DRIPS: A framework for Dimension Reduction and Interpolation in Parameter Space

Hannah Lu[a,1], Daniel M. Tartakovsky[a,*]

[a] *Department of Energy Science and Engineering, Stanford University, Stanford, CA 94305, USA*

Abstract

Reduced-order models are often used to describe the behavior of complex systems, whose simulation with a full model is too expensive, or to extract salient features from the full model's output. We introduce a new model-reduction framework DRIPS (dimension reduction and interpolation in parameter space) that combines the offline local model reduction with the online parameter interpolation of reduced-order bases (ROBs). The offline step of this framework relies on dynamic mode decomposition (DMD) to build a low-rank linear surrogate model, equipped with a local ROB, for quantities of interest derived from the training data generated by repeatedly solving the (nonlinear) high-fidelity model for multiple parameter points. The online step consists of the construction of a parametric reduced-order model for each target/test point in the parameter space, with the interpolation of ROBs done on a Grassman manifold and the interpolation of reduced-order operators done on a matrix manifold. The DMD component enables DRIPS to model (typically low-dimensional) quantities of interest directly, without having to access the (typically high-dimensional and possibly nonlinear) operators in a high-fidelity model that governs the dynamics of the underlying high-dimensional state variables, as required in projection-based reduced-order modeling. A series of numerical experiments suggests that DRIPS yields a model reduction, which is computationally more efficient than the commonly used projection-based proper orthogonal decomposition; it does so without requiring a prior knowledge of the governing equation for quantities of interest. Moreover, for the nonlinear systems considered, DRIPS is more accurate than Gaussian-process interpolation (Kriging).

Keywords: reduced-order modeling; data-driven learning; parametric systems; manifold interpolations

1. Introduction

Process-based models, typically based on appropriate conservation laws, play a central role in science and engineering. Rapid advances in software and hardware development, including high-performance computing, have led to the proliferation of more complex models that account for more (multiscale)

*Corresponding author

Email address: tartakovsky@stanford.edu (Daniel M. Tartakovsky)

[1] email: hannahlu@stanford.edu

phenomena and are solved on finer numerical meshes. Consequently, high-fidelity simulations of many complex systems remain a demanding and often elusive task. That is especially so in ensemble-based computations (e.g., global sensitivity analysis, uncertainty quantification, inverse modeling, and optimization) that require many solves of a model for different combinations of the input parameters. The high computational cost also precludes the use of such models to control a system's dynamic behavior in real time.

Model reduction techniques are designed to alleviate the (prohibitively) high computational cost of physics-based high-fidelity models (HFMs), while capturing the essential features of the underlying dynamics. Such techniques for the construction of reduced-order models (ROMs) can be grouped in two general categories, projection-based methods and data-driven methods. The former category relies on a Galerkin projection from a HFM of the system states onto a representative surrogate model on a low-dimensional "trial" subspace. Examples of these techniques are proper orthogonal decomposition (POD) [1, 2, 3] and discrete empirical interpolation (DEIM) [4] designed for linear and nonlinear systems, respectively. Both rely on the singular value decomposition (SVD) analysis in the time domain to obtain linear trial subspaces; related techniques in the frequency domain include balanced truncation [5, 6] and rational interpolation [7, 8]. Nonlinear trial manifolds can be obtained using deep convolutional autoencoders [9]. The computational saving of such methods stems from replacing the high-dimensional full system with its lower-dimensional counterpart that is used to make predictions. The resulting ROMs are physics-based in the sense that they inherit the dynamic operator from the projection.

Data-driven ROMs reduce the complexity of a HFM by learning the dynamics of the state variables or the derived quantities of interest (QoIs) directly from the HFM's output and/or observational data. Techniques from this category include Gaussian process regression (also known as Kriging or response surface estimation) [10, 11], random forest (RF) [12, 13], dynamic mode decomposition (DMD) [14, 15], operator inference [16, 17] and neural networks (NN) [18, 19]. The flexibility and robustness of this kind of the data-driven learning of ROMs can be enhanced by the infusion of physics ingredients, especially when data accessibility is limited [20, 21, 22, 23, 24, 25, 26, 27].

Many complex systems are designed and analyzed in terms of their dependency on the system/model parameters representing, e.g., variations in shape, material properties, loading, and boundary and initial conditions. Regardless of the strategy used for its construction, a ROM designed for one combination of the input parameters is not guaranteed to be accurate for another combination. Parametric ROMs or PROMs strive to enhance ROM's robustness and generalizability with respect to the choice of parameters. The limited generalizability of projection-based ROMs is well understood [28, 29, 30, 31] and ameliorated by PROMs that interpolate, on suitable manifolds, either a reduced-order basis (ROB) [32] or the underlying ROM [33]; subsequent improvements in interpolation include [34, 35, 36]. The projection-based PROMs are "physics hungry", because the projections can only be designed for the full state variables

with full knowledge of the dynamic operators. This complicates their application to multi-physics nonlinear problems with a large number of state variables that give rise to extremely high dimension after spatial discretization.

On the other hand, data-driven ROMs might fail to generalize to other parameter values when the time evolution of QoIs, expressed as function of the parameters and simulated predictors, is highly nonlinear or/and exhibits bifurcations. Conventional approximators like Gaussian process regression are known to struggle in capturing the dynamics precisely [33], an observation verified in our numerical example in Section 4.3. Nonlinear machine learning methods, such as RF and NN, act as accurate PROMs but only when enough high-fidelity data for a large range of input parameters are available for training [37, 38, 39]. Other strategies for the data-driven construction of PROMs include operator inference [17], which needs assumptions about polynomial structures of the underlying dynamics.

Given the aforementioned challenges, the development of efficient strategies to construct data-driven PROMs remains an active area of research. Our study contributes to this field by introducing a holistic PROM framework called DRIPS (dimension reduction and interpolation in parameter space), which combines the advantages of both categories of ROMs in two steps. The offline step builds a low-rank DMD-based (linear) ROM for observables of the QoIs at each training sample parameter point; the physics-guided selection of observables is based on the Koopman operator theory and not only increases the accuracy of the local ROMs but also provides a bridge between the understanding of data and physics. The online step constructs a new PROM with a corresponding ROB for each target/test point in the parameter space by interpolating the training PROMs and ROBs; the interpolation is done on a suitable manifold in the same fashion as in projection-based PROMs. Our framework obviates the need for both repeated access to expensive HFMs (the scourge of projection-based PROMs) and large amounts of data (the shortcoming of data-driven PROMs).

The remainder of this paper is organized as follows. Section 2 contains the problem formulation. In Section 3, we describe our general methodology and present algorithms for its implementation. Section 4 presents several numerical examples that demonstrate the accuracy and robustness of the DRIPS framework, with comparison to other methods. Key results, their implication for applications, and challenges and future work are summarized in Section 5.

2. Problem Formulation

We consider a multi-physics system described by N'_{sv} state variables $\mathbf{s}(\mathbf{x}, t; \mathbf{p}) = \{s_1, \ldots, s_{N'_{\text{sv}}}\}$, which vary in space $\mathbf{x} \in \mathcal{D}$ and time $t \in [0, T]$ throughout the simulation domain $\mathcal{D}$ during the simulation time interval $[0, T]$. For a given set of N_{par} parameters $\mathbf{p} = \{p_1, \ldots, p_{N_{\text{par}}}\} \in \Omega$ from the parameter range Ω, the spatiotemporal evolution of these state variables is described by a system of coupled partial-differential

equations (PDEs)

$$\frac{\partial s_i}{\partial t} = \phi_i(\mathbf{s};\mathbf{p}), \quad (\mathbf{x},t) \in \mathcal{D}\times(0,T]; \qquad i = 1,\dots,N'_{\rm sv}, \tag{2.1}$$

where ϕ_i are (linear/nonlinear) differential operators that contain spatial derivatives. When solved numerically, the spatial domain $\mathcal{D}$ is discretized into $N_{\rm el}$ elements/nodes, leading to a discretized state variable $\mathbf{S}(t;\mathbf{p})$ of high dimension $N_{\rm sv} = N'_{\rm sv} \times N_{\rm el}$, satisfying a system of coupled ordinary differential equations (ODEs)

$$\frac{\mathrm{d}\mathbf{S}}{\mathrm{d}t}(t;\mathbf{p}) = \mathbf{\Phi}(\mathbf{S}(t;\mathbf{p});\mathbf{p}), \tag{2.2}$$

where $\mathbf{\Phi}$ is a vector of linear/nonlinear functions.

More often than not, this model output has to be post-processed to compute N_Q QoIs $\mathbf{Q}(t;\mathbf{p}) = [Q_1(t;\mathbf{p}),\dots,Q_{N_Q}(t;\mathbf{p})]^\top$, such that

$$Q_i(t;\mathbf{p}) = \mathcal{M}_i(\mathbf{S}(t;\mathbf{p})), \quad \text{for} \quad i = 1,\dots,N_Q. \tag{2.3}$$

The maps $\mathcal{M}_i$ represent, e.g., numerical approximations of the integrals over $\mathcal{D}$ or a streamline. The QoIs are usually much easier to visualize and comprehend than the raw output $\mathbf{S}(t;\mathbf{p})$. In a typical application, $N_Q \ll N_{\rm sv}$ and, hence, it is more advantageous to solve the HFM for the QoIs $\mathbf{Q}(t;\mathbf{p})$,

$$\frac{\mathrm{d}\mathbf{Q}}{\mathrm{d}t}(t;\mathbf{p}) = \mathbf{F}(\mathbf{Q}(t;\mathbf{p});\mathbf{p}), \tag{2.4}$$

rather than the HFM (2.2) for the state variables $\mathbf{S}(t;\mathbf{p})$. The vector function $\mathbf{F}$ has a complicated, and potentially unknown form. The corresponding discrete-time dynamical system is described by

$$\mathbf{Q}(t_{k+1};\mathbf{p}) = \mathbf{F}_{\Delta t}(\mathbf{Q}(t_k;\mathbf{p});\mathbf{p}) := \mathbf{Q}(t_k;\mathbf{p}) + \int_{t_k}^{t_k+\Delta t} \mathbf{F}(\mathbf{Q}(\tau;\mathbf{p}))\mathrm{d}\tau, \tag{2.5}$$

for the uniform time discretization $t_k = k\Delta t \in [0,T]$ with $k = 0,\dots,N_T$.

The HFM model (2.2) and (2.3) is solved $N_{\rm MC}$ times for different combinations of the parameter values, i.e., points $\mathbf{p}^{(i)}$ $(i = 1,\dots,N_{\rm MC})$ in the parameter space Ω. Each solution is recorded as $N_{\rm snap}$ temporal snapshots (at times $t_0,\dots,t_{N_{\rm snap}}$) of the QoIs $\mathbf{Q}(t;\mathbf{p}^{(i)})$, forming a dataset

$$\{\mathbf{Q}(t_0;\mathbf{p}^{(1)}),\dots,\mathbf{Q}(t_{N_{\rm snap}};\mathbf{p}^{(1)}),\dots,\mathbf{Q}(t_0;\mathbf{p}^{(N_{\rm MC})}),\dots,\mathbf{Q}(t_{N_{\rm snap}};\mathbf{p}^{(N_{\rm MC})})\}, \quad N_{\rm snap} \le N_T. \tag{2.6}$$

Our goal is to construct from this dataset a reduced-dimension surrogate of the HFM (2.5). This will allow us to predict the trajectory of the QoIs $\mathbf{Q}(t;\mathbf{p}^*)$ at an unsampled parameter point $\mathbf{p}^* \notin \{\mathbf{p}^{(1)},\dots,\mathbf{p}^{(N_{\rm MC})}\}$, i.e., $\{\mathbf{Q}(t_0;\mathbf{p}^*),\dots,\mathbf{Q}(t_{N_T};\mathbf{p}^*)\}$, at a low cost without computing the complex HFM (2.2) and (2.3).

3. Methodology

Our DRIPS framework consists of the offline step and online step that are detailed in Sections 3.1 and 3.2, respectively.

3.1. Offline Step: DMD-Based Surrogates

For complex HFMs like (2.2) and (2.3), the dynamics of QoIs $\mathbf{Q}$, i.e., the functional form of $\mathbf{F}$ in (2.4) or $\mathbf{F}_{\Delta t}$ in (2.5) is generally unknown. In the first step of our algorithm, we replace the unknown discrete HFM (2.5) with its linear reduced-dimension surrogate constructed from the dataset (2.6). The latter task is facilitated by the Koopman operator theory, which allows one to handle the potential nonlinearity in the unknown dynamic $\mathbf{F}$ and $\mathbf{F}_{\Delta t}$:

Definition 3.1 (Koopman operator [15])**.** *For nonlinear dynamic system* (2.4)*, the Koopman operator* $\mathcal{K}^{\mathbf{p}}$ *is an infinite-dimensional linear operator that acts on all observable functions* $g : \mathcal{M} \to \mathbb{C}$ *so that*

$$\mathcal{K}^{\mathbf{p}} g(\mathbf{Q}(t;\mathbf{p})) = g(\mathbf{F}(\mathbf{Q}(t;\mathbf{p});\mathbf{p})). \tag{3.1}$$

For discrete dynamic system (2.5)*, the discrete-time Koopman operator* $\mathcal{K}^{\mathbf{p}}_{\Delta t}$ *is*

$$\mathcal{K}^{\mathbf{p}}_{\Delta t} g(\mathbf{Q}(t_{k+1};\mathbf{p})) = g(\mathbf{F}_{\Delta t}(\mathbf{Q}(t_k;\mathbf{p});\mathbf{p})) = g(\mathbf{Q}(t_{k+1};\mathbf{p})). \tag{3.2}$$

The Koopman operator transforms the finite-dimensional nonlinear problem (2.4) in the state space into the infinite-dimensional linear problem (3.1) in the observable space. Since $\mathcal{K}^{\mathbf{p}}_{\Delta t}$ is an infinite-dimensional linear operator, it is equipped with infinite eigenvalues $\{\lambda_k(\mathbf{p})\}_{k=1}^{\infty}$ and eigenfunctions $\{\phi_k(\mathbf{p})\}_{k=1}^{\infty}$. In practice, one has to make do with a finite number of the eigenvalues and eigenfunctions. The following assumption is essential to both a finite-dimensional approximation and the choice of observables.

Assumption 3.1. *Let* $\mathbf{y}(t_k;\mathbf{p})$ *denote an* $N \times 1$ *vector of observables,*

$$\mathbf{y}(t_k;\mathbf{p}) = \mathbf{g}(\mathbf{Q}(t_k;\mathbf{p})) = \begin{bmatrix} g_1(\mathbf{Q}(t_k;\mathbf{p})) \\ \vdots \\ g_N(\mathbf{Q}(t_k;\mathbf{p})) \end{bmatrix}, \tag{3.3}$$

where $g_j : \mathcal{M} \to \mathbb{C}$ *is an observable function with* $j = 1, \ldots, N$*. If the chosen observables* $\mathbf{g}$ *are restricted to an invariant subspace spanned by eigenfunctions of the Koopman operator* $\mathcal{K}^{\mathbf{p}}_{\Delta t}$*, then they induce a linear operator* $\mathbf{K}(\mathbf{p})$ *that is finite-dimensional and advances these eigen-observable functions on this subspace [40].*

Assumption 3.1 enables one to deploy the DMD Algorithm 1 to approximate the N-dimensional linear operator $\mathbf{K}(\mathbf{p})$ and its low-dimensional approximation $\mathbf{K}_r(\mathbf{p})$ of rank r. At each parameter point $\mathbf{p}^{(i)}$ with $i = 1, \ldots, N_{\mathrm{MC}}$, the discrete HFM (2.5) on state space is approximated by an N-dimensional linear surrogate model

$$\mathbf{y}(t_{k+1};\mathbf{p}^{(i)}) = \mathbf{K}(\mathbf{p}^{(i)})\mathbf{y}(t_k;\mathbf{p}^{(i)}) \tag{3.4}$$

on observable space. The two spaces are connected by the observable function $\mathbf{g}$ and its inverse $\mathbf{g}^{-1}$. Algorithm 1 directly induces the ROM for (3.4),

$$\mathbf{q}(t_{k+1};\mathbf{p}^{(i)}) = \mathbf{K}_r(\mathbf{p}^{(i)})\mathbf{q}(t_k;\mathbf{p}^{(i)}). \tag{3.5}$$

Here $\mathbf{q}(t_k;\mathbf{p}^{(i)})$ is the reduced-order observable vector of dimension r. In terms of a ROB $\mathbf{V}(\mathbf{p}^{(i)})$, these are expressed as

$$\mathbf{y}(t_k;\mathbf{p}^{(i)}) = \mathbf{V}(\mathbf{p}^{(i)})\mathbf{q}(t_k;\mathbf{p}^{(i)}) \quad \text{and} \quad \mathbf{K}_r(\mathbf{p}^{(i)}) = \mathbf{V}(\mathbf{p}^{(i)})^\star \mathbf{K}(\mathbf{p}^{(i)})\mathbf{V}(\mathbf{p}^{(i)}). \tag{3.6}$$

Algorithm 1: DMD algorithm on observable space [15] for parameter point $\mathbf{p}^{(i)}$, $i = 1, \ldots, N_{\mathrm{MC}}$.

Input: $\{\mathbf{Q}(t_0;\mathbf{p}^{(i)}), \ldots, \mathbf{Q}(t_{N_{\mathrm{snap}}};\mathbf{p}^{(i)})\}$, observable function $\mathbf{g}$

1. Create data matrices of the observables

$$\mathbf{Y}_1(\mathbf{p}^{(i)}) = \begin{bmatrix} | & & | \\ \mathbf{y}(t_0;\mathbf{p}^{(i)}) & \cdots & \mathbf{y}(t_{N_{\mathrm{snap}}-1};\mathbf{p}^{(i)}) \\ | & & | \end{bmatrix}, \quad \mathbf{Y}_2(\mathbf{p}^{(i)}) = \begin{bmatrix} | & & | \\ \mathbf{y}(t_1;\mathbf{p}^{(i)}) & \cdots & \mathbf{y}(t_{N_{\mathrm{snap}}};\mathbf{p}^{(i)}) \\ | & & | \end{bmatrix}, \tag{3.7}$$

   where each column is given by $\mathbf{y}(t_k;\mathbf{p}^{(i)}) = \mathbf{g}(\mathbf{Q}(t_k;\mathbf{p}^{(i)}))$.

2. Apply SVD $\mathbf{Y}_1(\mathbf{p}^{(i)}) \approx \mathbf{V}(\mathbf{p}^{(i)})\mathbf{\Sigma}(\mathbf{p}^{(i)})\mathbf{Z}(\mathbf{p}^{(i)})^\star$, where $\mathbf{V}(\mathbf{p}^{(i)}) \in \mathbb{C}^{N\times r}$, $\mathbf{\Sigma}(\mathbf{p}^{(i)}) \in \mathbb{C}^{r\times r}$, $\mathbf{Z}(\mathbf{p}^{(i)})^\star \in \mathbb{C}^{r\times N_{\mathrm{snap}}}$ is the conjugate transpose of $\mathbf{Z}(\mathbf{p}^{(i)})$, and r is the truncation rank chosen by certain criteria and kept the same for all $i = 1, \ldots, N_{\mathrm{MC}}$.
3. Compute $\mathbf{K}_r(\mathbf{p}^{(i)}) = \mathbf{V}(\mathbf{p}^{(i)})^\star \mathbf{Y}_2(\mathbf{p}^{(i)})\mathbf{Z}(\mathbf{p}^{(i)})\mathbf{\Sigma}(\mathbf{p}^{(i)})^{-1}$ as an $r \times r$ low-rank approximation of $\mathbf{K}(\mathbf{p}^{(i)})$.
4. Compute $\mathbf{P}^{(i,j)} = \mathbf{V}(\mathbf{p}^{(i)})^\star \mathbf{V}(\mathbf{p}^{(j)})$ for $i, j = 1, \ldots, N_{\mathrm{MC}}$.

Output: $\mathbf{V}(\mathbf{p}^{(i)})$, $\mathbf{K}_r(\mathbf{p}^{(i)})$ and $\mathbf{P}^{(i,j)}$.

Remark 3.1. The construction of DMD surrogates can be done offline, which allows one to precompute $\mathbf{P}^{(i,j)}$ for the later online step. Although this offline step takes a majority of the computational time in the whole framework, mostly due to the computation of the high-fidelity training data (2.6), its output can be precomputed and stored efficiently; the output storage is $(N \cdot r + r \cdot r + r \cdot r \cdot (N_{\mathrm{MC}} + 1)/2) \cdot N_{\mathrm{MC}}$.

Remark 3.2. A theorem in [41] establishes connections between the DMD theory and the Koopman spectral analysis under specific conditions on the observables and collected data. This theorem indicates that judicious selection of the observables is critical to the success of the Koopman method. There is no principled way to select observables without expert knowledge of a dynamical system. Machine learning techniques can be deployed to identify relevant terms in the dynamics from data, which then guides the

selection of the observables [42, 43]. In our numerical examples, we rely on knowledge of the underlying physics to select the observables, as was done, e.g., in [21, 22, 23, 24, 44]).

Remark 3.3. In general, the rank r is chosen to satisfy the energy criteria

$$r = \min_k \{ \frac{\sigma_k}{\sum_m \sigma_m} < \varepsilon \}$$

where σ_k are the diagonal elements of $\mathbf{\Sigma}$ in SVD, and ε is a user-supplied small number. Since r has to be kept the same for all training parameter points, we choose r in each problem heuristically by observing how the singular value decays in the training points.

3.2. Online Step: Interpolation of ROBs and PROMs

For an unsampled parameter point $\mathbf{p}^* \notin \{\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(N_{\mathrm{MC}})}\}$, the goal is to compute $\{\mathbf{Q}(t_1; \mathbf{p}^*), \ldots, \mathbf{Q}(t_{N_T}; \mathbf{p}^*)\}$ via the PROM

$$\mathbf{q}(t_{k+1}; \mathbf{p}^*) = \mathbf{K}_r(\mathbf{p}^*)\mathbf{q}(t_k; \mathbf{p}^*), \tag{3.8}$$

without evaluating the HFM (2.2) and (2.3). Subsequently, the observables $\mathbf{y}$ and the QoIs $\mathbf{Q}$ are estimated as

$$\mathbf{y}(t_k; \mathbf{p}^*) = \mathbf{V}(\mathbf{p}^*)\mathbf{q}(t_k; \mathbf{p}^*), \qquad \mathbf{Q}(t_k; \mathbf{p}^*) = \mathbf{g}^{-1}(\mathbf{y}(t_k; \mathbf{p}^*)). \tag{3.9}$$

Therefore, the online task comprises the computation of three quantities, $\mathbf{V}(\mathbf{p}^*)$, $\mathbf{K}_r(\mathbf{p}^*)$, and $\mathbf{q}(t_k; \mathbf{p}^*)$.

3.2.1. ROB Interpolation

We rely on interpolation on the Grassman manifold to compute the ROB $\mathbf{V}(\mathbf{p}^*)$ from the dataset $\{\mathbf{V}(\mathbf{p}^{(1)}), \ldots, \mathbf{V}(\mathbf{p}^{(N_{\mathrm{MC}})})\}$. We briefly review this interpolation approach [32] below.

Definition 3.2. *The following manifolds are of interest:*

- *Grassmann manifold $\mathcal{G}(r, N)$ is the set of all subspaces in $\mathbb{R}^N$ of dimension r;*
- *Stiefel manifold $\mathcal{ST}(r, N)$ is the set of orthogonal ROB matrices in $\mathbb{R}^{N \times r}$.*

The ROB $\mathbf{V}(\mathbf{p}^{(i)}) \in \mathbb{R}^{N \times r}$, with $i = 1, \ldots, N_{\mathrm{MC}}$ and $r \leq N$, is the full-rank column matrix, whose columns provide a basis of subspace $\mathcal{S}_i$ of dimension r in $\mathbb{R}^N$. While an associated ROM is not uniquely defined by the ROB, it is uniquely defined by the subspace $\mathcal{S}_i$. Therefore, the correct entity to interpolate is the subspace $\mathcal{S}_i$, rather than the ROB $\mathbf{V}(\mathbf{p}^{(i)})$. Hence, the goal is to compute $\mathcal{S}_* = \mathrm{range}(\mathbf{V}(\mathbf{p}^*))$ by interpolating between $\{\mathcal{S}_i\}_{i=1}^{N_{\mathrm{MC}}}$ with access to the ROB $\mathbf{V}(\mathbf{p}^*)$.

The subspaces $\mathcal{S}_1, \ldots, S_{N_{\mathrm{MC}}}$ belong to the Grassmann manifold $\mathcal{G}(r, N)$ [45, 46, 47, 48, 49] . Each r-dimensional subspace $\tilde{\mathcal{S}}$ of $\mathbb{R}^N$ is regarded as a point of $\mathcal{G}(r, N)$ and is nonuniquely represented by a matrix $\tilde{\mathbf{V}} \in \mathbb{R}^{N \times r}$, whose columns span the subspace $\tilde{\mathcal{S}}$. The matrix $\tilde{\mathbf{V}}$ is chosen among those whose columns form a set of orthonormal vectors in $\mathbb{R}^N$ and belong to the orthogonal Stiefel manifold $\mathcal{ST}(r, N)$ [45, 49].

There exists a projection map [45] from $\mathcal{ST}(r,N)$ onto $\mathcal{G}(r,N)$, as each matrix in $\mathcal{ST}(r,N)$ spans an r-dimensional subspace of $\mathbb{R}^N$ and different matrices can span the same subspace. At each point $\tilde{\mathcal{S}}$ of the manifold $\mathcal{G}(r,N)$, there exists a tangent space $\mathcal{T}_{\tilde{\mathcal{S}}}$ [45, 49] of the same dimension [49]. Each point in this space is represented by a matrix $\tilde{\mathbf{\Gamma}} \in \mathbb{R}^{N\times r}$. Since $\mathcal{T}_{\tilde{\mathcal{S}}}$ is a vector space, usual interpolation is allowed for the matrices representing its points. Let $\mathbf{\Gamma}^i = m_{\mathbf{V}}(\mathbf{V}(\mathbf{p}^{(i)}))$, where $m_{\mathbf{V}}$ denotes the map from the matrix manifolds $\mathcal{G}(r,N)$ onto the tangent space $\mathcal{T}_{\tilde{\mathcal{S}}}$. This suggests a strategy of computing $\mathbf{\Gamma}^*$ via usual interpolation between $\{\mathbf{\Gamma}^i\}_{i=1}^{N_{\mathrm{MC}}}$ and then evaluating $\mathbf{V}(\mathbf{p}^*)$ through the inverse map $m_{\mathbf{V}}^{-1}(\mathbf{\Gamma}^*)$.

The map $m_{\mathbf{V}}$ is chosen to be the logarithmic mapping, which maps the Grassmann manifold onto its tangent space, and $m_{\mathbf{V}}^{-1}$ is chosen to be the exponential mapping, which maps the tangent space onto the Grassmann manifold itself. This choice borrows concepts of geodesic path on a Grassmann manifold from differential geometry [45, 46, 50]. This strategy, discussed in detail in [32], is implemented in Algorithm 2.

Algorithm 2: Interpolation of ROBs [32, 51, 48]

Input: $\{\mathbf{V}(\mathbf{p}^{(i)})\}_{i=1}^{N_{\mathrm{MC}}}$, $\{\mathbf{P}^{(i,j)} = \mathbf{V}(\mathbf{p}^{(i)})^\star \mathbf{V}(\mathbf{p}^{(i)})\}_{i,j=1}^{N_{\mathrm{MC}}}$, $\{\mathbf{p}^{(i)}\}_{i=1}^{N_{\mathrm{MC}}}$ and target parameter point $\mathbf{p}^*$

1. Denote $\mathcal{S}_i = \mathrm{range}(\mathbf{V}(\mathbf{p}^{(i)}))$, for $i = 1,\dots N_{\mathrm{MC}}$. A point $\mathcal{S}_{i_0}$ with $i_0 \in \{1,\dots N_{\mathrm{MC}}\}$ of the manifold is chosen as a reference and origin point for interpolation.
2. Select points $\mathcal{S}_i$ with $i \in \mathcal{I}_{i_0} \subset \{1,\dots,N_{\mathrm{MC}}\}$, which lie in a sufficiently small neighborhood of $\mathcal{S}_{i_0}$, and use the logarithm map $\ln_{\mathcal{S}_{i_0}}$ to map $\{\mathcal{S}_i\}_{i\in\mathcal{I}_{i_0}}$ onto matrices $\{\mathbf{\Gamma}^i\}_{i\in\mathcal{I}_{i_0}}$ representing the corresponding points of $\mathcal{T}_{\mathcal{S}_{i_0}}$. This is computed as
$$\begin{aligned} &(\mathbf{I} - \mathbf{V}(\mathbf{p}^{(i_0)})\mathbf{V}(\mathbf{p}^{(i_0)})^\star)\mathbf{V}(\mathbf{p}^{(i)})(\mathbf{P}^{(i_0,i)})^{-1} = \mathbf{U}_i\mathbf{\Omega}_i\mathbf{W}_i^\star, \quad \text{(thin SVD)} \\ &\mathbf{\Gamma}^i = \mathbf{U}_i \tan^{-1}(\mathbf{\Omega}_i)\mathbf{W}_i^\star. \end{aligned} \tag{3.10}$$
3. Compute $\mathbf{\Gamma}^*$ by interpolating $\{\mathbf{\Gamma}^i\}_{i\in\mathcal{I}_{i_0}}$ entry by entry:
$$\Gamma_{ij}^* = \mathcal{P}(\mathbf{p}^*; \{\Gamma_{ij}^i, \mathbf{p}^{(i)}\}_{i\in\mathcal{I}_{i_0}}), \quad i = 1,\dots,N, \quad j = 1,\dots,r. \tag{3.11}$$
4. Use the exponential map $\exp_{\mathcal{S}_{i_0}}$ to map the matrix $\mathbf{\Gamma}^*$, representing a point of $\mathcal{T}_{\mathcal{S}_{i_0}}$, onto the desired subspace $\mathcal{S}_*$ on $\mathcal{G}(r,N)$ spanned by the ROB $\mathbf{V}(\mathbf{p}^*)$. This is computed as
$$\begin{aligned} &\mathbf{\Gamma}^* = \mathbf{U}_*\mathbf{\Omega}_*\mathbf{W}_*^\star, \quad \text{(thin SVD)} \\ &\mathbf{V}(\mathbf{p}^*) = \mathbf{V}(\mathbf{p}^{(i_0)})\mathbf{W}_* \cos(\mathbf{\Omega}_*)\mathbf{W}_*^\star + \mathbf{U}_* \sin(\mathbf{\Omega}_*)\mathbf{W}_*^\star. \end{aligned} \tag{3.12}$$

Output: $\mathbf{V}(\mathbf{p}^*)$

Remark 3.4. The choice of the interpolation method $\mathcal{P}$ depends on the dimension of the parameter space, N_{par}. If $N_{\mathrm{par}} = 1$, a univariate (typically, a Lagrange type) interpolation method is chosen. Otherwise, a multivariate interpolation scheme (see, e.g., [52, 53]) is chosen.

Remark 3.5. Since the logarithmic map $\ln_{\mathcal{S}_{i_0}}$ is defined in a neighborhood of $\mathcal{S}_{i_0} \in \mathcal{G}(r, N)$, the method is expected to be insensitive to the choice of the reference point $\mathcal{S}_{i_0}$ in step 1 of Algorithm 2. This is confirmed in numerical experiments [32].

3.2.2. PROM Interpolation

The reduced-order operator $\mathbf{K}_r(\mathbf{p}^*)$ in (3.8) is computed via interpolation on the matrix manifold between the ROMs $\{\mathbf{K}_r(\mathbf{p}^{(1)}), \ldots, \mathbf{K}_r(\mathbf{p}^{(N_{\mathrm{MC}})})\}$. This is done in two steps [33]:

- Step A). Since any ROM can be endowed with multiple alternative ROBs, the resulting ROMs may have been computed in different generalized coordinates system. The validity of an interpolation may crucially depend on the choice of the representative element within each equivalent class. Given the precomputed ROMs $\{\mathbf{K}_r(\mathbf{p}^{(1)}), \ldots, \mathbf{K}_r(\mathbf{p}^{(N_{\mathrm{MC}})})\}$, a set of congruence transformations is determined so that a representative element of the equivalent ROBs for each precomputed ROM is chosen to assign the precomputed ROMs into consistent sets of generalized coordinates. The consistency is enforced by solving the orthogonal Procrustes problems [54],

$$\min_{\mathbf{Q}_i, \mathbf{Q}_i^\star \mathbf{Q}_i = \mathbf{I}_r} \|\mathbf{V}(\mathbf{p}^{(i)})^\star \mathbf{Q}_i - \mathbf{V}(\mathbf{p}^{(i_0)})\|_F, \qquad \forall i = 1, \ldots, N_{\mathrm{MC}}, \tag{3.13}$$

  where $i_0 \in \{1, \ldots N_{\mathrm{MC}}\}$ is chosen as a reference configuration. The representative element is identified by solving the above problem with a direct procedure. This procedure is summarized in Algorithm 3.

Algorithm 3: Step A of the PROM interpolation [32]

Input: $\{\mathbf{K}_r(\mathbf{p}^{(1)}), \ldots, \mathbf{K}_r(\mathbf{p}^{(N_{\mathrm{MC}})})\}$, $\{\mathbf{P}^{(i,j)} = \mathbf{V}(\mathbf{p}^{(i)})^\star \mathbf{V}(\mathbf{p}^{(i)})\}_{i,j=1}^{N_{\mathrm{MC}}}$, reference configuration choice i_0

For $i \in \{1, \ldots N_{\mathrm{MC}}\} \setminus \{i_0\}$

- Compute $\mathbf{P}^{(i,i_0)} = \mathbf{U}_{i,i_0} \mathbf{\Sigma}_{i,i_0} \mathbf{Z}_{i,i_0}^\star$ (SVD),
- Compute $\mathbf{Q}_i = \mathbf{U}_{i,i_0} \mathbf{Z}_{i,i_0}^\star$,
- Transform $\tilde{\mathbf{K}}_r(\mathbf{p}^{(i)}) = \mathbf{Q}_i^\star \mathbf{K}_r(\mathbf{p}^{(i)}) \mathbf{Q}_i$

End

Output: $\{\tilde{\mathbf{K}}_r(\mathbf{p}^{(1)}), \ldots, \tilde{\mathbf{K}}_r(\mathbf{p}^{(N_{\mathrm{MC}})})\}$

Remark 3.6. An optimal choice of the reference configuration i_0, if it exists, remains an open problem.

Remark 3.7. Algorithm 3 is related to mode-tracking procedures based on the modal assurance criterion (MAC) [55]. This connection is elucidated in [33].

- Step B). The transformed ROMs $\{\tilde{\mathbf{K}}_r(\mathbf{p}^{(1)}), \ldots, \tilde{\mathbf{K}}_r(\mathbf{p}^{(N_{\mathrm{MC}})})\}$ are interpolated to compute a ROM $\mathbf{K}_r(\mathbf{p}^*)$. Similar to Section 3.2.1, this interpolation must be performed on a specific manifold containing both $\{\tilde{\mathbf{K}}_r(\mathbf{p}^{(1)}), \ldots, \tilde{\mathbf{K}}_r(\mathbf{p}^{(N_{\mathrm{MC}})})\}$ and $\mathbf{K}_r(\mathbf{p}^*)$, so that the distinctive properties (e.g., orthogonality, nonsingularity) are preserved. The main idea again is to first map all the precomputed matrices onto the tangent space to the matrix manifold of interest at a chosen reference point using the logarithm mapping, then interpolate the mapped data in this linear vector space, and finally map the interpolated result back onto the manifold of interest using the associated exponential map. This is done in Algorithm 4.

Algorithm 4: Step B of the PROM interpolation [32]

Input: $\{\tilde{\mathbf{K}}_r(\mathbf{p}^{(1)}), \ldots, \tilde{\mathbf{K}}_r(\mathbf{p}^{(N_{\mathrm{MC}})})\}$, reference configuration choice i_0

1. **For** $i \in \{1, \ldots N_{\mathrm{MC}}\} \setminus \{i_0\}$
   - Compute $\mathbf{\Gamma}^i = \ln_{\tilde{\mathbf{K}}_r(\mathbf{p}^{(i_0)})}(\tilde{\mathbf{K}}_r(\mathbf{p}^{(i)}))$

   End
2. Compute $\mathbf{\Gamma}^*$ by interpolating $\{\mathbf{\Gamma}^i\}_{i \in \mathcal{I}_{i_0}}$ entry by entry, as in (3.11)
3. Compute $\mathbf{K}_r(\mathbf{p}^*) = \exp_{\tilde{\mathbf{K}}_r(\mathbf{p}^{(i_0)})}(\mathbf{\Gamma}^*)$

Output: $\mathbf{K}_r(\mathbf{p}^*)$

Remark 3.8. The ln and exp in Algorithm 4 denote the matrix logarithm and exponential respectively. The specific expressions of different matrix manifolds of interest are listed in Table 4.1 of [33]. In general, the set $\{\mathbf{K}_r(\mathbf{p}^{(1)}), \ldots, \mathbf{K}_r(\mathbf{p}^{(N_{\mathrm{MC}})})\}$ obtained from DMD (Algorithm 1) does not have a special structure (e.g., symmetric positive definite), so we choose the interpolation on a matrix manifold for general linear group, i.e., $\ln_{\mathbf{X}}(\mathbf{Y}) = \mathbf{Y} - \mathbf{X}$ and $\exp_{\mathbf{X}}(\mathbf{\Gamma}) = \mathbf{X} + \mathbf{\Gamma}$.

Remark 3.9. We choose the same reference point i_0 for Algorithms 2–4, so that $\mathbf{V}(\mathbf{p}^*)$ is interpolated from the reference coordinate of $\mathbf{V}(\mathbf{p}^{(i_0)})$ and corresponds to $\mathbf{K}_r(\mathbf{p}^*)$ in the same consistent generalized coordinate. If the reference point i_0 is chosen differently for Algorithm 2 and Algorithms 3 or 4, then $\mathbf{K}_r(\mathbf{p}^*)$ needs to be adjusted in a way similar to Step A (Algorithm 3).

3.2.3. Iteration-free computation of the solution

With $\mathbf{K}_r(\mathbf{p}^*)$ and $\mathbf{V}(\mathbf{p}^*)$ computed, we use Algorithm 5 to obtain the solution using the iteration-free feature of DMD framework.

3.3. Algorithm Summary

The proposed DRIPS framework is implemented in Algorithm 6.

Algorithm 5: DMD reconstruction

Input: $\mathbf{K}_r(\mathbf{p}^*)$, $\mathbf{V}(\mathbf{p}^*)$, $\mathbf{y}(t=0;\mathbf{p}^*)$, $\mathbf{g}$

1. Compute the eigen-decomposition of $\mathbf{K}_r(\mathbf{p}^*)$:

$$\mathbf{K}_r(\mathbf{p}^*)\mathbf{\Psi}(\mathbf{p}^*) = \mathbf{\Psi}(\mathbf{p}^*)\mathbf{\Lambda}(\mathbf{p}^*), \tag{3.14}$$

   where columns of $\mathbf{\Psi}(\mathbf{p}^*)$ are eigenvectors and $\mathbf{\Lambda}(\mathbf{p}^*)$ is a diagonal matrix containing the corresponding eigenvalues λ_i with $i = 1, \ldots, r$.

2. Compute the DMD modes

$$\mathbf{\Phi}(\mathbf{p}^*) = \mathbf{V}(\mathbf{p}^*)\mathbf{\Psi}(\mathbf{p}^*). \tag{3.15}$$

3. Reconstruct the observables:

$$\mathbf{y}_{\mathrm{DMD}}(t_k;\mathbf{p}^*) = \mathbf{\Phi}(\mathbf{p}^*)\mathbf{\Lambda}(\mathbf{p}^*)^k \left(\mathbf{\Phi}(\mathbf{p}^*)^\dagger \mathbf{y}(0;\mathbf{p}^*)\right), \qquad k = 1, \ldots, N_T, \tag{3.16}$$

   where † is Moore–Penrose pseudoinverse.

4. Map the observables back to state space:

$$\mathbf{Q}_{\mathrm{DRIPS}}(t_k;\mathbf{p}^*) = \mathbf{g}^{-1}(\mathbf{y}_{\mathrm{DMD}}(t_k;\mathbf{p}^*)) \tag{3.17}$$

Output: $\mathbf{Q}_{\mathrm{DRIPS}}(t_k;\mathbf{p}^*)$

Algorithm 6: DRIPS framework

Offline Step:

For $i = 1, \ldots, N_{\mathrm{MC}}$, compute the high fidelity training data (2.6),

Input: $\{\mathbf{Q}(t_1;\mathbf{p}^{(i)}), \ldots, \mathbf{Q}(t_{N_{\mathrm{snap}}};\mathbf{p}^{(i)})\}$ and $\mathbf{g}$ $\xrightarrow{\text{Algorithm 1}}$ Output: $\mathbf{V}(\mathbf{p}^{(i)}), \mathbf{K}_r(\mathbf{p}^{(i)})$ and $\mathbf{P}^{(i,j)}$

End

Online Step:

• Interpolation of ROBs:

Input: $\{\mathbf{V}(\mathbf{p}^{(i)})\}_{i=1}^{N_{\mathrm{MC}}}$, $\{\mathbf{P}^{(i,j)}\}_{i,j=1}^{N_{\mathrm{MC}}}$, $\{\mathbf{p}^{(i)}\}_{i=1}^{N_{\mathrm{MC}}}$, $\mathbf{p}^*$ $\xrightarrow{\text{Algorithm 2}}$ Output: $\mathbf{V}(\mathbf{p}^*)$

• Interpolation of PROMs:

Input: $\{\mathbf{K}_r(\mathbf{p}^{(i)})\}_{i=1}^{N_{\mathrm{MC}}}$, $\{\mathbf{P}^{(i,j)}\}_{i,j=1}^{N_{\mathrm{MC}}}$, reference choice i_0 $\xrightarrow{\text{Algorithms 3 \& 4}}$ Output: $\mathbf{K}_r(\mathbf{p}^*)$

• DMD reconstruction:

Input: $\mathbf{K}_r(\mathbf{p}^*)$, $\mathbf{V}(\mathbf{p}^*)$, $\mathbf{y}(t=0;\mathbf{p}^*)$, $\mathbf{g}$ $\xrightarrow{\text{Algorithm 5}}$ Output: $\mathbf{Q}_{\mathrm{DRIPS}}(t_k;\mathbf{p}^*)$

Remark 3.10. The sampling strategy for $\{\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(N_{\mathrm{MC}})}\}$ in the parameter space plays a key role in the accuracy of the subspace approximation. The so-called "curse of dimensionality", i.e., the number of

training samples N_{MC} needed grows exponentially with the number of parameters, N_{par}, is a well-known challenge. In general, uniform sampling is used for $N_{\mathrm{par}} \leq 5$ and moderately computationally intensive HFMs, latin hypercube sampling is used for $N_{\mathrm{par}} > 5$ and moderately computationally intensive HFMs, and adaptive, goal-oriented, greedy sampling is used for $N_{\mathrm{par}} > 5$ and highly computationally intensive HFMs. We limit our numerical experiments to $N_{\mathrm{par}} = 3$ for simplicity, leaving the challenge posed by the curse of dimensionality for future studies.

4. Numerical Experiments

We use a series of numerical experiments to demonstrate the performance of DRIPS. In Section 4.1, we consider a linear advection-diffusion problem, in which the dynamical operators in the HFM are accessible for the construction of POD-PROM. Our DRIPS algorithm, i.e., DMD-PROM, is compared with the conventional projection-based POD-PROM to illustrate the benefits of knowing the physics in constructing accurate ROMs. Section 4.2 deals with a problem of linear heat conduction in a complex domain, which is solved by the Matlab PDE toolbox that provides a linear HFM without an explicit form. Unlike POD-PROM, DRIPS requires no large differential matrices and embeddings of complex boundary conditions; consequently it exhibits advantageous performance in constructing accurate ROMs for different QoIs. In Section 4.3, a nonlinear problem is used to demonstrate that our physics-aware framework DRIPS outperforms the pure data-driven approach Kriging and serves as a viable reduced-order modeling tool.

Kriging, aka Gaussian Process (GP) surrogates, provides a popular alternative to the DRIPS framework. Kriging interpolates the QoI solution map pixel by pixel based on prior covariances, which depend on the distance between the target point and the sampled points in the parameter space. It is a pure data-driven approach in the sense that it does not account for the underlying governing equations. Although Kriging is widely used in many applications (mostly for linear problems), such a pure data-driven approach may fail to detect the bifurcations in a complex dynamic system [33]. We will verify this observation by comparing the performance of Kriging and DRIPS in a series of numerical experiments. The GP surrogate is constructed at each time instance, with $N_{\mathrm{par}} = 3$ parameter values as input and the QoI at each time step as (N_Q-dimensional) output. We use a radial basis function kernel with default length scale 1 and default bound $(10^{-5}, 10^5)$.

In the following numerical tests, the performance of DRIPS, POD-PROM, and GP is quantified in

terms of the relative L_2 errors,

$$\begin{aligned}
\varepsilon_{\text{POD}}(t;\mathbf{p}^*) &= \frac{\|\mathbf{Q}_{\text{POD}}(t;\mathbf{p}^*) - \mathbf{Q}(t;\mathbf{p}^*)\|_2}{\|\mathbf{Q}(t;\mathbf{p}^*)\|_2}, \\
\varepsilon_{\text{GP}}(t;\mathbf{p}^*) &= \frac{\|\mathbf{Q}_{\text{GP}}(t;\mathbf{p}^*) - \mathbf{Q}(t;\mathbf{p}^*)\|_2}{\|\mathbf{Q}(t;\mathbf{p}^*)\|_2}, \\
\varepsilon_{\text{DRIPS}}(t;\mathbf{p}^*) &= \frac{\|\mathbf{Q}_{\text{DRIPS}}(t;\mathbf{p}^*) - \mathbf{Q}(t;\mathbf{p}^*)\|_2}{\|\mathbf{Q}(t;\mathbf{p}^*)\|_2},
\end{aligned} \tag{4.1}$$

and total relative L_2 errors

$$\begin{aligned}
\mathcal{E}_{\text{POD}}(\mathbf{p}^*) &= \frac{\sqrt{\sum_{k=1}^{N_T} \|\mathbf{Q}_{\text{POD}}(t_k;\mathbf{p}^*) - \mathbf{Q}(t_k;\mathbf{p}^*)\|_2^2}}{\sqrt{\sum_{k=1}^{N_T} \|\mathbf{Q}(t_k;\mathbf{p}^*)\|_2^2}}, \\
\mathcal{E}_{\text{GP}}(\mathbf{p}^*) &= \frac{\sqrt{\sum_{k=1}^{N_T} \|\mathbf{Q}_{\text{GP}}(t_k;\mathbf{p}^*) - \mathbf{Q}(t_k;\mathbf{p}^*)\|_2^2}}{\sqrt{\sum_{k=1}^{N_T} \|\mathbf{Q}(t_k;\mathbf{p}^*)\|_2^2}}, \\
\mathcal{E}_{\text{DRIPS}}(\mathbf{p}^*) &= \frac{\sqrt{\sum_{k=1}^{N_T} \|\mathbf{Q}_{\text{DRIPS}}(t_k;\mathbf{p}^*) - \mathbf{Q}(t_k;\mathbf{p}^*)\|_2^2}}{\sqrt{\sum_{k=1}^{N_T} \|\mathbf{Q}(t_k;\mathbf{p}^*)\|_2^2}}.
\end{aligned} \tag{4.2}$$

The codes of the numerical experiments are available on Github: `https://github.com/DDMS-ERE-Stanford/DRIPS`.

4.1. Advection-diffusion transport

Consider a state variable $s(\mathbf{x}, t)$, whose dynamics is governed by the two-dimensional advection-diffusion equation

$$\frac{\partial s}{\partial t} + u\frac{\partial s}{\partial x} - \kappa\nabla^2 s = 0, \qquad \mathbf{x} = (x, y)^\top \in \mathcal{D} = (0,1)\times(0,1), \quad t \in (0,1), \tag{4.3a}$$

with the initial and boundary conditions

$$\begin{aligned}
&s(\mathbf{x}, t=0) = 0, \\
&s(\mathbf{x}, t) = s_D(y, t), \quad \text{for } \mathbf{x} \in \Gamma_D = \{0\}\times[0,1], \\
&\nabla s(\mathbf{x}, t)\cdot\mathbf{n}(\mathbf{x}) = 0, \quad \text{for } \mathbf{x} \in \Gamma_N = \{\{1\}\times[0,1]\}\cup\{[0,1]\times\{1\}\}\cup\{[0,1]\times\{0\}\},
\end{aligned} \tag{4.3b}$$

where

$$s_D(y,t) = 300 + \begin{cases} 0 & \text{for} \quad y \in [0, \frac{1}{3}] \\ 325(\sin(3\pi|y-\bar{y}|)+1) & \text{for} \quad y \in [\frac{1}{3}, \frac{2}{3}] \\ 0 & \text{for} \quad y \in [\frac{2}{3}, 1], \end{cases} \tag{4.3c}$$

with $\mathbf{p} = \{u, \kappa, \bar{y}\} \in \Omega = [1000, 3000]\times[100, 300]\times[0.36, 0.42]$. The HFM represents a numerical solution of (4.3) obtained via finite differences, upwind scheme for the advection term and implicit center difference

scheme for the diffusion term, using 75 points in each spatial direction. The resulting discretized state variable $\mathbf{S}(t;p)$ forms a vector of dimension $N_{\text{sv}} = 75^2$ and is propagated in time with time step $\Delta t = 0.01$.

The QoIs in the first example are $\mathbf{Q}(t_n;p) = \mathbf{S}(t_n;p)$; they follow the linear HFM

$$\mathbf{Q}(t_{n+1};p) = \mathbf{A}(p)\mathbf{Q}(t_n;p) + \mathbf{b}(p), \tag{4.4}$$

where $\mathbf{A}$ is a tridiagonal matrix representing the advection and diffusion operators and $\mathbf{b}$ is a non-zero vector representing the inhomogeneous boundary conditions.

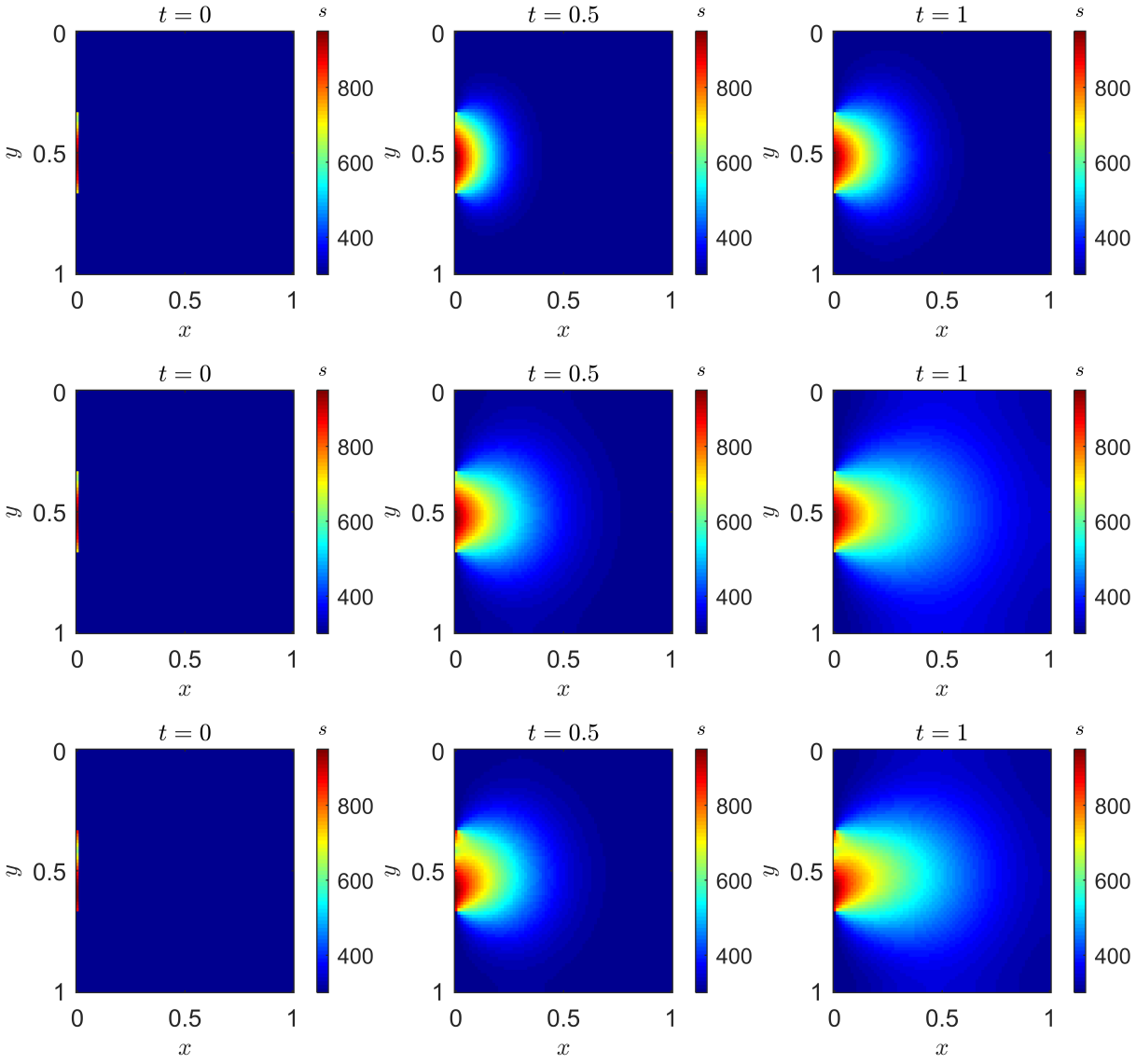

Figure 1: Temporal snapshots, at times $t = 0$, 0.5 and 1.0, of the training data generated via the HFM (4.4) for the parameter values $\mathbf{p}^{(1)} = \{1000, 100, 0.36\}$ (top row) and $\mathbf{p}^{(2)} = \{1000, 300, 0.36\}$ (middle row) and of the test data generated via the HFM (4.4) for the randomly selected parameter value $\mathbf{p}^* = \{2934.0597, 209.4464, 0.4184\}$ (bottom row).

The training data are generated from the HFM (4.4) evaluated on the Cartesian grid of the ($N_{\rm par} = 3$)-dimensional parameter space (a cube) with 3 points (two endpoints and one middle point) in each dimension, i.e., by solving (4.4) $N_{\rm MC} = 27$ times. Figure 1 exhibits temporal snapshots of two representative HFM solutions, $\mathbf{Q}(t_k; \mathbf{p}^{(1)})$ and $\mathbf{Q}(t_k; \mathbf{p}^{(2)})$, where $\mathbf{p}^{(1)} = \{1000, 100, 0.36\}$ and $\mathbf{p}^{(2)} = \{1000, 300, 0.36\}$. Our goal is to obtain a solution, $\mathbf{Q}(t_k; \mathbf{p}^*)$, for a randomly selected parameter value $\mathbf{p}^* = \{2934.0597, 209.4464, 0.4184\}$. $\in \Omega$, without computing (4.4). This solution is then compared with the reference solution $\mathbf{Q}(t; \mathbf{p}^*)$ of the HFM (4.4), which is also shown in Figure 1.

We compare the performance of DRIPS and POD-PROM, which differ in their computation of the ROM $\{\mathbf{A}_r(\mathbf{p}^{(i)}), \mathbf{b}_r(\mathbf{p}^{(i)})\}$ and $\mathbf{K}_r(\mathbf{p}^{(i)})$ in the offline step. The first relies on DMD, as described in Algorithm 6. The second replaces DMD with POD (Algorithm 7) to obtain $\{\mathbf{A}_r(\mathbf{p}^{(i)}), \mathbf{b}_r(\mathbf{p}^{(i)})\}$ from the Galerkin projection (A.1) (or alternative projections, e.g., Petrov-Galerkin, DEIM, etc.). This requires the full knowledge of $\mathbf{A}(\mathbf{p}^{(i)})$ and $\mathbf{b}(\mathbf{p}^{(i)})$, which is not needed in the DMD version that learns $\mathbf{K}_r(\mathbf{p}^{(i)})$ from the training dataset (2.6). Since the HFM (4.4) is linear, we set the observable function $\mathbf{g}$ as

$$\mathbf{y} = \mathbf{g}(\mathbf{Q}) = (1, \mathbf{Q}^\top)^\top, \tag{4.5}$$

so that (3.4) reduces to

$$\mathbf{y}(t_{n+1}; p) = \mathbf{K}(p)\mathbf{y}(t_n; p), \qquad \mathbf{K}(p) = \begin{bmatrix} 1 & 0 \\ \mathbf{b}(p) & \mathbf{A}(p) \end{bmatrix}. \tag{4.6}$$

Alternatively, one can use the extended DMD, xDMD [24], for the offline step.

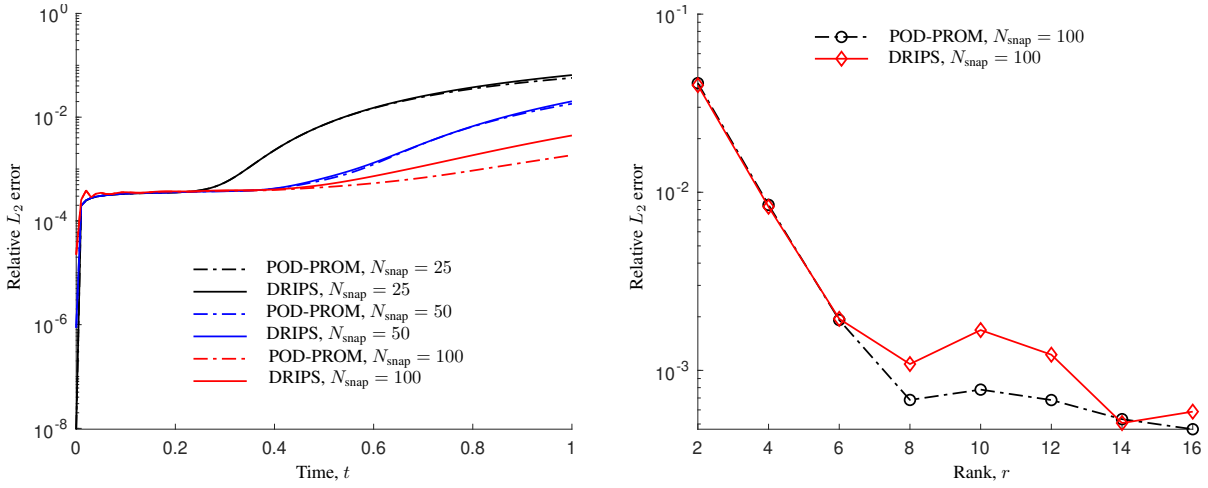

Figure 2: Left: temporal evolution of the relative L_2 error of POD-PROM and DRIPS, $\varepsilon_{\rm POD}$ and $\varepsilon_{\rm DRIPS}$ in (4.1), trained on different numbers of snapshots $N_{\rm snap}$. Right: dependence of the total relative L_2 error of POD-PROM and DRIPS, $\mathcal{E}_{\rm POD}$ and $\mathcal{E}_{\rm DRIPS}$ in (4.2), on the truncation rank r.

The performance of DRIPS and POD-PROM is quantified in terms of the relative L_2 errors defined in (4.1). They compare the predictions of the POD-PROM and DRIPS, $\mathbf{Q}_{\rm POD}(t; \mathbf{p}^*)$ and $\mathbf{Q}_{\rm DRIPS}(t; \mathbf{p}^*)$,

to the reference HFM prediction $\mathbf{Q}(t;\mathbf{p}^*)$ at the test parameter point $\mathbf{p}^*$. Figure 2 displays these errors for POD-PROM and DRIPS of rank $r = 10$, computed from different numbers of snapshots, N_{snap}. For both methods, the error decreases with N_{snap}. That is to be expected since the low-dimensional subspace, represented by the ROBs determined from the training data, becomes more optimal as the amount of informative data increases. When the data are relatively scarce ($N_{\text{snap}} = 25$ and 50), $\varepsilon_{\text{DRIPS}}(t;p^*) \geq \varepsilon_{\text{POD}}(t;p^*)$ because of the loss of accuracy in learning $\mathbf{K}_r(p^{(i)})$ from data in the DMD framework. Given enough data ($N_{\text{snap}} = 100$), the discrepancy between POD-PROM and DRIPS diminishes as the learned $\mathbf{K}_r(p^{(i)})$ becomes sufficiently accurate.

We use another metric, the total relative L_2 errors defined in (4.2) to elucidate the impact of the rank selection on the performance of POD-PROM and DRIPS algorithm. These errors are shown in Figure 2 for the same number of snapshots, $N_{\text{snap}} = N_T = 100$. Both $\mathcal{E}_{\text{POD}}$ and $\mathcal{E}_{\text{DRIPS}}$ decrease with the rank r, saturating after $r = 8$, which verifies the low-rank nature of this problem. DRIPS is slightly less accurate than POD-PROM for small r because of the loss in accuracy of learning $\mathbf{K}_r(p^{(i)})$. The accuracy of the DMD-based ROMs is affected by the number of snapshots and the rank truncation, as studied in, e.g., [22].

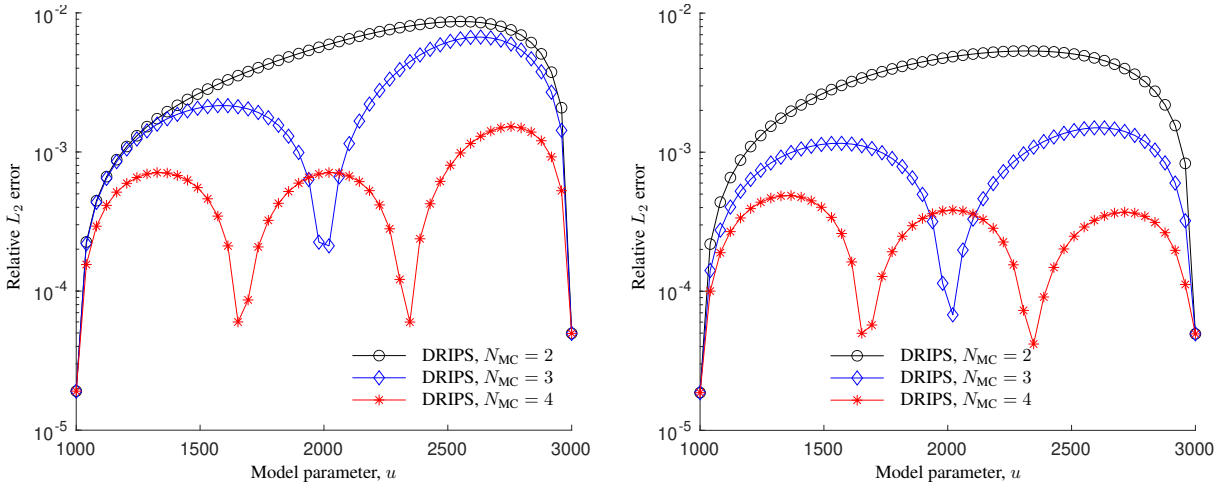

Figure 3: The total relative L_2 error of DRIPS (left) and POD-PROM (right) in the target parameter space using different number of training parameter points. $N_{\text{MC}} = 2$ is the case of using the two end points 1000 and 3000; $N_{\text{MC}} = 3$ is the case of using three uniform points 1000, 2000 and 3000; $N_{\text{MC}} = 4$ is the case of using four uniform end points 1000, 5000/3, 7000/3 and 3000.

The accuracy of the interpolated PROMs also depends on the location of the target points in the parameter space. To better visualize the distance in the parameter space, we fix $\kappa = 250$ and $\bar{y} = 0.4$, thus reducing the dimension of the parameter space to $N_{\text{par}} = 1$. Figure 3 shows the accuracy of DRIPS and POD-PROM at different target points in the one-dimensional parameter space, trained from $N_{\text{MC}} = 2, 3, 4$ training parameter points, respectively. The errors of both methods reach minimum at

the training parameter points and increase with the distance from these points. The prediction accuracy (i.e., accuracy at unsampled points in the whole parameter space) of the PROMs constructed with both methods increases with the number of training points, N_{MC}.

Next, we repeatedly draw $N_{\mathrm{test}} = 100$ random samples from the $N_{\mathrm{par}} = 3$ parameter space as test data and compare the computational cost and accuracy of DRIPS (the last column in Table 1), POD-PROM (the second column in Table 1) and GP (the third column in Table 1). POD-PROM and DRIPS provide a comparable speedup, but the POD-PROM serves as a more accurate surrogate than DRIPS. More than 95% of the computational time is spent on generating high-fidelity training data ($N_{\mathrm{MC}} = 27$ solved of the HFM) in the offline step of both methods. The offline step of DRIPS is more expensive than that of POD due to the construction of the $\mathbf{K}_r$ from data (rather than directly from Galerkin projection of $\mathbf{A}(\mathbf{p}^{(i)})$ and $\mathbf{b}(\mathbf{p}^{(i)})$ in POD). The offline training of GP is more expensive in this example because the interpolation is done pixel by pixel on the data of size $N_Q (N_Q \gg 1)$. Once the POD-ROM surrogate is trained, it provides more accurate predictions at the test points than either DRIPS or GP does. The online speedup of GP is superior to that of the other two methods. However, GP cannot be generalized to future time predictions.

The QoI in the second example is the time series $Q(t_n; \mathbf{p}) = s(\mathbf{x}_s, t_n; \mathbf{p})$, where $\mathbf{x}_s = (0.25, 0.5)^\top$ is the location of a sensor. To allow enough time for the flux to reach the sensor location, we are only interested in $t_n > 0.5$. The QoI follows the linear HFM

$$\begin{aligned} \mathbf{S}(t_{n+1}; \mathbf{p}) &= \mathbf{A}(p)\mathbf{S}(t_n; \mathbf{p}) + \mathbf{b}(\mathbf{p}), \\ Q(t_{n+1}; \mathbf{p}) &= \mathbf{C}\mathbf{S}(t_{n+1}; \mathbf{p}). \end{aligned} \tag{4.7}$$

Here $\mathbf{C}$ is a $1 \times N_{\mathrm{sv}}$ matrix with entries

$$\mathbf{C}_{1,i} = \begin{cases} 1 & i = i_s, \\ 0 & \text{otherwise}, \end{cases} \tag{4.8}$$

where i_s is an index in the discretized vector $\mathbf{S}$ corresponding to $\mathbf{x}_s$ in $\mathbf{s}(\mathbf{x})$. In this example, POD follows the same strategy as before, which computes the ROM $\{\mathbf{A}_r(\mathbf{p}^{(i)}), \mathbf{b}_r(\mathbf{p}^{(i)}), \mathbf{C}_r(\mathbf{p}^{(i)})\}$ from the Galerkin projection (A.1) with access to $\mathbf{S}(\mathbf{p}^{(i)})$ and required full knowledge of $\mathbf{A}(\mathbf{p}^{(i)})$, $\mathbf{b}(\mathbf{p}^{(i)})$ and $\mathbf{C}$. On the contrary, DMD constructs a surrogate for Q without any knowledge about the HFM (4.7). The observable function $\mathbf{g}$ is chosen as Hermite polynomials $\mathcal{H}$ of $Q(t^n; \mathbf{p})$ up to order $m = 4$,

$$\mathbf{y} = \mathbf{g}(Q) = [\mathcal{H}_0(Q), \ldots, \mathcal{H}_m(Q)]^\top. \tag{4.9}$$

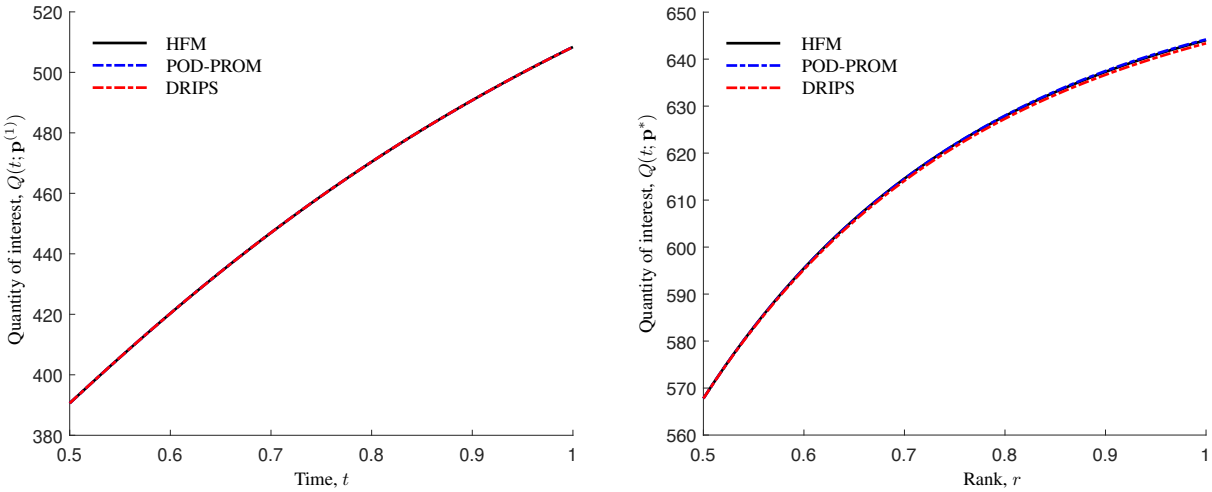

Figure 4: Left: the reference HFM, POD-PROM and DRIPS approximations of $Q(t;\mathbf{p}^{(1)}) = s(\mathbf{x}_s, t;\mathbf{p}^{(1)})$ for $t \in [0.5, 1]$ at the training data point $\mathbf{p}^{(1)} = \{1000, 100, 0.36\}$. Right: the reference HFM, POD-PROM and DRIPS approximations of $Q(t;\mathbf{p}^{*}) = s(\mathbf{x}_s, t;\mathbf{p}^{*})$ for $t \in [0.5, 1]$ at the test data point $\mathbf{p}^{*} = \{2934.0597, 209.4464, 0.4184\}$.

	HFM	$\mathbf{Q} = \mathbf{S}$			$Q = s(\mathbf{x}_s)$		
		POD	GP	DRIPS	POD	GP	DRIPS
Offline (sec)	149.10	150.57	156.18	155.04	152.99	151.03	151.22
Online (sec)	552.20	7.06	0.77	6.92	7.13	0.04	0.28
Online speedup	–	78.25	714.27	79.80	77.48	14569.97	1977.01
Total speedup	–	4.45	4.47	4.33	4.38	4.64	4.63
Average $\mathcal{E}$	–	$1.5 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$	$2.5 \cdot 10^{-3}$	$1.7 \cdot 10^{-4}$	$8.0 \cdot 10^{-4}$	$7.9 \cdot 10^{-4}$

Table 1: Simulation time and accuracy of the alternative calculations of the QoIs for test data consisting of $N_{\text{test}} = 100$ randomly drawn realization of (4.3).

Figure 4 shows that POD-PROM and DRIPS algorithm agree well with the reference HFM for both training data (left) and test data (right). Table 1 compares the computational costs and accuracy of DRIPS, POD-PROM and GP for a randomly drawn $N_{\text{test}} = 100$ test dataset. Since the dominant computational costs in online step stem from the entry-by-entry interpolation of ROBs (3.11), the online part of DRIPS is significantly faster than that of POD-PROM because the size of the ROBs in DRIPS, $(m+1) \cdot r$, is much smaller than that in POD-PROM, $N_{\text{sv}} \cdot r$ with $N_{\text{sv}} \gg m$. In this experiment, GP is more efficient than the other two methods because the interpolation is now done in the low-dimensional parameter space, $N_Q = 1$. However, without any knowledge of the underlying physics, the accuracy of GP cannot compete with the physics-aware POD-PROM.

These computational examples illustrate the benefits of the knowledge of the physics, i.e., of the specific values of $\mathbf{A}(\mathbf{p}^{(i)})$ and $\mathbf{b}(\mathbf{p}^{(i)})$, for the construction of accurate ROMs (as conventionally done in POD). It also demonstrates that one can construct adequate ROMs without full knowledge of the physics, as long as enough data (temporal snapshots) are available. By deploying DMD for data-driven learning, DRIPS compensates for the incomplete knowledge of the physics. The data-driven learning component of DRIPS can be improved by a proper selection of the the observable function $\mathbf{g}$ [22, 21, 23, 24, 20, 56]. The following numerical examples show how to incorporate physics ingredients into the DRIPS framework.

4.2. Heat conduction in a complex domain

Consider heat conduction in a two-dimensional multi-connected domain $\mathcal{D}$, an 800×800 square with an S-shaped cavity in the middle (Figure 5). The spatiotemporal distribution of temperature $s(\mathbf{x}, t)$ is governed by a diffusion equation with inhomogeneous boundary conditions,

$$\begin{cases} \rho c \dfrac{\partial s}{\partial t} - k\nabla^2 s = 0, \qquad \mathbf{x} = (x, y)^\top \in \mathcal{D}, \quad t \in (0, 5000] \\ s(\mathbf{x}, 0) = 0; \\ s(0, y, t) = s_L, \quad s(800, y, t) = s_R, \quad \dfrac{\partial s}{\partial y}(x, 0, t) = \dfrac{\partial s}{\partial y}(x, 800, t) = 0, \quad s(\mathbf{x}, t) = 3 \quad \text{on } \partial\mathcal{S}, \end{cases} \tag{4.10}$$

where ∂D_L(green) and ∂D_R (blue) are the Dirichlet boundary segments representing the left and right sides of the computational domain, respectively; and ∂S is the bounding surface of the cavity in Figure 5. The material properties are set to $\rho = 1$, $k = 1$ and $\mathbf{p} = \{c, s_L, s_R\} \in [1, 2] \times [1, 3] \times [1, 3]$; all defined in some consistent units. The training data and reference solutions are obtained via the Matlab PDE toolbox on the finite-element mesh with 4287 elements (Figure 5).

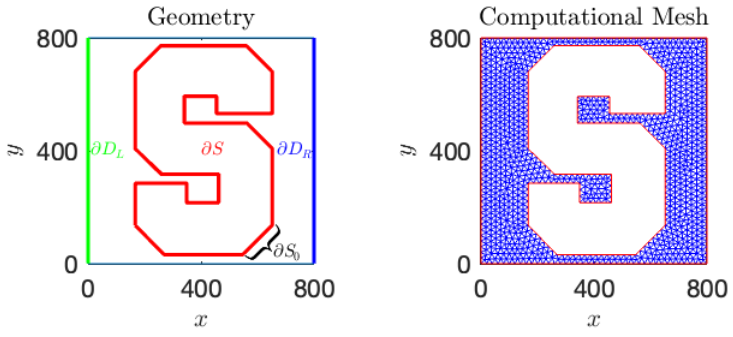

Figure 5: Multi-connected simulation domain $\mathcal{D}$ (left) and the mesh used in the finite-element solution of (4.10) (right).

In the first set of experiments, our QoIs are the spatiotemporal distribution of temperature, $\mathbf{Q}(t^n; \mathbf{p}) = \mathbf{S}(t^n; \mathbf{p})$, and the observable function $\mathbf{g}$ is given by (4.5). The training data are the repeated solutions of (4.10) with the three parameters $\mathbf{p}$ sampled on the Cartesian grid of the cube $[1, 2] \times [1, 3] \times [1, 3]$. Each of these parameters is sampled at two endpoints and one middle point, which yields the training

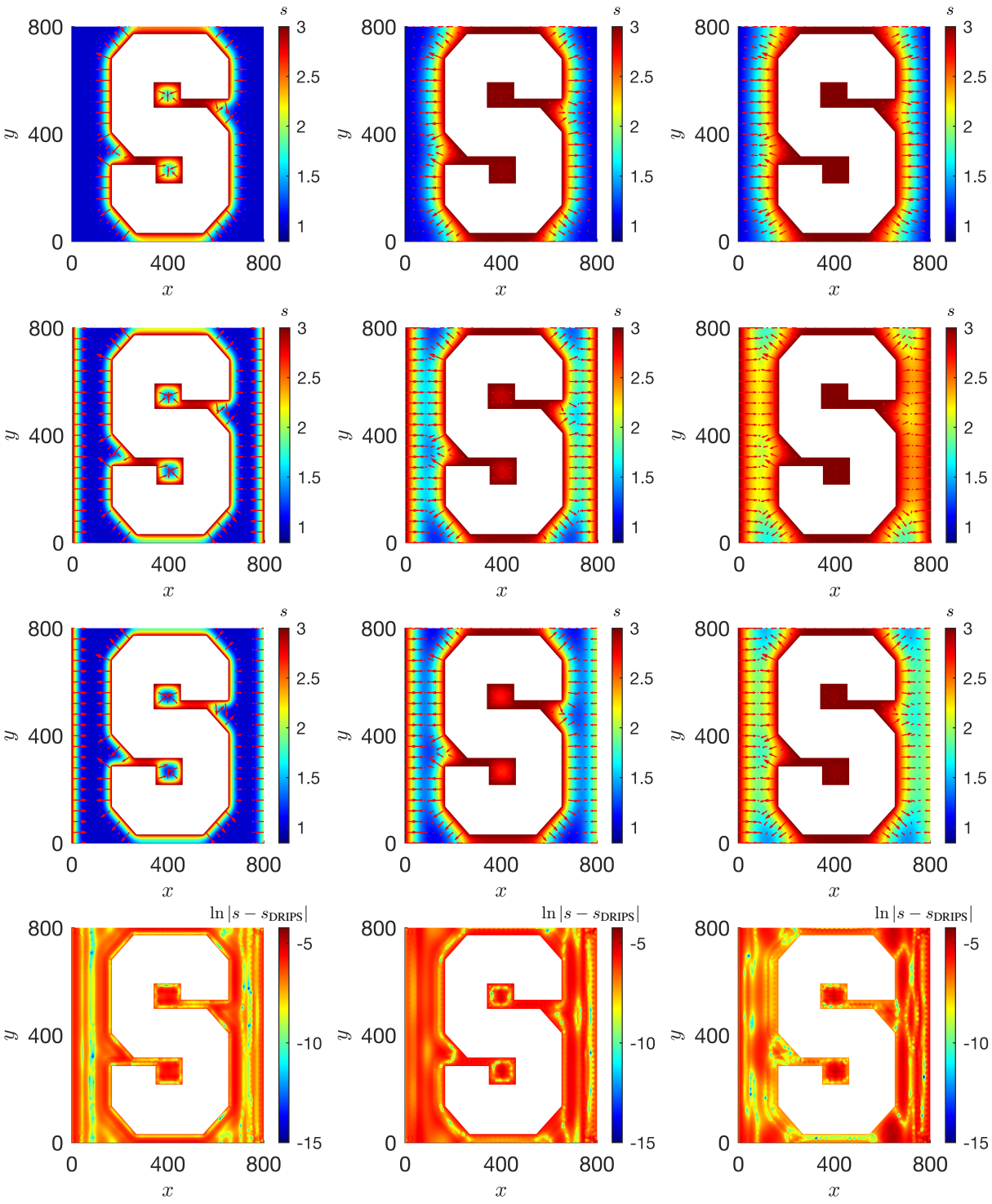

Figure 6: Temporal snapshots, at times $t = 500$ (left column), $t = 2500$ (middle column) and $t = 5000$ (right column), of the training data generated by solving (4.10) with parameter values $\mathbf{p}^{(1)} = \{1, 1, 1\}$ (first row) and $\mathbf{p}^{(2)} = \{2, 3, 3\}$ (second row). Also shown is the reference solution of (4.10) with parameter value $\mathbf{p}^* = \{1.9670, 2.0945, 2.9454\}$ (third row). The log absolute errors of the DRIPS estimator of this reference solution are shown in the last row.

data set consisting of $N_{\mathrm{MC}} = 27$ realizations of the HFM. The reference solution of (4.10) corresponds to
285 a randomly selected parameter value $\mathbf{p}^* = \{1.9670, 2.0945, 2.9454\}$. Representative temporal snapshots of these data are plotted in Figure 6 together with the log absolute error map of the DRIPS estimate of $\mathbf{Q}(t^n; p^*)$. The HFM has the form of (4.4), as implemented in the Matlab PDE toolbox. Even in the absence of an explicit HFM formulation, DRIPS provides an accurate ROM with the (small) rank of $r = 10$.

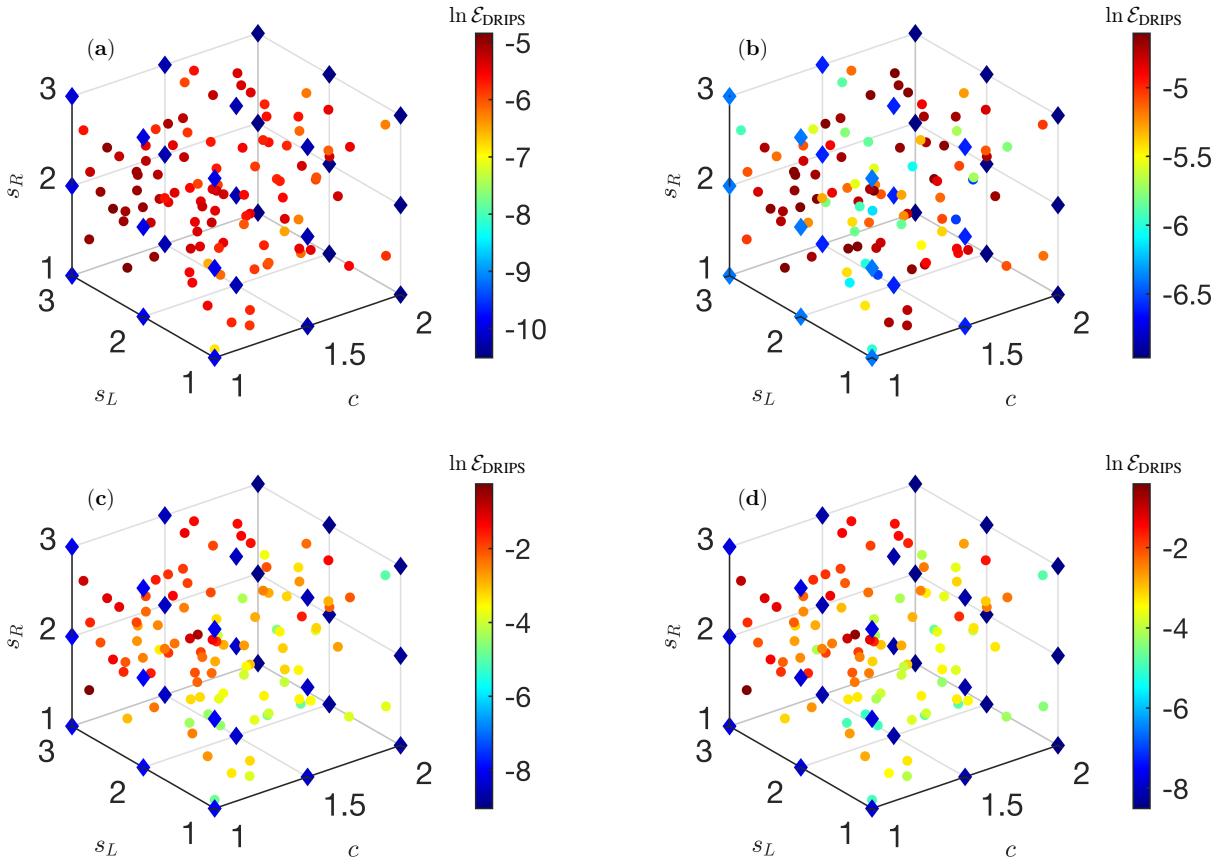

Figure 7: The log total relative error, $\ln(\mathcal{E}_{\mathrm{DRIPS}}(\mathbf{p}^*))$, of the DRIPS estimators of **(a)** the spatial distribution of temperature, $\mathbf{Q} = \mathbf{S}$; **(b)** the heat rate across the interface ∂S_0, $Q = -\int_{\partial S_0} k \nabla s \cdot \mathbf{n} \mathrm{d}A$; and **(c)** and **(d)** the spatial distributions of the two components of heat flux, $\mathbf{Q}_1 = -k\partial_x \mathbf{S}$ and $\mathbf{Q}_2 = -k\partial_y \mathbf{S}$. These estimates utilize DMD of rank $r = 10$, 8, and 20, respectively. The diamonds represent training data and the circles represent 100 test data, which are randomly drawn from the parameter space on the Cartesian grid.

290 If the POD-PROM were to be used instead, one would have to know the exact values of $\mathbf{A}(\mathbf{p}^{(i)})$ and $\mathbf{b}(\mathbf{p}^{(i)})$ at the Cartesian grid of the training data set, which requires large numerical differential matrices and embeddings of the complex boundary conditions. Figure 7a shows the accuracy of the $r = 10$ DRIPS trained on the Cartesian grid (shown in diamonds) for different values of $\mathbf{p}^* \in \Omega$ (shown in circles). The total relative error $\mathcal{E}_{\mathrm{DRIPS}}(\mathbf{p}^*)$, averaged over 100 randomly selected values of $\mathbf{p}^*$, is 0.0041. The error

$\mathcal{E}_{\text{DRIPS}}(\mathbf{p}^*)$ increases with the distance from the training points. It reaches its minimum at some training data points, where the error reflects the discrepancy between the HFMs and the ROMs constructed via local ROBs for $\mathbf{p}^* = \mathbf{p}^{(i)}$.

The second set of experiments treats the spatial distribution of the heat flux as QoIs, i.e., $\mathbf{Q}_1(t_n;\mathbf{p}) = -k\partial_x\mathbf{S}(t_n;\mathbf{p})$ and $\mathbf{Q}_2(t_n;\mathbf{p}) = -k\partial_y\mathbf{S}(t_n;\mathbf{p})$. The heat flux, $-k\nabla s$, for $\mathbf{p}^{(1)}$, $\mathbf{p}^{(2)}$, and $\mathbf{p}^*$, is plotted in Figure 6 using arrows. The observable function $\mathbf{g}$ is chosen as

$$\mathbf{y} = \mathbf{g}(\mathbf{Q}_1, \mathbf{Q}_2) = (1, \mathbf{Q}_1^\top, \mathbf{Q}_2^\top)^\top. \tag{4.11}$$

Figures 7c and 7d exhibit the total relative error $\mathcal{E}_{\text{DRIPS}}(\mathbf{p}^*)$ of the DRIPS estimators (with rank $r = 20$) of $\mathbf{Q}_1$ and $\mathbf{Q}_2$ for different $\mathbf{p}^* \in \Omega$. The total relative error $\mathcal{E}_{\text{DRIPS}}(\mathbf{p}^*)$, averaged over 100 randomly drawn values of $\mathbf{p}^*$, are 0.1126 and 0.0994 for $\mathbf{Q}_1$ and $\mathbf{Q}_2$, respectively. The error's behavior is identical to the previous case, but its magnitude is about an order of magnitude higher.

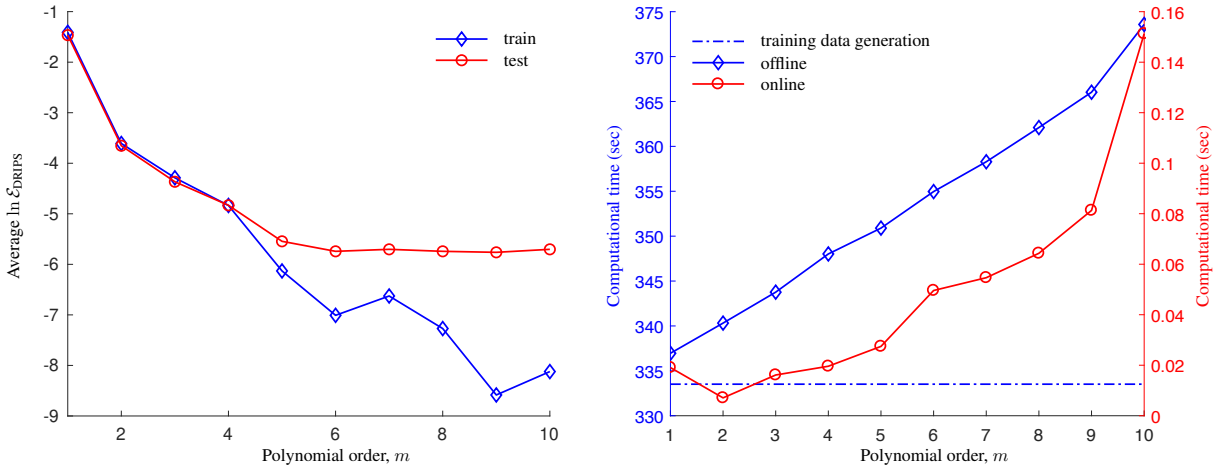

Figure 8: (Left) Log average total relative error of the training and testing datasets and (Right) Computational costs, both plotted as function of the Hermite polynomial order m.

In the final set of experiments, our QoI is the heat loss through the segment ∂S_0 of the cavity's surface, $Q(t^n;\mathbf{p}) = -\int_{\partial S_0} k\nabla s \cdot \mathbf{n} dA$, where $\mathbf{n}$ is the unit normal vector to ∂S_0 (Figure 5). The Hermite polynomials $\mathcal{H}[Q(t^n;\mathbf{p})]$ of order up to $m = 8$, defined in (4.9), serve as the observable function $\mathbf{g}$. The total relative error $\mathcal{E}_{\text{DRIPS}}(\mathbf{p}^*)$ of the DRIPS estimators of Q (with rank $r = 8$) is shown in Figure 7b, for different values of $\mathbf{p}^* \in \Omega$; its behavior is identical to that observed in the previous experiment, with the average (over the test data for 100 randomly drawn realizations of $\mathbf{p}^*$) error of 0.0067. Figure 8 demonstrates the importance of choosing the observable $\mathbf{g}$ by exhibiting the accuracy and computational cost of DRIPS as functions of the polynomial order m. The left graph shows the log total relative error, averaged over the 27 training data points and 100 randomly drawn test data points, respectively. Incorporating higher-order polynomials into the observable results in higher-order of accuracy in both

training and test datasets. The latter saturates after $m = 5$ as the interpolation error becomes dominant. The right graph shows the computational costs of the offline step and online step, for different m. The costs of both the offline and online steps increase with the polynomial order in the observable. However, the cost of the online step is a negligible fraction of the cost of the offline step, wherein the generation of the training data dominates the computational time.

Table 2 contains the computational cost and accuracy of DRIPS and GP, averaged over a randomly drawn $N_{\text{test}} = 100$ test points. For the three experiments reported above, DRIPS is more than four times faster than the HFM computation, while maintaining satisfactory accuracy. As discussed earlier, the generation of high-fidelity training data during the offline step dominates the total computational cost of DRIPS. The online speedup becomes more significant when N, the size of the surrogate model for a given QoI, is much smaller than N_{sv}, the size of the state variable, due to the reduced size of the expensive entry-by-entry interpolation of ROBs. As before, GP provides accurate predictions but is computationally more expensive in the offline training process of interpolating high-dimensional data, $N_Q \gg 1$. GP differs from DRIPS (and POD-PROM) in the sense that it only interpolates in the parameter space without any dimension reduction in the state/QoI space.

	HFM	$\mathbf{Q} = \mathbf{S}$		$\mathbf{Q} = -k\nabla\mathbf{s}$		$Q = -\int_{\partial S_0} k\nabla s \cdot \mathbf{n} dA$	
		GP	DRIPS	GP	DRIPS	GP	DRIPS
Offline (sec)	333.51	340.84	337.16	348.24	340.23	334.75	364.55
Online (sec)	1235.24	12.47	23.76	17.48	4.50	0.07	0.16
Online speedup	–	99.03	51.98	70.68	275.05	18820.65	7700.99
Total speedup	–	4.44	4.35	4.29	4.55	4.69	4.30
Average $\mathcal{E}$	–	0.0009	0.0004	$(0.28, 0.11)^\top$	$(0.11, 0.10)^\top$	0.0005	0.0007

Table 2: The computational cost and accuracy of DRIPS estimates of several QoIs in the HFM (4.10), averaged over 100 randomly drawn test data.

4.3. Laminar flow past a cylinder

Consider two-dimensional flow of an incompressible fluid with density ρ and dynamic viscosity ν (these and other quantities are reported in consistent units) around an impermeable circle of diameter $D = 0.1$. The flow, which takes place inside a rectangular domain $\mathcal{D} = \{\mathbf{x} = (x, y)^\top : (x, y) \in [0, 2] \times [0, 1]\}$, is driven by an externally imposed pressure gradient; the center of the circular inclusion is $\mathbf{x}_{\text{circ}} = (0.3, 0.5)^\top$. Dynamics of the three state variables, flow velocity $\mathbf{s}(\mathbf{x}, t) = (s_1, s_2)^\top$ and fluid pressure $P(\mathbf{x}, t)$, is

described by the two-dimensional Navier-Stokes equations,

$$\begin{cases} \dfrac{\partial s_1}{\partial x} + \dfrac{\partial s_2}{\partial y} = 0; \\ \dfrac{\partial s_1}{\partial t} + s_1 \dfrac{\partial s_1}{\partial x} + s_2 \dfrac{\partial s_1}{\partial y} = -\dfrac{1}{\rho}\dfrac{\partial P}{\partial x} + \nu \left(\dfrac{\partial^2 s_1}{\partial x^2} + \dfrac{\partial^2 s_1}{\partial y^2} \right), \qquad \mathbf{x} \in \mathcal{D}, \quad t > 0; \\ \dfrac{\partial s_2}{\partial t} + s_1 \dfrac{\partial s_2}{\partial x} + s_2 \dfrac{\partial s_2}{\partial y} = -\dfrac{1}{\rho}\dfrac{\partial P}{\partial y} + \nu \left(\dfrac{\partial^2 s_2}{\partial x^2} + \dfrac{\partial^2 s_2}{\partial y^2} \right); \end{cases} \tag{4.12a}$$

subject to initial conditions $\mathbf{s}(x, y, 0) = (0, 0)^\top$ and $P(x, y, 0) = 0$; and boundary conditions

$$P(2, y, t) = 0, \quad \partial_{\mathbf{n}} P_{|_{\partial\mathcal{D}\setminus\{x=2\}}} = 0, \quad \mathbf{s}(0, y, t) = (s_1^0, 0)^\top, \ \partial_{\mathbf{n}}\mathbf{s}(2, y, t) = \mathbf{0}, \quad \mathbf{s}(x, 0, t) = \mathbf{s}(x, 1, t) = \mathbf{0}. \tag{4.12b}$$

Here $\partial_{\mathbf{n}} = \mathbf{n} \cdot \nabla$, with $\mathbf{n}$ denoting the unit normal vector to the bounding surface. The model's predictions are affected by the uncertain parameter set $\mathbf{p} = \{s_1^0, \rho, \mu\} \in [0.8, 1] \times [0.9, 1.1] \times [1/700, 1/500]$.

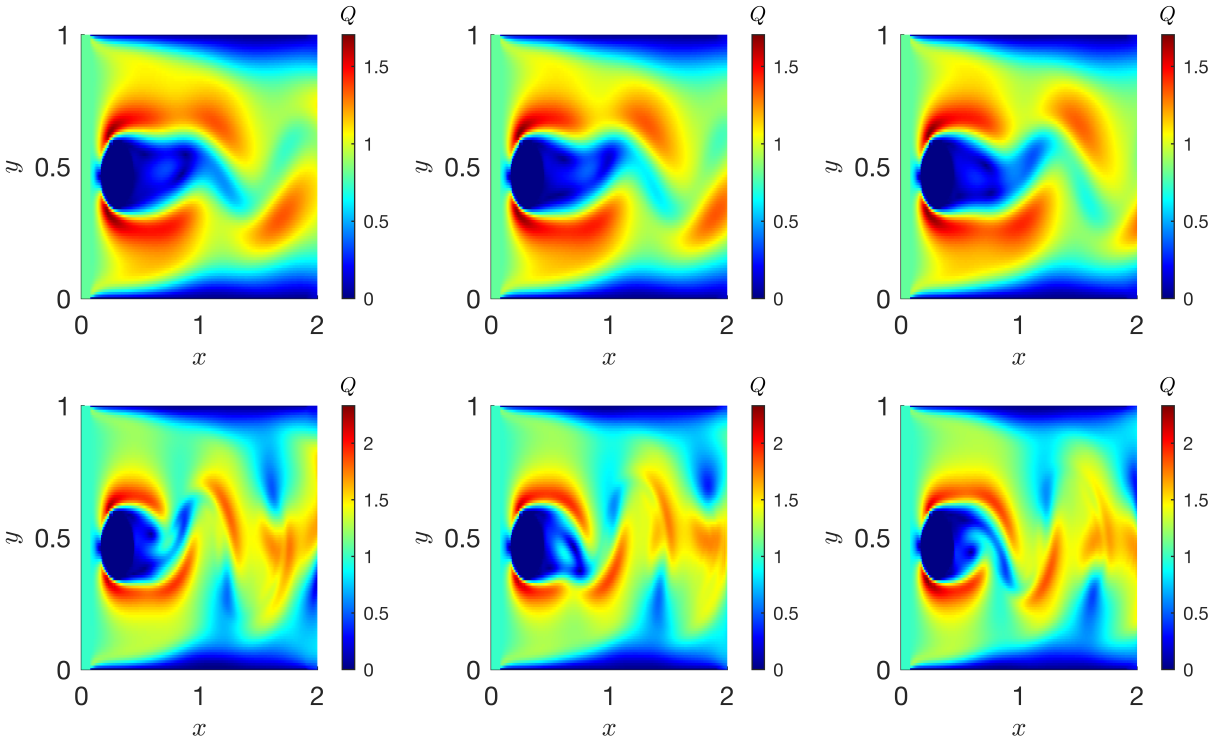

Figure 9: Training data consisting of temporal snapshots, at times $t = 4.125$ (left), $t = 4.3125$ (middle) and $t = 4.5$ (right), of the QoI $\mathbf{Q}$ computed with (4.12) for $\mathbf{p}^{(1)} = \{0.8, 0.9, 1/500\}$ (top) and $\mathbf{p}^{(2)} = \{1, 1.1, 1/700\}$ (bottom).

330 Our QoI is the magnitude of the flow velocity, $Q(\mathbf{x}, t; \mathbf{p}) = \sqrt{s_1^2 + s_2^2}$, or its discretized and vectorized form $\mathbf{Q}(t^n; \mathbf{p}) = \sqrt{\mathbf{S}_1(t^n; \mathbf{p})^2 + \mathbf{S}_2(t^n; \mathbf{p})^2}$. The training data (Figure 9) and reference solution (Figure 10) are obtained with the Matlab code [57], which implements a finite-difference scheme on the staggered grid with $\Delta x = \Delta y = 0.02$ and $\Delta t = 0.0015$. In training, we collect $N_{\text{snap}} = 250$ snapshots of $\mathbf{Q}$ from $t = 4.125$ to $t = 4.5$ defined on the Cartesian grid spanning the three-dimensional cube in the parameter space. Each

dimension includes two endpoints and one middle point, giving a training data set comprising $N_{\mathrm{MC}} = 27$ HFM outputs. The training data exhibit various dynamic patterns (two examples are shown in Figure 9), from which DMD learns the nonlinear dynamics from a 10-rank surrogate using the same observable as in (4.5). This construction is equivalent to the xDMD; the details of this numerical example can be found in section 4.3 of [24]. The reference solution of a test parameter point $\mathbf{p}^* = \{0.9934, 1.0094, 0.0014\}$, together with the absolute error map of DRIPS, are presented in Figure 10.

Figure 11 shows the accuracy of the $r = 10$ DRIPS and Kriging surrogates trained on the Cartesian grid (diamonds) for different values of $\mathbf{p}^* \in \Omega$ (circles). The total relative errors $\mathcal{E}_{\mathrm{DRIPS}}(\mathbf{p}^*)$ and $\mathcal{E}_{\mathrm{GP}}(\mathbf{p}^*)$, both averaged over 100 randomly drawn test data, are reported in Table 3, together with the computational costs of the two methods. As in the previous examples, the majority of the computational costs stems from generating the training data. The online step of DRIPS is slightly more expensive than that of Kriging due to the entry-by-entry interpolation of ROBs, i.e., (3.11), which scales with N. Although Kriging provides a larger online speedup, DRIPS offers the same total speedup as Kriging and has much better accuracy in all test parameter points (Figure 11 and Table 3).

	HFM	Kriging	DRIPS
Offline (sec)	1618.45	1643.77	1634.17
Online (sec)	5994.26	15.67	54.49
Online speedup	–	382.52	110.01
Total speedup	–	4.59	4.51
Average total relative error	–	0.14	0.04

Table 3: Computational cost and accuracy of Kriging and DRIPS, averaged over 100 randomly drawn test data.

5. Conclusion

We proposed a new physics-aware data-driven framework, DRIPS, to construct reduced-order models of parametric complex systems. DRIPS combines the advantages of the data-driven modeling tool DMD and ROBs used for manifold interpolations in the parameter space. Attractive features of DRIPS include

- the ability to handle QoIs directly, without having to access the HFMs for the underlying high-dimensional state variables, as required in projection-based PROM methods;
- the improved accuracy relative to that of the conventional pure-data driven techniques such as Kriging (Gaussian processes); and
- the low training data requirements relative to data-intensive nonlinear machine-learning methods such as deep neural networks, which require $\mathcal{O}(10^6)$ training data points.

Our numerical experiments suggest that DRIPS is a robust and flexible framework, which bridges the understanding of data and physics. The general framework can be applied to many applications and be extended to higher parameter dimensions with adaptive sampling strategy. Compared to expensive high-fidelity simulations, the significant speedup provided by DRIPS makes it is suitable for real-time processing.

In the follow-up studies we will develop the current framework in several directions. First, to further improve the online speedup, we will explore more efficient manifold interpolation methods, e.g., [36]. Second, for problems with high-dimensional parameter spaces, we will design adaptive sampling strategies and combine DRIPS with parameter space reduction techniques [58]. Third, further studies on error estimation DRIPS are needed in order to construct reliable surrogate models, which may further enable outer-loop applications such as design, inverse problems, optimization and uncertainty quantification.

6. Acknowledgments

The research was partially supported in part by the Air Force Office of Scientific Research under grant FA9550-21-1-0381, by the National Science Foundation under award 2100927, by the Office of Advanced Scientific Computing Research (ASCR) within the Department of Energy Office of Science under award number DE-SC0023163, and by the Strategic Environmental Research and Development Program (SERDP) of the Department of Defense under award RC22-3278.

Appendix A. Implementation of the POD-PROM Framework [33]

Algorithm 7: POD-PROM framework for linear system (4.4) and (4.7)

Offline Step:

For $i = 1, \ldots, N_{\mathrm{MC}}$,

1. Compute the HFM (4.4)/(4.7), collect $\{\mathbf{S}(t_0; \mathbf{p}^{(i)}), \ldots, \mathbf{S}(t_{N_{\mathrm{snap}}}; \mathbf{p}^{(i)})\}$, $\mathbf{A}(\mathbf{p}^{(i)})$, $\mathbf{b}(\mathbf{p}^{(i)})$ and $\mathbf{C}$,
2. Apply SVD $[\mathbf{S}(t_0; \mathbf{p}^{(i)})^\top, \ldots, \mathbf{S}(t_{N_{\mathrm{snap}}}; \mathbf{p}^{(i)})^\top]^\top \approx \mathbf{V}(\mathbf{p}^{(i)})\mathbf{\Sigma}(\mathbf{p}^{(i)})\mathbf{Z}(\mathbf{p}^{(i)})^\star$ with $\mathbf{V}(\mathbf{p}^{(i)}) \in \mathbb{C}^{N \times r}$, $\mathbf{\Sigma}(\mathbf{p}^{(i)}) \in \mathbb{C}^{r \times r}$, $\mathbf{Z}(\mathbf{p}^{(i)})^\star \in \mathbb{C}^{r \times N_{\mathrm{snap}}}$, where r is the truncated rank chosen by certain criteria and should be the same for all $i = 1, \ldots, N_{\mathrm{MC}}$.
3. Use Galerkin-projection to compute local ROMs:

$$\begin{aligned} &\mathbf{A}_r(\mathbf{p}^{(i)}) = \mathbf{V}(\mathbf{p}^{(i)})^\star \mathbf{A}(\mathbf{p}^{(i)})\mathbf{V}(\mathbf{p}^{(i)}), \mathbf{b}_r(\mathbf{p}^{(i)}) = \mathbf{V}(\mathbf{p}^{(i)})^\star \mathbf{b}(\mathbf{p}^{(i)}), \\ &\mathbf{C}_r(\mathbf{p}^{(i)}) = \mathbf{C}\mathbf{V}(\mathbf{p}^{(i)}), \end{aligned} \tag{A.1}$$

4. Compute $\mathbf{P}^{(i,j)} = \mathbf{V}(\mathbf{p}^{(i)})^\star \mathbf{V}(\mathbf{p}^{(j)})$ for $i, j = 1, \ldots, N_{\mathrm{MC}}$.

Output: $\mathbf{V}(\mathbf{p}^{(i)})$, $\mathbf{P}^{(i,j)}$, $\mathbf{A}_r(\mathbf{p}^{(i)})$, $\mathbf{b}_r(\mathbf{p}^{(i)})$ and $\mathbf{C}_r(\mathbf{p}^{(i)})$.

End

Online Step:

- Interpolation of ROBs:

  Input: $\{\mathbf{V}(\mathbf{p}^{(i)})\}_{i=1}^{N_{\mathrm{MC}}}$, $\{\mathbf{P}^{(i,j)}\}_{i,j=1}^{N_{\mathrm{MC}}}$, $\{\mathbf{p}^{(i)}\}_{i=1}^{N_{\mathrm{MC}}}$, $\mathbf{p}^*$ $\xrightarrow{\text{Algorithm 2}}$ Output: $\mathbf{V}(\mathbf{p}^*)$

- Interpolation of PROMs:

  Input: $\{\mathbf{A}_r(\mathbf{p}^{(i)}), \mathbf{b}_r(\mathbf{p}^{(i)}), \mathbf{C}_r(\mathbf{p}^{(i)}), \mathbf{P}^{(i,j)}\}_{i,j=1}^{N_{\mathrm{MC}}}$, reference choice i_0,

  – Compute $\mathbf{P}^{(i,i_0)} = \mathbf{U}_{i,i_0}\mathbf{\Sigma}_{i,i_0}\mathbf{Z}_{i,i_0}^\star$ (SVD), $\mathbf{Q}_i = \mathbf{U}_{i,i_0}\mathbf{Z}_{i,i_0}^\star$,

  – Transform $\tilde{\mathbf{A}}_r(\mathbf{p}^{(i)}) = \mathbf{Q}_i^\star \mathbf{A}_r(\mathbf{p}^{(i)})\mathbf{Q}_i$, $\tilde{\mathbf{b}}_r(\mathbf{p}^{(i)}) = \mathbf{Q}_i^\star \mathbf{b}_r(\mathbf{p}^{(i)})$ and $\tilde{\mathbf{C}}_r(\mathbf{p}^{(i)}) = \mathbf{C}_r(\mathbf{p}^{(i)})\mathbf{Q}_i$.

  – Input: $\{\tilde{\mathbf{A}}_r(\mathbf{p}^{(i)}), \tilde{\mathbf{b}}_r(\mathbf{p}^{(i)}), \tilde{\mathbf{C}}_r(\mathbf{p}^{(i)})\}_{i,j=1}^{N_{\mathrm{MC}}}, i_0$ $\xrightarrow{\text{Algorithm 4}}$ Output: $\{\mathbf{A}_r(\mathbf{p}^*), \mathbf{b}_r(\mathbf{p}^*), \mathbf{C}_r(\mathbf{p}^*)\}$.

- POD reconstruction:

  Input: $\mathbf{A}_r(\mathbf{p}^*)$, $\mathbf{b}_r(\mathbf{p}^*)$, $\mathbf{V}(\mathbf{p}^*)$ and $\mathbf{S}(t_0; \mathbf{p}^*)$,

  $\mathbf{q}(t_0; \mathbf{p}^*) = \mathbf{V}(\mathbf{p}^*)^\star \mathbf{S}(t_0; \mathbf{p}^*)$

  For $k = 1, \ldots, N_T$,

  $\mathbf{q}(t_k; \mathbf{p}^*) = \mathbf{A}_r(\mathbf{p}^*)\mathbf{q}(t_{k-1}; \mathbf{p}^*) + \mathbf{b}_r(\mathbf{p}^*)$, $\mathbf{Q}_{\mathrm{POD}}(t_k; \mathbf{p}^*) = \mathbf{C}_r(\mathbf{p}^*)\mathbf{q}(t_k; \mathbf{p}^*)$,

  End

  Output: $\mathbf{Q}_{\mathrm{POD}}(t_k; \mathbf{p}^*)$.

References

[1] J. L. Lumley, The structure of inhomogeneous turbulent flows, in: A. M. Yaglom, V. I. Tartarsky (Eds.), Atmospheric turbulence and radio wave propagation, Nauka, Moscow, USSR, 1967, pp. 166–177.

[2] G. Kerschen, J.-C. Golinval, A. F. Vakakis, L. A. Bergman, The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview, Nonlinear Dynamics 41 (1) (2005) 147–169.

[3] B. Haasdonk, Reduced basis methods for parametrized PDEs—a tutorial introduction for stationary and instationary problems, Model reduction and approximation: theory and algorithms 15 (2017) 65.

[4] S. Chaturantabut, D. C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM Journal on Scientific Computing 32 (5) (2010) 2737–2764.

[5] C. W. Rowley, Model reduction for fluids, using balanced proper orthogonal decomposition, International Journal of Bifurcation and Chaos 15 (03) (2005) 997–1013.

[6] B. Moore, Principal component analysis in linear systems: Controllability, observability, and model reduction, IEEE Transactions on Automatic Control 26 (1) (1981) 17–32.

[7] S. Gugercin, A. C. Antoulas, C. Beattie, H_2 model reduction for large-scale linear dynamical systems, SIAM Journal on Matrix Analysis and Applications 30 (2) (2008) 609–638.

[8] A. C. Antoulas, C. A. Beattie, S. Güğercin, Interpolatory methods for model reduction, SIAM, 2020.

[9] K. Lee, K. T. Carlberg, Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, Journal of Computational Physics 404 (2020) 108973.

[10] C. E. Rasmussen, Gaussian processes in machine learning, in: Summer school on machine learning, Springer, 2003, pp. 63–71.

[11] G. S. H. Pau, Y. Zhang, S. Finsterle, Reduced order models for many-query subsurface flow applications, Computational Geosciences 17 (4) (2013) 705–721.

[12] D. J. Booker, R. A. Woods, Comparing and combining physically-based and empirically-based approaches for estimating the hydrology of ungauged catchments, Journal of Hydrology 508 (2014) 227–239.

[13] S. A. Naghibi, H. R. Pourghasemi, B. Dixon, GIS-based groundwater potential mapping using boosted regression tree, classification and regression tree, and random forest machine learning models in Iran, Environmental Monitoring and Assessment 188 (1) (2016) 1–27.

[14] P. J. Schmid, Dynamic mode decomposition of numerical and experimental data, Journal of Fluid Mechanics 656 (2010) 5–28.

[15] J. N. Kutz, S. L. Brunton, B. W. Brunton, J. L. Proctor, Dynamic mode decomposition: data-driven modeling of complex systems, SIAM, 2016.

[16] B. Peherstorfer, K. Willcox, Data-driven operator inference for nonintrusive projection-based model reduction, Computer Methods in Applied Mechanics and Engineering 306 (2016) 196–215.

[17] S. A. McQuarrie, P. Khodabakhshi, K. E. Willcox, Non-intrusive reduced-order models for parametric partial differential equations via data-driven operator inference, SIAM J. Sci. Comput. 45 (4) (2023) A1917–A1946. doi:10.1137/21M1452810.

[18] J. S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, Journal of Computational Physics 363 (2018) 55–78.

[19] T. Qin, K. Wu, D. Xiu, Data driven governing equations approximation using deep neural networks, Journal of Computational Physics 395 (2019) 620–635.

[20] J. N. Kutz, J. L. Proctor, S. L. Brunton, Applied Koopman theory for partial differential equations and data-driven modeling of spatio-temporal systems, Complexity 2018 (2018).

[21] H. Lu, D. M. Tartakovsky, Lagrangian dynamic mode decomposition for construction of reduced-order models of advection-dominated phenomena, J. Comput. Phys. (2020) 109229.

[22] H. Lu, D. M. Tartakovsky, Prediction accuracy of dynamic mode decomposition, SIAM Journal on Scientific Computing 42 (3) (2020) A1639–A1662.

[23] H. Lu, D. M. Tartakovsky, Dynamic mode decomposition for construction of reduced-order models of hyperbolic problems with shocks, Journal of Machine Learning for Modeling and Computing 2 (1) (2021).

[24] H. Lu, D. M. Tartakovsky, Extended dynamic mode decomposition for inhomogeneous problems, Journal of Computational Physics 444 (2021) 110550.

[25] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nature Reviews Physics 3 (6) (2021) 422–440.

[26] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, Journal of Computational Physics 394 (2019) 56–81.

[27] E. Qian, B. Kramer, B. Peherstorfer, K. Willcox, Lift & Learn: Physics-informed machine learning for large-scale nonlinear dynamical systems, Physica D: Nonlinear Phenomena 406 (2020) 132401.

[28] B. I. Epureanu, A parametric analysis of reduced order models of viscous flows in turbomachinery, Journal of Fluids and Structures 17 (7) (2003) 971–982.

[29] C. Homescu, L. R. Petzold, R. Serban, Error estimation for reduced-order models of dynamical systems, SIAM Journal on Numerical Analysis 43 (4) (2005) 1693–1714.

[30] R. Serban, C. Homescu, L. R. Petzold, The effect of problem perturbations on nonlinear dynamical systems and their reduced-order models, SIAM Journal on Scientific Computing 29 (6) (2007) 2621–2643.

[31] T. Lieu, C. Farhat, Adaptation of aeroelastic reduced-order models and application to an F-16 configuration, AIAA Journal 45 (6) (2007) 1244–1257.

[32] D. Amsallem, C. Farhat, Interpolation method for adapting reduced-order models and application to aeroelasticity, AIAA Journal 46 (7) (2008) 1803–1813.

[33] D. Amsallem, C. Farhat, An online method for interpolating linear parametric reduced-order models, SIAM Journal on Scientific Computing 33 (5) (2011) 2169–2198.

[34] N. T. Son, A real time procedure for affinely dependent parametric model order reduction using interpolation on Grassmann manifolds, International Journal for Numerical Methods in Engineering 93 (8) (2013) 818–833.

[35] R. Zimmermann, A locally parametrized reduced-order model for the linear frequency domain approach to time-accurate computational fluid dynamics, SIAM Journal on Scientific Computing 36 (3) (2014) B508–B537.

[36] R. Zhang, S. Mak, D. Dunson, Gaussian process subspace regression for model reduction, SIAM J. Sci. Comput. 44 (3) (2022) A1428–A1449.

[37] H. Lu, D. Ermakova, H. M. Wainwright, L. Zheng, D. M. Tartakovsky, Data-informed emulators for multi-physics simulations, Journal of Machine Learning for Modeling and Computing 2 (2) (2021) 33–54.

[38] K. Bhattacharya, B. Hosseini, N. B. Kovachki, A. M. Stuart, Model reduction and neural networks for parametric PDEs, The SMAI Journal of Computational Mathematics 7 (2021) 121–157.

[39] P. Sentz, K. Beckwith, E. C. Cyr, L. N. Olson, R. Patel, Reduced basis approximations of parameterized dynamical partial differential equations via neural networks, arXiv preprint arXiv:2110.10775 (2021).

[40] S. L. Brunton, B. W. Brunton, J. L. Proctor, J. N. Kutz, Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control, PloS One 11 (2) (2016) e0150171.

[41] J. H. Tu, Dynamic mode decomposition: Theory and applications, Ph.D. thesis, Princeton University (2013).

[42] M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, Science 324 (5923) (2009) 81–85.

[43] W.-X. Wang, R. Yang, Y.-C. Lai, V. Kovanis, C. Grebogi, Predicting catastrophes in nonlinear dynamical systems by compressive sensing, Physical Review Letters 106 (15) (2011) 154101.

[44] Q. Li, F. Dietrich, E. M. Bollt, I. G. Kevrekidis, Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator, Chaos: An Interdisciplinary Journal of Nonlinear Science 27 (10) (2017) 103111.

[45] P.-A. Absil, R. Mahony, R. Sepulchre, Riemannian geometry of Grassmann manifolds with a view on algorithmic computation, Acta Applicandae Mathematica 80 (2) (2004) 199–220.

[46] W. M. Boothby, W. M. Boothby, An introduction to differentiable manifolds and Riemannian geometry, Revised, Vol. 120, Gulf Professional Publishing, 2003.

[47] S. Helgason, Differential Geometry, Lie Groups, and Symmetric Spaces, Graduate Texts in Mathematics, AMS, 2001.

[48] I. U. Rahman, I. Drori, V. C. Stodden, D. L. Donoho, P. Schröder, Multiscale representations for manifold-valued data, Multiscale Modeling & Simulation 4 (4) (2005) 1201–1232.

[49] A. Edelman, T. A. Arias, S. T. Smith, The geometry of algorithms with orthogonality constraints, SIAM Journal on Matrix Analysis and Applications 20 (2) (1998) 303–353.

[50] D. Carmo, M. Perdigao, F. J. Flaherty, Riemannian geometry, Vol. 6, Springer, 1992.

[51] R. Zimmermann, Manifold interpolation, in: P. Benner, S. Grivet-Talocia, A. Quarteroni, G. R. W. Schilders, L. M. Silveira (Eds.), Model Order Reduction, Vol. 1 System- and Data-Driven Methods and Algorithms, De Gruyter, 2021, Ch. 7, pp. 229–274.

[52] H. Späth, One dimensional spline interpolation algorithms, AK Peters/CRC Press, 1995.

[53] C. De Boor, A. Ron, Computational aspects of polynomial interpolation in several variables, Mathematics of Computation 58 (198) (1992) 705–727.

[54] C. F. Van Loan, G. Golub, Matrix computations, 3rd Edition, Johns Hopkins studies in mathematical sciences, Johns Hopkins University Press, 1996.

[55] D. J. Ewins, Modal testing: theory, practice and application, John Wiley & Sons, 2009.

[56] M. O. Williams, I. G. Kevrekidis, C. W. Rowley, A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition, Journal of Nonlinear Science 25 (6) (2015) 1307–1346.

[57] J. Johns, A Matlab code for numerical solution of Navier-Stokes equations for two-dimensional incompressible flow (velocity-pressure formulation) along with ability for importing custom scenarios for the fluid flow, `https://github.com/JamieMJohns/Navier-stokes-2D-numerical-solve-incompressible-flow-with-custom-scenarios-MATLAB-` (2018).

[58] C. Lieberman, K. Willcox, O. Ghattas, Parameter and state model reduction for large-scale statistical inverse problems, SIAM Journal on Scientific Computing 32 (5) (2010) 2523–2542.

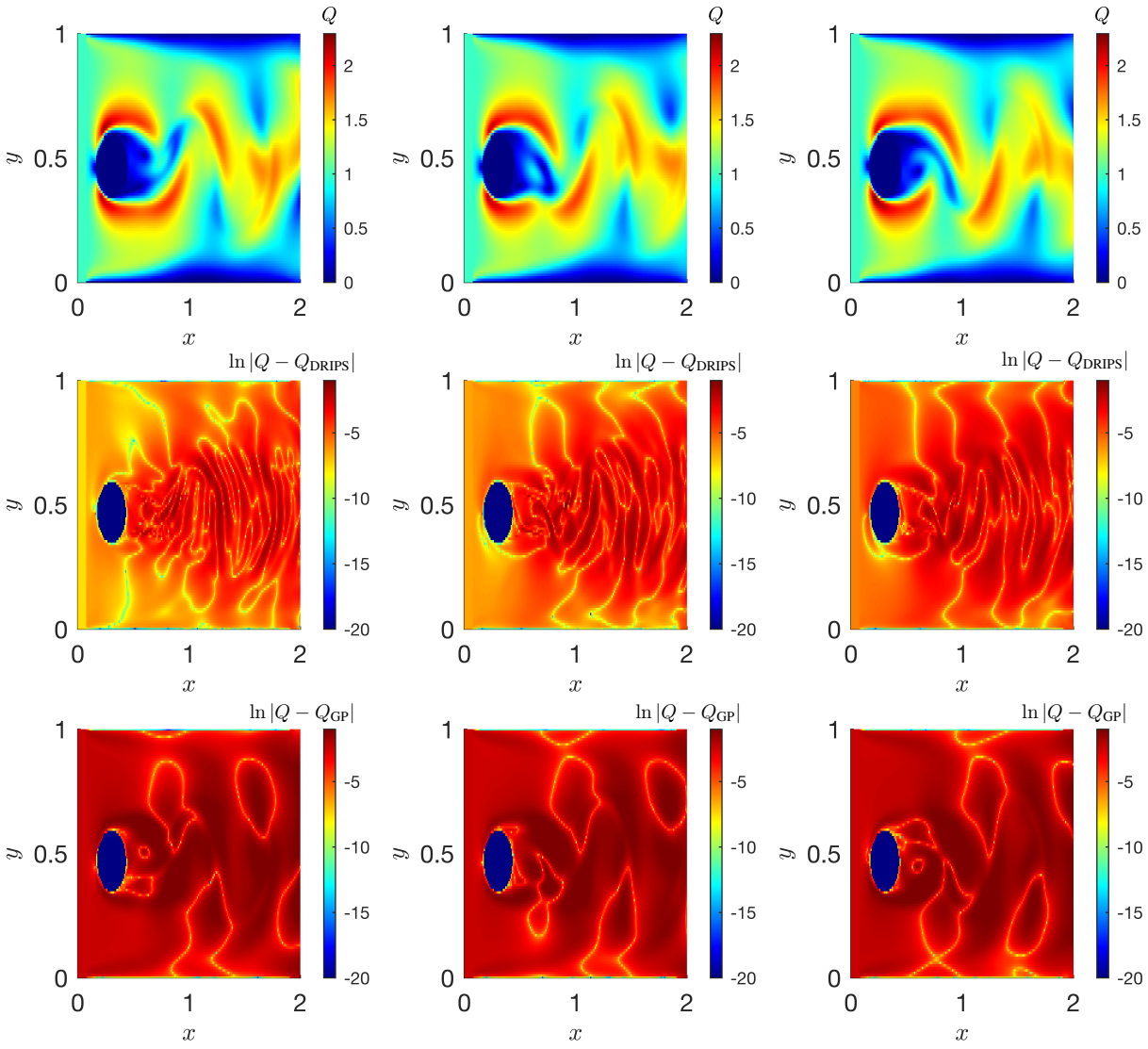

Figure 10: Temporal snapshots, at times $t = 4.125$ (left column), $t = 4.3125$ (middle column) and $t = 4.5$ (right column), of the reference solution of (4.12) with $\mathbf{p}^* = \{0.9934, 1.0094, 0.0014\}$ (top row), and the absolute errors in its estimation via DRIPS (middle row) and Kriging (bottom row) in logarithmic scale.

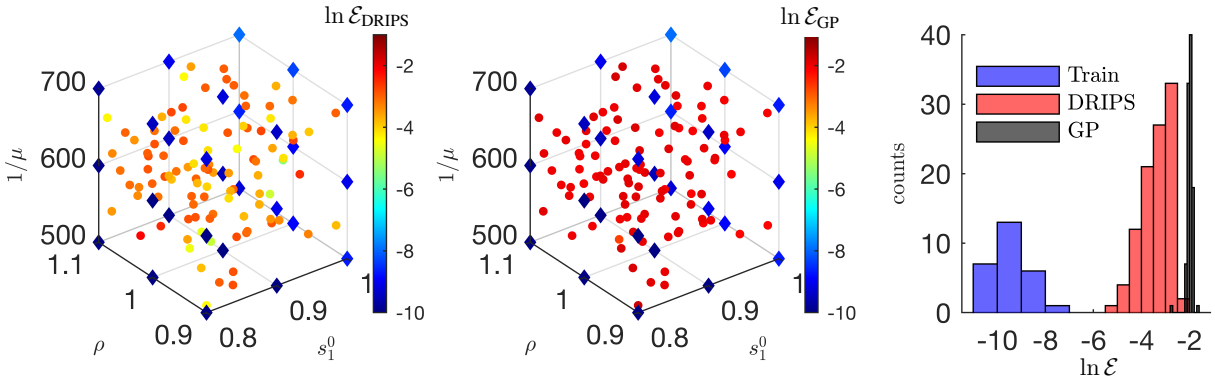

Figure 11: (Left) Log total relative error of the DRIPS estimator, $\ln(\mathcal{E}_{\text{DRIPS}}(\mathbf{p}^*))$; (Middle) log total relative error of the Kriging estimators, $\ln(\mathcal{E}_{\text{GP}}(\mathbf{p}^*))$; and (Right) histogram of the log total relative error of the two estimators, for several test values of the parameter set $\mathbf{p}^\star$ and the 27 training values of the parameter set on the Cartesian grid.