM	L-ROM
$E(N_\mu, N_t)$	1	0.19	0.18	0.84

Computations performed with $N_\mu, N_t = 3$, $N_x = 600$, and $n = N_x^2 \times 10^{-2}$. Error comparison between the different ROMs listed in Table abbrvROM



(a) Error versus runtime (b) Error versus speedup w.r.t the S-ROM

Fig. 5 Results for the moving box, computed with the Adp-S-ROM. The dashed line represents **(a)** the time taken by the S-ROM, and **(b)** a speedup of one

Both the Adp-S-ROM and the S-ROM outperform the L-ROM. The error values resulting from the L-ROM are almost 4.5 times of those resulting from the Adp-S-ROM. At least for the present test case and our choice of n, our adaptive hyper-reduction technique introduces almost no error in the S-ROM.

Unlike the previous test case, the N-Adp-S-ROM did not exhibit large oscillations, instabilities or error values going towards infinity. The reason being that the minimization problem in (23) does not involve a time-stepping scheme—see Remark 5.3 above. This prevents error accumulation over time, which, along with a poor placement of the reduced mesh, was one of the reasons why the N-Adp-S-ROM was unstable in the previous study. For the N-Adp-S-ROM, there exist target parameters without a single reduced mesh element lying inside the support of the residual. As a result, the solution to the minimization problem (23) is zero but the approximation is not good.

Consider the average runtime $\mathcal{C}$ defined in (20). For the Adp-S-ROM, Fig. 5 plots the runtime and the speedup against the error. A few observations follow. (i) The error decreases monotonically upon increasing n, which is desirable. (ii) At best, for $n = 1.8 \times 10^3$, the Adp-S-ROM is 30 times faster and 1.3 times worse in accuracy than the S-ROM. (iii) At worst, for $n = 115.2 \times 10^3$, the Adp-S-ROM is almost 8.5 times faster (and similar in accuracy) than the S-ROM. (iv) Computing the FOM involves projecting a function onto the FV approximation space, which is a cheap operation. Therefore, none of the ROMs offer any speedup. We refer to the following test case that considers a more realistic scenario and presents the speedup offered by our hyper-reduction technique.

Table 5 Results for 2D transport

	N-Adp-S-ROM	Adp-S-ROM	S-ROM	L-ROM
$E(N_\mu, N_t)$	18.75×10^3	0.29	0.21	1.06

Computations performed with $N_\mu, N_t = 6, N_x = 800$, and $n = N_x \times 2 \times 10^{-2}$. Error comparison between the different ROMs listed in Table 1. The N-Adp-S-ROM showed large oscillations and appeared to be unstable, hence the extremely large error values

5.4 2D Collisionless radiative transport

We consider the 2D collisionless radiative transport equation given as [9]

$$\begin{aligned} &\partial_t u(x,t,\mu) + \cos(\mu)\partial_{x_1} u(x,t,\mu) \\ &\quad + \sin(\mu)\partial_{x_2} u(x,t,\mu) = 0, \quad \forall (x,t,\mu) \in \Omega \times D \times \mathcal{P}. \end{aligned} \tag{24}$$

The initial data reads

$$u_0(x,\mu) = \begin{cases} 1, & \|x\|_2 \leq 0.2 \\ 0, & \text{else} \end{cases}, \quad \forall \mu \in \mathcal{P}. \tag{25}$$

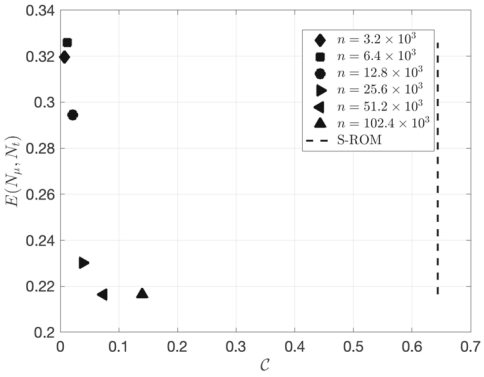
We set $\Omega = [-1, 1]^2$, $\mathcal{P} = [0, 2\pi]$, and $D = [0, 0.5]$.

We discretize Ω with a $N_x \times N_x$ Cartesian grid and we choose $N_x = 800$. We use a constant time step of $\Delta t = \Delta x/2$. We set $N_\mu = N_t = 6$. To collect snapshots of the residual, we take 5 uniformly placed samples from $\mathcal{P}$ i.e., $m_{hyp} = 5$. We study the ROM at the target parameters $\mathcal{Z}_{target} = \{(t_i, \tilde{\mu}_j)\}_{i,j}$. Here, $\{t_j\}_j$ represent the time-instances at which we compute the ROM, and $\{\tilde{\mu}_i\}_i$ are 50 uniformly placed samples inside $\mathcal{P}$.

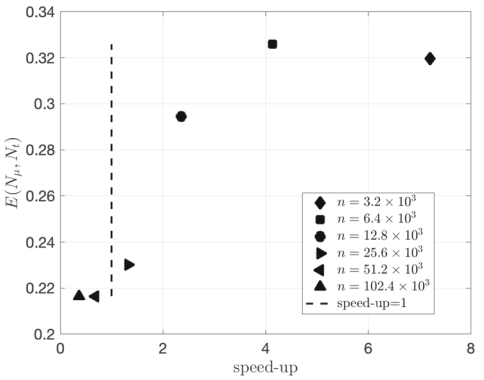
We choose a reduced mesh that contains 2% of the total mesh elements. Table 5 shows the error $E(N_\mu, N_t)$ for the different ROMs. We make the following observations. (i) Both the Adp-S-ROM and the S-ROM outperform the ROM. (ii) The maximum error resulting from the Adp-S-ROM is almost 1.3 times of that resulting from the S-ROM. Given that Adp-S-ROM is 50 times more efficient than the S-ROM—see the discussion below—we insist that the loss in accuracy introduced via hyper-reduction is acceptable. (iii) The N-Adp-S-ROM shows large oscillations resulting in large error values. Nonetheless, same as earlier, increasing the size of the reduced mesh removes these instabilities and provides an acceptable accuracy.

5.4.1 Runtime versus the error

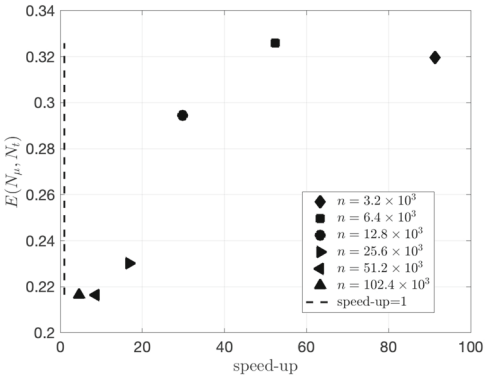
For the Adp-S-ROM, Fig. 6 plots the average runtime and the speedup versus the error $E(N_\mu, N_t)$. We make the following observations. (i) Increasing n increases the runtime and decreases the speedup, which is as expected. Beyond $n = 25.6 \times 10^3$, as compared to the FOM, the Adp-S-ROM does not offer any speedup. (ii) The lowest runtime and the maximum speedup of 7.8 corresponds to a reduced mesh that contains 0.5% of the total mesh elements. The relative error is 0.32, which is one-third of that resulting from the L-ROM and is 1.5 times of that resulting from the S-ROM—see Table 5. The accuracy loss as compared to the S-ROM is acceptable given that the Adp-S-ROM offers a speedup of two orders-of-magnitude.



(a) Error vs runtime



(b) Error vs speedup w.r.t to the FOM



(c) Error vs speedup w.r.t the S-ROM

Fig. 6 Results for test 2D transport, computed with the Adp-S-ROM. Computations performed with $N_\mu = N_t = 6$, and $N_x = 800$. See (16) and (20) for a definition of $\mathcal{E}(N_\mu, N_t)$ and $\mathcal{C}$, respectively. The dotted line indicates (a) the runtime of the S-ROM, (b) and (c) speedup of one

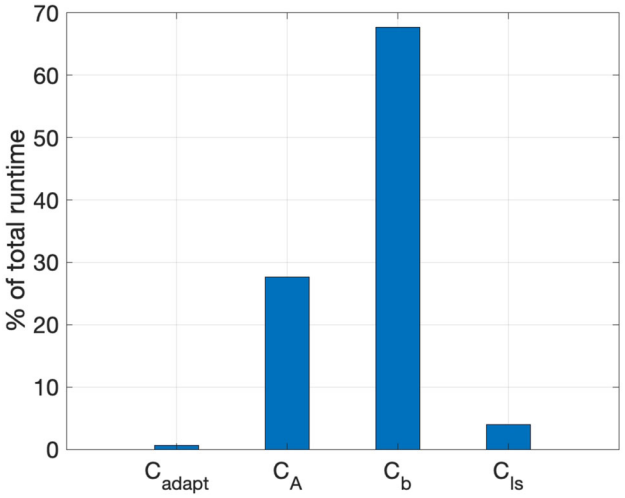


Fig. 7 Results for the 2D transport. Runtime split for the adaptive hyper-reduced ROM. Computations performed with $N_t = N_\mu = 6$

5.4.2 Runtime split

We split the runtime $\mathcal{C}$ into four major parts. (i) $\mathcal{C}_{adapt}$, the average runtime to adapt the reduced mesh, (ii) $\mathcal{C}_A$, the average runtime to compute the matrix $A(t_{k+1}, \mu)$ (or $A[\mathcal{E}_{t_{k+1},\mu}(t_{k+1}, \mu)]$ in the case of hyper-reduction). (iii) $\mathcal{C}_b$, the average runtime to compute the vector $b(t_k, \mu)$. (iv) $\mathcal{C}_{ls}$, the average runtime to solve the least-squares problem in (13). For $N_x = 800$ and $n = N_x^2 \times 2 \times 10^{-2}$, Fig. 7 compares the different runtime for the `Adp-S-ROM`. By far, computing the vector $b(t_k, \mu)$ is the most expensive part of the algorithm—it takes almost 70% of the total runtime. It is noteworthy that the combined cost of solving the least-squares problem and adapting the reduced mesh is less than 10% of the total runtime. Although not shown in the plot, increasing N_x has almost no effect on the runtime.

6 Conclusions

We propose an adaptive hyper-reduction technique for the nonlinear reduced order modelling of transport dominated problems. Our nonlinear approximation space is a span of shifted snapshots and we seek a solution using residual minimization. Through a cost analysis, we conclude that residual minimization is (at least) as expensive as the full-order model. To reduce the cost of residual minimization, we perform residual minimization over a reduced mesh i.e., over a subset of the full mesh. Using numerical and analytical examples we show that, similar to the solution, the residual exhibits a transport-type behaviour. This makes the use of a fixed parameter-independent reduced mesh both inaccurate and inefficient. To account for the transport-type behaviour of the residual, we introduce adaptivity in the reduced mesh.

In the numerical example we restrict to a one dimensional parameter space and a grid sampling. However we can use any sampling and the techniques still work. We also restrict the discussion to a spatial transformation function that is just a spatial shift. However other transformation function and even a collection can be used without harming the adaptive

hyperreduction technique which speeds up the computation of the ROM but keeps most of the accuracy gained from introducing a nonlinear ansatz space.

Empirically, we establish that for the same size of the reduced mesh, the adaptive technique greatly outperforms a non-adaptive technique. For a sufficiently small reduced mesh—almost 1% to 2% the size of the full mesh—the adaptive technique provides reasonable accuracy. In contrast, for such small sizes of the reduced mesh, the non-adaptive technique lead to an unstable reduced-order model, resulting in oscillations and extremely large error values. Nonetheless, at the expense of a high computational cost, increasing the size of the reduced mesh to about 20–50% of the total mesh size improved the accuracy of the non-adaptive technique.

In future work in particular the issue on compact solutions needs to be addressed more carefully. One should considered parametrized boundary conditions as well as periodic boundary conditions. Those can be included within the framework with some work. Numerical test for more complicated nonlinear functions including different sampling strategies together with a collection of spatial transformation functions should be included as well.

Author Contributions Both authors worked out the methods and wrote on the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. The authors thank the German Federal Ministry for Economic Affairs and Energy, in the joint project: "MathEnergy—Mathematical Key Technologies for Evolving Energy Grids", sub-project: Model Order Reduction (Grant number: 0324019B).

Data availability The study does not use data sets or other material.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose. All authors contributed to the methodology and writing of the paper.

References

1. Abgrall, R., Amsallem, D., Crisovan, R.: Robust model reduction by L1-norm minimization and approximation via dictionaries: application to nonlinear hyperbolic problems. Adv. Model. Simul. Eng. Sci. **3**, 1 (2016)
2. Astrid, P.: Fast reduced order modeling technique for large scale LTV systems. In: Proceedings of the 2004 American Control Conference, vol. 1, pp. 762–767 (2004)
3. Astrid, P., Weiland, S., Willcox, K., Backx, T.: Missing point estimation in models described by proper orthogonal decomposition. IEEE Trans. Autom. Control **53**, 2237–2251 (2008)
4. Benner, P., Gugercin, S., Willcox, K.: A survey of projection-based model reduction methods for parametric dynamical systems. SIAM Rev. **57**, 483–531 (2015)
5. Benner, P., Ohlberger, M., Cohen, A., Willcox, K.: Model Reduction and Approximation. Society for Industrial and Applied Mathematics, Philadelphia (2017)
6. Black, F., Schulze, P., Unger, B.: Projection-based model reduction with dynamically transformed modes. ESAIM Math. Model. Numer. Anal. **54**, 2011–2043 (2020)

7. Cagniart, N., Maday, Y., Stamm, B.: Model order reduction for problems with large convection effects. In: Contributions to Partial Differential Equations and Applications, pp. 131–150. Springer International Publishing, Cham (2019)
8. Carlberg, K., Farhat, C., Cortial, J., Amsallem, D.: The GNAT method for non-linear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows. J. Comput. Phys. **242**, 623–647 (2013)
9. Cercignani, C.: The Boltzmann Equation and Its Applications, vol. 67. Springer, Berlin (1988)
10. Dahmen, W., Plesken, C., Welper, G.: Double greedy algorithms: reduced basis methods for transport dominated problems. ESAIM: M2AN **48**, 623–663 (2014)
11. Ehrlacher, V., Lombardi, D., Mula, O., Vialard, F.-X.: Nonlinear model reduction on metric spaces. Application to one-dimensional conservative PDEs in Wasserstein spaces. ESAIM: Math. Model. Numer. Anal. **54**, 2159–2197 (2020)
12. Grundel, S., Herty, M.: Model-order reduction for hyperbolic relaxation systems. Int. J. Nonlinear Sci. Numer. Simul. **24**(7) (2022). https://doi.org/10.1515/ijnsns-2021-0192
13. Krivodonova, L., Xin, J., Remacle, J.-F., Chevaugeon, N., Flaherty, J.: Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws. Appl. Numer. Math. **48**, 323–338 (2004)
14. Venturi, F.B.L., Rozza, G.: A weighted pod method for elliptic PDEs with random inputs. J. Sci. Comput. **81**, 136–153 (2019)
15. LeVeque, R.J.: Finite Volume Methods for Hyperbolic Problems, Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge (2002)
16. Mojgani, R., Balajewicz, M.: Low-rank registration based manifolds for convection-dominated PDEs. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 399–407 (2021)
17. Nair, N.J., Balajewicz, M.: Transported snapshot model order reduction approach for parametric, steady-state fluid flows containing parameter-dependent shocks. In. J. Numer. Methods Eng. **117**, 1234–1262 (2019)
18. Peherstorfer, B.: Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling (2018). arXiv:1812.02094
19. Quarteroni, A., Manzoni, A., Negri, F.: Reduced Basis Methods for Partial Differential Equations: An Introduction. Springer International Publishing, Cham (2016)
20. Ray, D., Hesthaven, J.S.: Detecting troubled-cells on two-dimensional unstructured grids using a neural network. J. Comput. Phys. **397**, 108845 (2019)
21. Reiss, J., Schulze, P., Sesterhenn, J., Mehrmann, V.: The shifted proper orthogonal decomposition: a mode decomposition for multiple transport phenomena. SIAM J. Sci. Comput. **40**, A1322–A1344 (2018)
22. Rim, D., Mandli, K.T.: Displacement interpolation using monotone rearrangement. SIAM/ASA J. Uncertain. Quantif. **6**, 1503–1531 (2018)
23. Rim, D., Moe, S., LeVeque, R.J.: Transport reversal for model reduction of hyperbolic partial differential equations. SIAM/ASA J. Uncertain. Quantif. **6**, 118–150 (2018)
24. Rim, D., Peherstorfer, B., Mandli, K.T.: Manifold approximations via transported subspaces: model reduction for transport-dominated problems (2019). arXiv:1912.13024
25. Rowley, C.W., Marsden, J.E.: Reconstruction equations and the Karhunen–Loéve expansion for systems with symmetry. Physica D Nonlinear Phenom. **142**, 1–19 (2000)
26. Taddei, T.: A registration method for model order reduction: data compression and geometry reduction. SIAM J. Sci. Comput. **42**, A997–A1027 (2020)
27. Taddei, T., Perotto, S., Quarteroni, A.: Reduced basis techniques for nonlinear conservation laws. ESAIM: M2AN **49**, 787–814 (2015)
28. Torlo, D.: Model reduction for advection dominated hyperbolic problems in an ale framework: Offline and online phases, arXiv preprint (2020). arXiv:2003.13735
29. Vuik, M.J., Ryan, J.K.: Multiwavelet troubled-cell indicator for discontinuity detection of discontinuous Galerkin schemes. J. Comput. Phys. **270**, 138–160 (2014)
30. Wasilkowski, G.W., Wozniakowski, H.: Explicit cost bounds of algorithms for multivariate tensor product problems. J. Complex. **11**(1), 1–56 (1995)
31. Welper, G.: Interpolation of functions with parameter dependent jumps by transformed snapshots. SIAM J. Sci. Comput. **39**, A1225–A1250 (2017)
32. Welper, G.: Transformed snapshot interpolation with high resolution transforms. SIAM J. Sci. Comput. **42**, A2037–A2061 (2020)