SIAM J. SCI. COMPUT.
Vol. 44, No. 3, pp. A1428–A1449

GAUSSIAN PROCESS SUBSPACE PREDICTION FOR MODEL REDUCTION[*]

RUDA ZHANG[†], SIMON MAK[‡], AND DAVID DUNSON[§]

Abstract. Subspace-valued functions arise in a wide range of problems, including parametric reduced order modeling (PROM), parameter reduction, and subspace tracking. In PROM, each parameter point can be associated with a subspace, which is used for Petrov–Galerkin projections of large system matrices. Previous efforts to approximate such functions use interpolations on manifolds, which can be inaccurate and slow. To tackle this, we propose a novel Bayesian nonparametric model for subspace prediction: the Gaussian process subspace (GPS) model. This method is extrinsic and intrinsic at the same time: with multivariate Gaussian distributions on the Euclidean space, it induces a joint probability model on the Grassmann manifold, the set of fixed-dimensional subspaces. The GPS adopts a simple yet general correlation structure, and a principled approach for model selection. Its predictive distribution admits an analytical form, which allows for efficient subspace prediction over the parameter space. For PROM, the GPS provides a probabilistic prediction at a new parameter point that retains the accuracy of local reduced models, at a computational complexity that does not depend on system dimension, and thus is suitable for online computation. We give four numerical examples to compare our method to subspace interpolation, as well as two methods that interpolate local reduced models. Overall, GPS is the most data efficient, more computationally efficient than subspace interpolation, and gives smooth predictions with uncertainty quantification.

Key words. Gaussian process, Grassmann manifold, parameter adaptation, reduced order modeling, subspace, uncertainty quantification

AMS subject classifications. 14M15, 35B30, 37M99, 37M99, 60B20, 60G15

DOI. 10.1137/21M1432739

1. Introduction.

In this paper we propose a method to solve the following formal problem. Consider a subspace-valued mapping $f : \Theta \mapsto G_{k,n}$ from a parameter space $\Theta \subset \mathbb{R}^d$ to the Grassmann manifold $G_{k,n}$, which is the set of all k-dimensional subspaces of the Euclidean space $\mathbb{R}^n$. Given function evaluations at l points, $(\boldsymbol{\theta}_i, \mathfrak{X}_i = f(\boldsymbol{\theta}_i))_{i=1}^l$, construct a probabilistic surrogate model g such that $g(\boldsymbol{\theta}_*)$ is a probability distribution on $G_{k,n}$ concentrated near $f(\boldsymbol{\theta}_*)$ for any point $\boldsymbol{\theta}_* \in \Theta$.

1.1. Motivation.

Numerical models can accurately predict many phenomena in science and engineering, with wide-ranging applications such as turbomachinery [29], ocean modeling [48], and biomedicine [10]. Yet, high-fidelity models must resolve multiple physics, multiple scales, complex geometry, and stochasticity. This leads to large-scale dynamical systems that incur major computational costs, especially when they need to be solved repeatedly. Other applications require real-time or embedded computing based on limited computational resources. In both cases, one needs to reduce the cost of solving large systems of differential equations. Reduced order modeling (ROM) approximates the full model with a reduced order model, which is

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section July 12, 2021; accepted for publication (in revised form) February 15, 2022; published electronically June 2, 2022.
https://doi.org/10.1137/21M1432739
Funding: This work was supported by the NSF through grant DMS-1638521 and by the ONR through grant N00014-21-1-2510-01.
[†]Department of Mathematics, Duke University, Durham, NC 27710 USA (ruda.zhang@duke.edu).
[‡]Department of Statistical Science, Duke University, Durham, NC 27710 USA (sm769@duke.edu).
[§]Department of Mathematics and Department of Statistical Science, Duke University, Durham, NC 27710 USA (dunson@duke.edu).

a much smaller system of differential equations that takes significantly less time and storage to simulate. ROM often provides a speedup of several orders of magnitude and has been used in many types of problems in scientific computing [8].

In many use cases, the full model itself depends on some parameters, to allow variations in material, geometry, loading, initial conditions, or boundary conditions. However, the accuracy of reduced models often declines quickly as parameters change, so we want to develop a reduced model that is also a function of the parameters. This is called parametric reduced order modeling (PROM), which is useful for design, control, optimization, uncertainty quantification, and inverse problems. Since most ROM methods are based on Petrov–Galerkin projection, which projects the model state space onto a low-dimensional subspace, one approach to PROM is to approximate the mapping from the parameters to such subspaces [8].

Subspace-valued mappings also arise in other areas of scientific computing. Active subspace methods [12] reduce the input space of a real-valued function to a low-dimensional subspace called the active subspace. For functional outputs, e.g., spatially varying fields or time series, such active subspaces become a function of space or time. Time-varying subspaces also arise in subspace tracking [13] and ROM [6].

1.2. Previous methods. A natural idea to solve this problem is to interpolate subspaces as a function of the parameters. However, this is infeasible since the Grassmann manifold is not a vector space and linear combination is undefined. To circumvent this difficulty, [3] proposed a method that takes the interpolation to tangent spaces of the Grassmann manifold, which are vector spaces. It comes in three steps. Given a target parameter point, it chooses a few nearby parameter points and maps the associated subspaces to the tangent space of one of them via the Riemannian logarithm. Then the tangent vectors are interpolated as a function of the parameters, using any traditional interpolation method. Finally, the interpolated tangent vector is mapped back to the Grassmann manifold via the Riemannian exponential, which gives the predicted subspace. We will refer to this method as *subspace interpolation*. In fact, this three-step approach applies to any Riemannian manifold, as long as effective algorithms exist for the Riemannian exponential and logarithm maps [2, 4]. This approach is extrinsic, i.e., referring to other sets and structures, which introduces distortions to the map.

Another type of method uses the Riemannian center of mass of weighted data points. The global or local Riemannian center of mass is the set of global or local minimizers of the sum of weighted squared Riemannian distances [1]. As before, the parameter-dependent weights can use any interpolation scheme such as splines [19] or Lagrange polynomials [40], both of which were introduced in the context of geodesic finite elements. Similarly, in the statistics literature, [37] proposed global and local regression models with predictors in a Euclidean space and random responses in a metric space. These methods are intrinsic, i.e., involving operations entirely on the manifold, so they avoid the limitations of mapping to a tangent space. However, their computation requires iterative algorithms for Riemannian optimization, and only local minimizers can be found. So far their uses are mostly for low-dimensional manifolds, with limited applications in PROM [35].

Zimmermann [49] reviewed interpolation methods on the Grassmann manifold and other matrix manifolds in the context of model reduction. More recently, he introduced Hermite interpolation of parameterized curves on Riemannian manifolds [50], which uses derivative data. All these methods are deterministic, while probabilistic methods for subspace approximation have not been explored in the literature.

1.3. Contribution. We propose a new Gaussian process (GP) model for the approximation of subspace-valued functions, which we call the Gaussian process subspace (GPS) model. Instead of using differential geometric structures of the Grassmann manifold as in [3], the GPS uses matrix-variate Gaussian distributions on the Euclidean space to induce a probability model on the Grassmann manifold. Our method therefore yields a probabilistic prediction of the subspace response, with intrinsic characterization of its predictive mean and uncertainty. Specifically, the mean prediction is a k-subspace of the span of the observed subspaces, and the latter also covers most of the predictive uncertainty. This GP model is flexible and yet well-guided: it can be used with any correlation function on the parameter space, and the function form and hyperparameters can be optimized via specific model selection criteria.

The main advantages of our method are summarized as follows. (1) *Data efficient*: Accurate prediction requires only a small sample size l, even when subspace dimension k and parameter dimension d are large. (2) *Computationally efficient*: Its prediction cost does not depend on ambient dimension n, and thus it is suitable for large-scale problems and online computation. (3) *Flexible*: It is a flexible Bayesian nonparametric model that is robust against model misspecification. (4) It provides *uncertainty quantification*, which gives a probabilistic description of a predicted subspace.

In our observation, GPS is much more accurate than subspace interpolation [3], which is in turn much more accurate than other PROM methods [4, 36]. Such data efficiency can be attributed to two factors. First, our method is intrinsic, so it does not suffer from distortions due to pulling back the mapping to a tangent space. Second, it has clear rules for model selection, while the other methods are often subject to model misspecification, due to arbitrary choices of reference point, subsample points, and interpolation schemes.

1.4. Related work. The authors have worked on estimating functions whose domains or codomains are manifolds. For inputs on an unknown embedded submanifold, [44] proposed a GP model that attains the minimax-optimal convergence rate, without estimating the manifold. To allow for noisy inputs and better scalability, [21] first projects the input to random subspaces, and then applies a GP model. For inputs on a known embedded submanifold, [25] proposed an extrinsic GP, while [33] proposed an intrinsic GP, with heat kernel as the covariance function. For outputs on an embedded submanifold, [26] proposed a non-GP method, which applies an extrinsic local regression and then obtains manifold estimates via projection [46].

While our method extends GPs to mappings that take values in the Grassmann manifold, we are not the first to define GPs on Riemannian manifolds. Wrapped Gaussian process regression [30] approximates mappings to a general Riemannian manifold, using distributions induced by Gaussian distributions on tangent spaces. However, this approach encounters problems when the manifold has a finite injectivity radius, as is the case for Grassmann manifolds. In particular, one cannot calculate the induced probability density function (PDF) on the manifold or the intrinsic mean. In contrast, our proposed approach produces analytic forms for predictive quantities that admit efficient computation, albeit restricted to Grassmann manifolds.

1.5. Article structure and notation. Section 2 provides basics of the algebra and statistics of some matrix manifolds. Section 3 presents the theoretical foundation of our GPS model, and section 4 gives an algorithm for prediction. Section 5 discusses model selection for our model. Section 6 overviews ROM and discusses the use of GPS in PROM in the context of existing methods. Section 7 gives several numerical experiments: one to visualize the posterior process, and three to access its accuracy in benchmark PROM problems. Section 8 concludes with a discussion on practical issues. Additional text is included in the Supplementary Materials. An `R` package accompanying this paper is available at https://github.com/rudazhang/gpsr.

Notation. Scalars are in lowercase, n, k, l, d; vectors in boldface lowercase, $\mathbf{m}, \mathbf{x}_i$, $\boldsymbol{\theta}$; and matrices in boldface uppercase, $\mathbf{M}, \mathbf{X}_i, \mathbf{K}_l$. Sets are in nonboldface uppercase, $\Theta, G_{k,n}$; subspaces in Fraktur script, $\mathfrak{X}, \mathfrak{M}$; and equivalence classes in brackets, $[\mathbf{M}], [\mathbf{m}]$.

2. Preliminaries.

Because we are building a probabilistic surrogate of subspace-valued mappings, it is helpful to review the algebra and statistics of the Grassmann manifold and some related matrix manifolds. For some basics of the algebra and differential geometry, see, e.g., [7, 47]; for a reference on the statistics, see [11].

2.1. Matrix manifolds.

Let $M_{n,k}$ be the set of all n-by-k real matrices, which can be identified as the Euclidean space $\mathbb{R}^{n\times k}$. The set of all full-rank n-by-k matrices is $M^*_{n,k} = \{\mathbf{M} \in M_{n,k} : \operatorname{rank}(\mathbf{M}) = \min(n, k)\}$. When $k = n$, it coincides with the general linear group GL_n, which consists of full-rank order-n matrices.

The Stiefel manifold $V_{k,n}$ consists of all orthonormal k-frames in the Euclidean n-space: $V_{k,n} = \{\mathbf{X} \in M^*_{n,k} : \mathbf{X}^T\mathbf{X} = \mathbf{I}_k\}$, where $k \le n$ and $\mathbf{I}_k$ is the order-k identity matrix. The order of the subscripts is reversed by convention. When $k = n$, the Stiefel manifold coincides with the orthogonal group $O(n)$. Define projection $\pi : M^*_{n,k} \mapsto V_{k,n}$ such that for any $\mathbf{M} \in M^*_{n,k}$ with a thin singular value decomposition (SVD) $\mathbf{M} = \mathbf{V\Sigma U}^T$, $\mathbf{V} \in V_{k,n}$, $\mathbf{U} \in O(k)$, we have $\pi(\mathbf{M}) = \mathbf{VU}^T$. Although the SVD is not unique, this mapping is uniquely defined.

The Grassmann manifold $G_{k,n}$ consists of all k-subspaces of the Euclidean n-space: $G_{k,n} = \{\operatorname{span}(\mathbf{M}) : \mathbf{M} \in M^*_{n,k}\}$, where $\operatorname{span}(\mathbf{M})$ denotes the subspace spanned by the columns of $\mathbf{M}$. Every element of $G_{k,n}$ is a subspace, which is often represented by a basis. For example, every $\mathbf{M} \in M^*_{n,k}$ represents $\mathfrak{M} = \operatorname{span}(\mathbf{M})$, the column vectors of $\mathbf{M}$ form a basis of $\mathfrak{M}$, and every element in its equivalence class $[\mathbf{M}] = \{\mathbf{MA} : \mathbf{A} \in \mathrm{GL}_k\}$ represents $\mathfrak{M}$ as well. We call $\mathbf{M}$ a basis representation of $\mathfrak{M}$. In particular, every $\mathbf{X} \in V_{k,n}$ represents $\mathfrak{X} = \operatorname{span}(\mathbf{X})$, and its column vectors form an orthonormal basis of $\mathfrak{X}$. We call $\mathbf{X}$ a Stiefel representation of $\mathfrak{X}$.

The Grassmann manifold is often identified with the set of rank-k symmetric projection matrices $P_{k,n}$: let $\mathcal{S}(n)$ be the set of order-n symmetric matrices, and define $P_{k,n} = \{\mathbf{P} \in \mathcal{S}(n) : \mathbf{P}^2 = \mathbf{P}, \operatorname{rank}(\mathbf{P}) = k\}$. This identification is possible because span() is a bijection from $P_{k,n}$ to $G_{k,n}$. Given a Stiefel representation $\mathbf{X}$, a subspace $\mathfrak{X}$ can thus be uniquely identified as $\mathbf{XX}^T$. Due to this explicit identification, probability distributions on the Grassmann manifold can be induced through distributions on $P_{k,n}$, with the corresponding PDF being $p : P_{k,n} \mapsto \mathbb{R}_{\ge 0}$, $\int_{P_{k,n}} p(\mathbf{P})\mu(d\mathbf{P}) = 1$, where μ is the normalized invariant measure on $P_{k,n}$ under the group action of GL_n.

2.2. Probability distributions.

Let $\mathcal{S}_+(n)$ be the set of order-n positive-definite matrices. Let $\mathbf{M} \in M_{n,k}$, $\boldsymbol{\Sigma}_1 \in \mathcal{S}_+(n)$, and $\boldsymbol{\Sigma}_2 \in \mathcal{S}_+(k)$. The n-by-k matrix-variate Gaussian distribution $N_{n,k}(\mathbf{M}; \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ is the distribution of $\mathbf{Y} = \boldsymbol{\Sigma}_1^{1/2}\mathbf{Z}\boldsymbol{\Sigma}_2^{1/2} + \mathbf{M}$, where $\mathbf{Z}$ is a random n-by-k matrix whose entries are independent standard Gaussian random variables. The vectorized matrix $\mathbf{Y}$ is an (nk)-dimensional Gaussian random vector with a special form of covariance matrix: $\operatorname{vec}(\mathbf{Y}) \sim N_{nk}(\operatorname{vec}(\mathbf{M}), \boldsymbol{\Sigma}_2 \otimes \boldsymbol{\Sigma}_1)$, where vec() denotes vectorization of a matrix by stacking its columns, and $\otimes$ is the Kronecker product.

The matrix angular central Gaussian distribution $\mathrm{MACG}(\mathbf{\Sigma})$ is a probability distribution on $V_{k,n}$, with PDF $p(\mathbf{X};\mathbf{\Sigma}) = z^{-1}|\mathbf{X}^T\mathbf{\Sigma}^{-1}\mathbf{X}|^{-n/2}$, where $|\cdot|$ denotes the determinant, normalizing constant $z = |\mathbf{\Sigma}|^{k/2}$, and parameter $\mathbf{\Sigma} \in \mathcal{S}_+(n)$. This parametric family contains the uniform distribution: since $p(\mathbf{X};\mathbf{I}_n) = 1$, we have $\mathrm{MACG}(\mathbf{I}_n) \sim$ Uniform. The parameter of the MACG distribution is identified up to scaling: for all $\mathbf{\Sigma} \in \mathcal{S}_+(n)$ and $c \in \mathbb{R}_{>0}$, $\mathrm{MACG}(\mathbf{\Sigma}) = \mathrm{MACG}(c\mathbf{\Sigma})$.

Any probability distribution on $M_{n,k}$ or $V_{k,n}$ that is invariant under right-orthogonal transformation induces a probability distribution on $G_{k,n}$ [11, Thm 2.4.8]: let p be a PDF on $M_{n,k}$ such that $p(\mathbf{M}) = p(\mathbf{MQ})$ for all $\mathbf{M} \in M_{n,k}$ and $\mathbf{Q} \in O(k)$, if $\mathbf{M} \sim p$, let $\mathbf{X} = \pi(\mathbf{M}) \sim p_V$ and $\mathbf{XX}^T \sim p_G$, then $p_V(\mathbf{X}) = p_V(\mathbf{XQ})$ for all $\mathbf{Q} \in O(k)$, and $p_G(\mathbf{XX}^T) = p_V(\mathbf{X})$. Because the MACG distribution on $V_{k,n}$ is invariant under right-orthogonal transformation, it defines a family of distributions on $G_{k,n}$ with the same PDF. We call it the MACG distribution on $G_{k,n}$.

These three distributions are related: let $\mathbf{M} \sim N_{n,k}(0;\mathbf{\Sigma},\mathbf{I}_k)$ where $\mathbf{\Sigma} \in \mathcal{S}_+(n)$; let $\mathbf{X} = \pi(\mathbf{M})$, then $\mathbf{X} \sim \mathrm{MACG}(\mathbf{\Sigma})$ and $\mathbf{XX}^T \sim \mathrm{MACG}(\mathbf{\Sigma})$. Due to this property, one can easily sample $\mathrm{MACG}(\mathbf{\Sigma})$: generate $\mathbf{M} \sim N_{n,k}(0;\mathbf{\Sigma},\mathbf{I}_k)$, and project it via π.

3. Gaussian process subspace prediction.

We now present the proposed GPS model. Because GP models take values in Euclidean spaces, they are not directly applicable to approximate subspace-valued mappings $f : \Theta \mapsto G_{k,n}$, where the codomain is the Grassmann manifold. Instead, we may find vector-valued mappings $\overline{f} : \Theta \mapsto \mathbb{R}^{nk}$ that are representations of f, in the sense that $f = \mathrm{span} \circ \mathrm{vec}^{-1} \circ \overline{f}$. Here, $\circ$ denotes the composition of two mappings and $\mathrm{vec}^{-1} : \mathbb{R}^{nk} \mapsto M_{n,k}$ denotes the "inverse" of vec(), that is, constructing a matrix columnwise from a vector. Such representations are not unique, and we denote the set of representations as $\overline{F} = \{\overline{f} : f = \mathrm{span} \circ \mathrm{vec}^{-1} \circ \overline{f}\}$. Now f can be identified with $\overline{F}$, or equivalently, any distribution supported on $\overline{F}$.

GP models extend naturally to approximate distributions on a set of functions. Let $\mathfrak{X} = f(\boldsymbol{\theta})$ with a basis representation $\mathbf{X}$. Recall that $\mathbf{X}$ has an equivalence class $[\mathbf{X}] = \{\mathbf{XA} : \mathbf{A} \in \mathrm{GL}_k\}$. Let $\mathbf{x} = \mathrm{vec}(\mathbf{X})$, whose equivalence class can be written as $[\mathbf{x}] = \{\mathrm{vec}(\mathbf{XA}) : \mathbf{A} \in \mathrm{GL}_k\}$. Assuming that $\overline{f}$ has a GP prior, we may assign equal likelihood to $[\mathbf{x}]$. We can then proceed to derive the posterior and the predictive distributions. In the following, we provide modeling details and analytical solutions for this approach.

3.1. Model specification.

We start by specifying a prior for the representations. Without other information on f, an uninformative prior is for $f(\boldsymbol{\theta})$ to be uniformly distributed on $G_{k,n}$. We can achieve this by assigning $\overline{f}(\boldsymbol{\theta}) \sim N_{nk}(0, \mathbf{I}_{nk})$, the nk-dimensional standard Gaussian. To see this, let matrix $\mathbf{M} = \mathrm{vec}^{-1}(\overline{f}(\boldsymbol{\theta}))$, then $\mathbf{M} \sim N_{n,k}(0;\mathbf{I}_n,\mathbf{I}_k)$ is a matrix-variate standard Gaussian; let subspace $\mathfrak{M} = \mathrm{span}(\mathbf{M})$, then $\mathfrak{M} \sim \mathrm{MACG}(\mathbf{I}_n) \sim$ Uniform. We assign a correlation structure as follows. Let $k : \Theta \times \Theta \mapsto [-1, 1]$ be a correlation function, i.e., a positive-definite kernel with $k(\boldsymbol{\theta},\boldsymbol{\theta}) = 1$ for all $\boldsymbol{\theta} \in \Theta$. For any finite collection of input points $\boldsymbol{\theta} = (\boldsymbol{\theta}_i)_{i=1}^l$, let $\mathbf{m}_i = \overline{f}(\boldsymbol{\theta}_i)$, and let $\mathbf{K}_l$ be the order-l correlation matrix with entry $[\mathbf{K}_l]_{ij} = k(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j)$. We assign the function values $\mathbf{m} = (\mathbf{m}_i)_{i=1}^l$ a prior joint distribution $\mathbf{m} \sim N_{nkl}(0, \mathbf{K}_l \otimes \mathbf{I}_{nk})$. Compactly, we can write this GP prior as $\overline{f} \sim \mathcal{GP}(0, k \otimes \mathbf{I}_{nk})$. This is the simplest covariance structure for $\overline{f}$.

Without a likelihood function, this GP prior gives predictions as follows. Let $\boldsymbol{\theta}_*$ be a target point and $\mathbf{m}_* = \overline{f}(\boldsymbol{\theta}_*)$. We have the prior joint distribution:

$$(\mathbf{m}_*, \mathbf{m}) \sim N_{nk(l+1)}(0, \mathbf{K}_{l+1} \otimes \mathbf{I}_{nk}), \tag{3.1}$$

where $\mathbf{K}_{l+1} = [1\ \mathbf{k}_l^T; \mathbf{k}_l\ \mathbf{K}_l]$ and $\mathbf{k}_l = (k(\boldsymbol{\theta}_*, \boldsymbol{\theta}_i))_{i=1}^l$. If we write $\mathbf{K}_{22} = \mathbf{K}_l \otimes \mathbf{I}_{nk}$ and $\mathbf{K}_{12} = \mathbf{k}_l^T \otimes \mathbf{I}_{nk}$, by properties of multivariate Gaussian distributions, the conditional distribution of $\mathbf{m}_*$ given $\mathbf{m}$ can be written as

$$\begin{aligned}\mathbf{m}_*|\mathbf{m} &\sim N_{nk}(\mathbf{K}_{12}\mathbf{K}_{22}^{-1}\mathbf{m}, \mathbf{I}_{nk} - \mathbf{K}_{12}\mathbf{K}_{22}^{-1}\mathbf{K}_{12}^T) \\ &= N_{nk}\left(\sum_{i=1}^{l}[\mathbf{K}_l^{-1}\mathbf{k}_l]_i\mathbf{m}_i, (1 - \mathbf{k}_l^T\mathbf{K}_l^{-1}\mathbf{k}_l)\mathbf{I}_{nk}\right).\end{aligned} \tag{3.2}$$

We assign equal likelihood to the equivalence class of representations. Assume that we have function evaluations $\mathfrak{X}_i = f(\boldsymbol{\theta}_i)$ with Stiefel representations $\mathbf{X}_i \in V_{k,n}$. Let $\mathbf{x}_i = \mathrm{vec}(\mathbf{X}_i)$ and $[\mathbf{x}_i] = \{\mathrm{vec}(\mathbf{X}_i\mathbf{A}) : \mathbf{A} \in \mathrm{GL}_k\}$. For $\mathbf{m}_i = \overline{f}(\boldsymbol{\theta}_i)$, the likelihood function gives

$$L(\mathbf{m}_i|\mathfrak{X}_i) = 1(\mathbf{m}_i \in [\mathbf{x}_i]). \tag{3.3}$$

The posterior distribution of $\mathbf{m}$ given observations $\mathfrak{X} = (\mathfrak{X}_i)_{i=1}^l$ is derived from the prior and the likelihood (3.3) via the Bayes' rule:

$$p(\mathbf{m}|\mathfrak{X}) \propto \exp\left\{-\frac{1}{2}\mathbf{m}^T(\mathbf{K}_l \otimes \mathbf{I}_{nk})^{-1}\mathbf{m}\right\}\prod_{i=1}^{l} 1(\mathbf{m}_i \in [\mathbf{x}_i]). \tag{3.4}$$

3.2. Predictive distributions. The predictive distribution of $\mathbf{m}_*$ given observations $\mathfrak{X}$ is obtained by integrating the conditional distribution (3.2) over the posterior distribution (3.4). We summarize the result as follows.

THEOREM 3.1. *Let* $\mathbf{X} = [\mathbf{X}_1 \ \cdots \ \mathbf{X}_l]$ *be the matrix that combines* $\mathbf{X}_i$ *by columns, and* $\mathbb{X} = \mathrm{diag}(\mathbf{X}_i)_{i=1}^l$ *be the matrix with* $\mathbf{X}_i$ *as diagonal blocks. Let* $\varepsilon^2 = 1 - \mathbf{k}_l^T\mathbf{K}_l^{-1}\mathbf{k}_l$ *and* $\mathbf{v} = \mathbf{K}_l^{-1}\mathbf{k}_l$. *If* $\mathbf{v} \in \mathbb{R}^l_{\neq 0}$,[1] *let* $\mathbf{D}_\mathbf{v} = \mathrm{diag}(\mathbf{v})$ *and* $\widetilde{\mathbf{K}}_l = (\mathbf{D}_\mathbf{v}\mathbf{K}_l\mathbf{D}_\mathbf{v})^{-1}$. *The predictive distribution of* $\mathbf{m}_*$ *is*

$$\begin{aligned}&\mathbf{m}_*|\mathfrak{X} \sim N_{nk}(0, \mathbf{I}_k \otimes \boldsymbol{\Sigma}), \\ &\boldsymbol{\Sigma} = \varepsilon^2\mathbf{I}_n + \mathbf{X}[\mathbb{X}^T(\widetilde{\mathbf{K}}_l \otimes \mathbf{I}_n)\mathbb{X}]^{-1}\mathbf{X}^T.\end{aligned} \tag{3.5}$$

The proof is quite lengthy and thus deferred to section SM1. This theorem shows that, given observations (1) the matrix $\mathbf{M}_* = \mathrm{vec}^{-1}(\mathbf{m}_*)$ has a matrix-variate Gaussian distribution, $\mathbf{M}_*|\mathfrak{X} \sim N_{n,k}(0; \boldsymbol{\Sigma}, \mathbf{I}_k)$; and (2) the subspace $\mathfrak{M}_* = \mathrm{span}(\mathbf{M}_*)$ has an MACG distribution, $\mathfrak{M}_*|\mathfrak{X} \sim \mathrm{MACG}(\boldsymbol{\Sigma})$ (see subsection 2.2).

The predictive distributions admit an intuitive interpretation. Since $\boldsymbol{\Sigma}$ is positive semidefinite, there is an eigenvalue decomposition (EVD) $\boldsymbol{\Sigma} = \mathbf{Q}\,\mathrm{diag}(\boldsymbol{\lambda})\mathbf{Q}^T$, where $\boldsymbol{\lambda} \in \mathbb{R}^n_{\geq 0}$ are in decreasing order and $\mathbf{Q} \in O(n)$. Therefore we can simulate $\mathbf{M}_*|\mathfrak{X}$ as $\mathbf{M}_* = \boldsymbol{\Sigma}^{1/2}\mathbf{Z} = \mathbf{Q}\,\mathrm{diag}(\boldsymbol{\lambda})^{1/2}\mathbf{Q}^T\mathbf{Z}$, where $\mathbf{Z} \in M_{n,k}$ is a random matrix of standard Gaussians. The columns of $\mathbf{Z}$ are scaled by the square root of the eigenvalue in each eigenspace; therefore the range (i.e., column space) of $\mathbf{M}_*$ is more likely to align with the top eigenspaces of $\boldsymbol{\Sigma}$. Recall that $\mathfrak{M}_* = \mathrm{span}(\mathbf{M}_*)$. We have the following results. (1) The global Riemannian center of mass of $\mathfrak{M}_*|\mathfrak{X}$ is $\mathrm{span}(\mathbf{V})$, where $\mathbf{V}$ is the first k columns of $\mathbf{Q}$. (2) The uncertainty of $\mathfrak{M}_*|\mathfrak{X}$ is compactly described by the eigenvalues

[1]The condition of no zero entry in $\mathbf{v}(\boldsymbol{\theta})$ holds almost everywhere in Θ, but it breaks most notably when predicting at sample points, $\mathbf{v}(\boldsymbol{\theta}_i) = \mathbf{e}_i$, which means $\boldsymbol{\Sigma}$ is singular at sample points and close to singular nearby. In these cases, one needs to be careful with matrix inversion in implementing the prediction algorithm in section 4.

$\boldsymbol{\lambda}$: the larger an eigenvalue is, the more important is the associated eigenspace; and the mean prediction is more useful if $(\lambda_i)_{i=k+1}^n$ are small relative to $(\lambda_i)_{i=1}^k$.

A main feature of our GP model is that, while its construction involves the extrinsic Euclidean space $\mathbb{R}^{nk}$ of basis representations of subspaces, its predictive distribution is intrinsic to the Grassmann manifold $G_{k,n}$. In particular, our model does not involve tangent spaces or the Riemannian exponential, and thus it is not subject to the distortions associated with applying local tangent approximations. Moreover, the function space explored by the GPS is much broader than the existing interpolation methods, so our model is more flexible and robust to model misspecification. Perhaps surprisingly, the GPS has closed-form expressions for its predictive distributions, which enables efficient computation for subspace prediction and uncertainty quantification.

While Theorem 3.1 is concerned with point predictions on the Grassmann manifold, our GPS model also induces joint distributions on $G_{k,n}$ and can be used to generate random subspace-valued functions (see section SM2).

4. Prediction algorithm.

From Theorem 3.1 and the discussion thereafter we see that, to compute the predictive distribution, one needs the EVD of $\boldsymbol{\Sigma}$. Even with $\boldsymbol{\Sigma}$ available, the EVD would cost $\mathcal{O}(n^3)$, which is intractable for large n. Here we give an efficient method to compute this.

4.1. Efficient EVD of $\boldsymbol{\Sigma}$.

Denote $\boldsymbol{\Pi} = \mathbb{X}^T(\widetilde{\mathbf{K}}_l \otimes \mathbf{I}_n)\mathbb{X}$ and $\check{\boldsymbol{\Sigma}} = \mathbf{X}\boldsymbol{\Pi}^{-1}\mathbf{X}^T$. We note that $\widetilde{\mathbf{K}}_l, \boldsymbol{\Pi} > 0$ and $\check{\boldsymbol{\Sigma}} \geq 0$. Let $r = \operatorname{rank}(\mathbf{X}) \leq \min(n, kl)$, then $\check{\boldsymbol{\Sigma}}$ also has rank r and therefore r positive eigenvalues. From the form of $\check{\boldsymbol{\Sigma}}$, we see that its top-$r$ eigenvectors span the range of $\mathbf{X}$. Let $\mathbf{X} = \widetilde{\mathbf{V}}\widetilde{\mathbf{R}}\widetilde{\mathbf{P}}^T$ be a rank-revealing QR decomposition, such that $\widetilde{\mathbf{V}} \in V_{r,n}$ has r orthonormal columns, $\widetilde{\mathbf{R}} \in M_{r,kl}$ is upper triangular, and $\widetilde{\mathbf{P}}$ is a permutation matrix. Denote order-r matrix $\mathbf{S} = \widetilde{\mathbf{V}}^T\check{\boldsymbol{\Sigma}}\widetilde{\mathbf{V}}$ and let $\mathbf{S} = \mathring{\mathbf{Q}} \operatorname{diag}(\mathring{\boldsymbol{\lambda}})\mathring{\mathbf{Q}}^T$ be an EVD where $\mathring{\boldsymbol{\lambda}}$ is descending and $\mathring{\mathbf{Q}} \in O(r)$. Let $\mathbf{V} = \widetilde{\mathbf{V}}\mathring{\mathbf{Q}}$ and let $\mathbf{Q} = (\mathbf{V}, \mathbf{V}_\perp) \in O(n)$ be an orthogonal completion. Let $\check{\boldsymbol{\lambda}} = (\mathring{\boldsymbol{\lambda}}, \mathbf{0}_{n-r})$ where $\mathbf{0}_{n-r}$ is the vector of zeros with length $n-r$. Then we have an EVD: $\check{\boldsymbol{\Sigma}} = \mathbf{Q} \operatorname{diag}(\check{\boldsymbol{\lambda}})\mathbf{Q}^T$. Because $\boldsymbol{\Sigma} = \check{\boldsymbol{\Sigma}} + \varepsilon^2\mathbf{I}_n$, we have an EVD of $\boldsymbol{\Sigma}$:

$$\boldsymbol{\Sigma} = \mathbf{Q} \operatorname{diag}(\check{\boldsymbol{\lambda}} + \varepsilon^2\mathbf{1}_n)\mathbf{Q}^T. \tag{4.1}$$

Here $\mathbf{1}_n$ is the vector of ones with length n. We see that, for a complete probabilistic prediction, we only need a rank-revealing QR of $\mathbf{X}$, an EVD of $\mathbf{S}$, and ε^2. For the mean prediction, we only need the top-k eigenvectors of $\mathbf{S}$.

We can simplify the computation of $\mathbf{S}$ as follows. Note that $\widetilde{\mathbf{V}}^T\mathbf{X} = \widetilde{\mathbf{R}}\widetilde{\mathbf{P}}^T$ and $\widetilde{\mathbf{P}}^{-1} = \widetilde{\mathbf{P}}^T$. Because $\mathbf{S} = \widetilde{\mathbf{V}}^T\check{\boldsymbol{\Sigma}}\widetilde{\mathbf{V}}$ and $\check{\boldsymbol{\Sigma}} = \mathbf{X}\boldsymbol{\Pi}^{-1}\mathbf{X}^T$, we have $\mathbf{S} = \widetilde{\mathbf{R}}(\widetilde{\mathbf{P}}\boldsymbol{\Pi}\widetilde{\mathbf{P}}^T)^{-1}\widetilde{\mathbf{R}}^T$. Let order-$(kl)$ Gram matrix $\square = \mathbf{X}^T\mathbf{X}$, which has a block matrix structure $\square = [\square_{ij}]_{i,j=1}^l$ with $\square_{ij} = \mathbf{X}_i^T\mathbf{X}_j$. Note that $\boldsymbol{\Pi}$ similarly has a block matrix structure $\boldsymbol{\Pi} = [\boldsymbol{\Pi}_{ij}]_{i,j=1}^l$ with $\boldsymbol{\Pi}_{ij} = \widetilde{k}_{ij}\square_{ij}$, where $\widetilde{k}_{ij} = [\widetilde{\mathbf{K}}_l]_{i,j}$. The construction of $\boldsymbol{\Pi}$ can be written in a compact form: $\boldsymbol{\Pi} = \square \circ (\widetilde{\mathbf{K}}_l \otimes \mathbf{J}_k)$, where $\circ$ denotes the Hadamard product and $\mathbf{J}_k = \mathbf{1}_k\mathbf{1}_k^T$ is the order-k matrix of ones. Let $\widetilde{\boldsymbol{\Pi}} = \widetilde{\mathbf{P}}\boldsymbol{\Pi}\widetilde{\mathbf{P}}^T$ and let $\widetilde{\boldsymbol{\Pi}} = \mathbf{L}\mathbf{L}^T$ be a Cholesky decomposition, where $\mathbf{L} \in M_{kl,kl}$ is lower triangular. Let $\widetilde{\mathbf{L}} = \mathbf{L}^{-1}\widetilde{\mathbf{R}}^T \in M_{kl,r}$ by solving linear equations, which is also lower triangular; then we have $\mathbf{S} = \widetilde{\mathbf{L}}^T\widetilde{\mathbf{L}}$.

We formally describe the prediction procedure in two parts: Algorithm 4.1 only needs to be done once, and Algorithm 4.2 is needed for each prediction.

Algorithm 4.1. GPS: Preprocessing

```
Input: observation $\mathbf{X} = [\mathbf{X}_1 \ \cdots \ \mathbf{X}_l]$.
1: Compute Gram matrix: $\Box \leftarrow \mathbf{X}^T\mathbf{X}$.
2: Rank-revealing QR: $\mathbf{X} = \widetilde{\mathbf{V}}\widetilde{\mathbf{R}}\widetilde{\mathbf{P}}^T$.
Output: Gram matrix $\Box$; global basis $\widetilde{\mathbf{V}}$; upper triangular $\widetilde{\mathbf{R}}$; pivoting $\widetilde{\mathbf{P}}$.
```

Algorithm 4.2. GPS: Prediction

```
Require: correlation function $k(\cdot,\cdot)$; preprocessing output $(\Box, \widetilde{\mathbf{V}}, \widetilde{\mathbf{R}}, \widetilde{\mathbf{P}})$.
Input: sample $(\boldsymbol{\theta}_i)_{i=1}^l$; target $\boldsymbol{\theta}_*$; truncation size $t \in \{k, k+1, \ldots, r\}$.
1: Construct correlation matrix and vector: $k_{ij} \leftarrow k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$, $k_i \leftarrow k(\boldsymbol{\theta}_*, \boldsymbol{\theta}_i)$.
2: Solve linear equations: $\mathbf{v} \leftarrow \mathrm{solve}(\mathbf{K}, \mathbf{k})$, $\widehat{\mathbf{K}} \leftarrow \mathrm{solve}(\mathbf{K}, \mathrm{diag}(\mathbf{v})^{-1})$.
3: Construct matrix: $\boldsymbol{\Pi} \leftarrow [\boldsymbol{\Pi}_{ij}]_{i,j=1}^l$, where $\boldsymbol{\Pi}_{ij} \leftarrow v_i^{-1}\widehat{k}_{ij}\Box_{ij}$.
4: Cholesky decomposition: $\widetilde{\mathbf{P}}\boldsymbol{\Pi}\widetilde{\mathbf{P}}^T = \mathbf{L}\mathbf{L}^T$.
5: Solve linear equations: $\widetilde{\mathbf{L}} \leftarrow \mathrm{solve}(\mathbf{L}, \widetilde{\mathbf{R}}^T)$
6: Cross product: $\mathbf{S} \leftarrow \widetilde{\mathbf{L}}^T\widetilde{\mathbf{L}}$.
7: Truncated EVD: $\mathbf{S} = \mathring{\mathbf{V}}\,\mathrm{diag}(\mathring{\boldsymbol{\lambda}})\mathring{\mathbf{V}}^T$, where $\mathring{\boldsymbol{\lambda}}$ has length $t$.
8: Compute noise variance: $\varepsilon^2 \leftarrow 1 - \mathbf{k}^T\mathbf{v}$.
Output: principal directions $\mathbf{V} = \widetilde{\mathbf{V}}\mathring{\mathbf{V}}$; principal variances $\mathring{\boldsymbol{\lambda}}$; noise variance $\varepsilon^2$.
Note: May return $\widetilde{\mathbf{V}}$ and $\mathring{\mathbf{V}}$ instead of $\mathbf{V}$ to avoid matrix multiplication.
```

4.2. Computational cost. Here we analyze the computational cost of each step in floating point operations (flops), accurate up to the dominant term. In Algorithm 4.1, line 1 takes nk^2l^2 flops; line 2 takes $\mathcal{O}(nklr)$ flops, and if $r \approx kl$, this requires about $4nk^2l^2$ flops using the Householder QR with column pivoting [16]. In Algorithm 4.2, line 1 evaluates the correlation function $l^2/2$ times; line 2 takes $l^3/3$ flops for Cholesky decomposition, and $2l^3$ for forward and back substitution; line 3 takes $k^2l^2/2$ flops; line 4 takes $k^3l^3/3$ flops; line 5 takes $k^3l^3/3 - (kl-r)^3/3$ flops, due to the upper triangular structure in $\widetilde{\mathbf{R}}$; line 6 takes $r^3/3 + (kl-r)r^2$ flops, due to the lower triangular structure in $\widetilde{\mathbf{L}}$; line 7 takes $\mathcal{O}(r^2t)$ with classical or randomized algorithms [22]; and line 8 takes $2l$ flops. Note that $\mathbf{K}$ and its Cholesky decomposition can be reused for future predictions. Overall, with $n > kl$ and assuming $r \approx kl$ and $t = k$, Algorithm 4.1 gives an overhead cost of about $5nk^2l^2$ flops if we use the Householder QR with column pivoting, and Algorithm 4.2 gives a cost of about k^3l^3 flops per prediction.

An alternative version of Algorithm 4.2 is to conduct a truncated SVD, $\widetilde{\mathbf{L}} = \mathring{\mathbf{V}}\,\mathrm{diag}(\mathring{\boldsymbol{\sigma}})\mathbf{W}^T$, and then return $\mathring{\mathbf{V}}$ and $\mathring{\boldsymbol{\lambda}} = \mathring{\boldsymbol{\sigma}}^2$. Although this avoids the cross product in line 6 and thus saves about $k^3l^3/3$ flops, the truncated SVD can take a significant amount of time and eliminate the savings. Theoretically, the truncated SVD takes $\mathcal{O}(rklt)$ with classical algorithms and $\mathcal{O}(rkl\log t)$ with randomized algorithms [22]. But in practice, the truncated SVD appears to be more costly than the truncated EVD. Since truncated SVD gives a less accurate result than truncated EVD, we consider Algorithm 4.2 as the reference version.

Note that the matrix multiplication $\mathbf{V} = \widetilde{\mathbf{V}}\mathring{\mathbf{V}}$ takes $2nrk$ flops for $t = k$, which would dominate the prediction cost if $n > kl^2/2$. However, this cost can be avoided if $\mathbf{V}$ is not explicitly needed. In PROM problems, to compute an order-k reduced matrix $\mathbf{A}_k = \mathbf{V}^T\mathbf{A}\mathbf{V}$, one may precompute an order-r matrix $\mathbf{A}_r = \widetilde{\mathbf{V}}^T\mathbf{A}\widetilde{\mathbf{V}}$, and then compute $\mathbf{A}_k = \mathring{\mathbf{V}}^T\mathbf{A}_r\mathring{\mathbf{V}}$. Since $\mathbf{A}$ is usually sparse, the cost of a matrix-vector multiplication $\mathbf{Ax}$ is usually $T_{\text{mult}} = \mathcal{O}(n)$. Then this approach has an overhead cost of $2nk^2l^2 + klT_{\text{mult}}$ flops, and only takes about $2k^3l^2$ flops per prediction.

5. Model selection.

To make predictions with a GP model, we need to specify a covariance function; this is called model selection. Although the kernel $k(\cdot,\cdot)$ can be arbitrary, it is often specified in a form that depends on some hyperparameters [38, Chap. 4]. For example, the squared exponential (SE) kernel is

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\beta}) = \prod_{i=1}^{d} \exp\left[-\frac{(\theta_i - \theta_i')^2}{2\beta_i^2}\right], \tag{5.1}$$

where length-scales $\boldsymbol{\beta} = (\beta_i)_{i=1}^d$ are the hyperparameters. GP models with the SE kernel are smooth, and the length-scales can be understood as characteristic distances along each parameter dimension before the function values become uncorrelated.

One can set the hyperparameters to optimize a certain criterion; see, e.g., [38, sect. 5.4] and [41, sect. 3.3]. For GPS, we recommend minimizing the leave-one-out cross validation (LOOCV) predictive error, measured in Riemannian distances. (Other distances between subspaces may be used as well, but we choose Riemannian distance for concreteness.) In this section we analyze and give an algorithm to compute this criterion. Section SM3 provides a procedure to compute its gradient, and section SM4 discusses some alternative criteria.

A rule-of-thumb length-scale. In our experience, the predictive performance of GPS is not very sensitive to hyperparameters, so one may use certain default values to trade accuracy for reduced computational cost. For the SE kernel, one may set the length-scales to $3d^{3/2}/l$ relative to the parameter ranges and expect good predictions.

5.1. LOOCV predictive error.

To measure predictive error, we need a score of dissimilarity for pairs of subspaces. There are many metrics defined on the Grassmann manifold; see, e.g., [45] for a list. Among them, the most commonly used is the Riemannian distance, which is the length of the shortest curves connecting two points in a Riemannian manifold. The Riemannian distance between subspaces $\mathfrak{X}, \mathfrak{Y} \in G_{k,n}$ is the 2-norm of their principal angles, which can be computed as [7]

$$d_g(\mathfrak{X}, \mathfrak{Y}) = \|\arccos \boldsymbol{\sigma}(\mathbf{X}^T\mathbf{Y})\|. \tag{5.2}$$

Here, $\mathbf{X}, \mathbf{Y} \in V_{k,n}$ are representations of the subspaces, and $\boldsymbol{\sigma}(\cdot)$ denotes the singular values of a matrix. Let $\mathbf{V}_{-i}$ represent the mean prediction for target $\boldsymbol{\theta}_i$, using the remaining data points $(\boldsymbol{\theta}_j, \mathbf{X}_j)_{j\neq i}$. The LOOCV predictive error can be defined as

$$L_{\text{LOO}} = \sum_{i=1}^{l} d_g^2(\mathbf{X}_i, \mathbf{V}_{-i}) = \sum_{i=1}^{l}\sum_{j=1}^{k} \left(\arccos \sigma_j(\mathbf{X}_i^T\mathbf{V}_{-i})\right)^2. \tag{5.3}$$

Here we use the sum of squared errors for its smoothness and, with a slight abuse of notation, we replace the subspaces with their Stiefel representations.

5.2. Efficient computation of L_{LOO}.

To compute the LOOCV predictive error in (5.3), we need $\mathbf{X}_i^T\mathbf{V}_{-i}$. First we derive a form of $\mathbf{V}_{-i}$. Analogous to (3.5), for the leave-one-out prediction we have

$$\boldsymbol{\Sigma}_{-i} = \varepsilon_{-i}^2\mathbf{I}_n + \mathbf{X}_{-i}[\mathbb{X}_{-i}^T(\widetilde{\mathbf{K}}_{-i}\otimes\mathbf{I}_n)\mathbb{X}_{-i}]^{-1}\mathbf{X}_{-i}^T. \tag{5.4}$$

Here, all the quantities are defined without the ith observation. Similar to the analysis in subsection 4.1, denote $\boldsymbol{\Pi}_{-i} = \mathbb{X}_{-i}^T(\widetilde{\mathbf{K}}_{-i}\otimes\mathbf{I}_n)\mathbb{X}_{-i}$ and $\check{\boldsymbol{\Sigma}}_{-i} = \mathbf{X}_{-i}(\boldsymbol{\Pi}_{-i})^{-1}\mathbf{X}_{-i}^T$. Let

$r_{-i} = \operatorname{rank}(\mathbf{X}_{-i})$; then the top-$r_{-i}$ eigenvectors of $\check{\boldsymbol{\Sigma}}_{-i}$ span the range of $\mathbf{X}_{-i}$, which is a subset of the range of $\mathbf{X}$. Recall that $\mathbf{X} = \widetilde{\mathbf{V}}\widetilde{\mathbf{R}}\widetilde{\mathbf{P}}^T$ is a rank-revealing QR. Let $\mathbf{S}_{-i} = \widetilde{\mathbf{V}}^T\check{\boldsymbol{\Sigma}}_{-i}\widetilde{\mathbf{V}}$ and let $\mathbf{S}_{-i} \approx \mathring{\mathbf{V}}_{-i}\operatorname{diag}(\mathring{\boldsymbol{\lambda}}_{-i})\mathring{\mathbf{V}}_{-i}^T$ be a rank-k truncated EVD; then $\widetilde{\mathbf{V}}\mathring{\mathbf{V}}_{-i}$ are the top-k eigenvectors of $\check{\boldsymbol{\Sigma}}_{-i}$. Since $\mathbf{V}_{-i}$ consists of the top-k eigenvectors of $\boldsymbol{\Sigma}_{-i}$ and $\boldsymbol{\Sigma}_{-i} = \varepsilon_{-i}^2\mathbf{I}_n + \check{\boldsymbol{\Sigma}}_{-i}$, we have $\mathbf{V}_{-i} = \widetilde{\mathbf{V}}\mathring{\mathbf{V}}_{-i}$.

To avoid big matrix multiplication, let $\widetilde{\mathbf{C}} = \widetilde{\mathbf{V}}^T\mathbf{X} = \widetilde{\mathbf{R}}\widetilde{\mathbf{P}}^T$, which has the form $\widetilde{\mathbf{C}} = [\widetilde{\mathbf{C}}_1 \ \cdots \ \widetilde{\mathbf{C}}_l]$ where $\widetilde{\mathbf{C}}_i = \widetilde{\mathbf{V}}^T\mathbf{X}_i \in M_{r,k}$. We have $\mathbf{X}_i^T\mathbf{V}_{-i} = \mathbf{X}_i^T\widetilde{\mathbf{V}}\mathring{\mathbf{V}}_{-i} = \widetilde{\mathbf{C}}_i^T\mathring{\mathbf{V}}_{-i}$. Similarly, let $\widetilde{\mathbf{C}}_{-i} = \widetilde{\mathbf{V}}^T\mathbf{X}_{-i} = [\cdots \ \widetilde{\mathbf{C}}_j \ \cdots]_{j\neq i}$, and we have $\mathbf{S}_{-i} = \widetilde{\mathbf{C}}_{-i}(\boldsymbol{\Pi}_{-i})^{-1}\widetilde{\mathbf{C}}_{-i}^T$.

We can express $\boldsymbol{\Pi}_{-i}$ using entries of $\mathbf{K}^{-1}$. Let $\mathbf{K}_{-i} = [k_{pq}]_{p,q\neq i}$ and $\mathbf{k}_{-i} = (k_{pi})_{p\neq i}$. Let $\overline{\mathbf{K}} = \mathbf{K}^{-1}$, $\overline{\mathbf{K}}_{-i} = [\overline{k}_{pq}]_{p,q\neq i}$, and $\overline{\mathbf{k}}_{-i} = (\overline{k}_{pi})_{p\neq i}$. We can write $\mathbf{v}_{-i} = (\mathbf{K}_{-i})^{-1}\mathbf{k}_{-i}$ as $\mathbf{v}_{-i} = -\overline{\mathbf{k}}_{-i}/\overline{k}_{ii}$ and $(\mathbf{K}_{-i})^{-1} = \overline{\mathbf{K}}_{-i} - \overline{k}_{ii}\mathbf{v}_{-i}\mathbf{v}_{-i}^T$ (see, for example, [38, sect. 5.4.2]). With $\widetilde{\mathbf{K}}_{-i} = (\mathbf{D}_{\mathbf{v}_{-i}}\mathbf{K}_{-i}\mathbf{D}_{\mathbf{v}_{-i}})^{-1}$ and $\mathbf{D}_{\mathbf{v}_{-i}} = \operatorname{diag}(\mathbf{v}_{-i})$, we have $\widetilde{\mathbf{K}}_{-i} = \overline{k}_{ii}^{-1}\Delta_{-i}$ where $\Delta_{-i} = [\overline{k}_{pq}\overline{k}_{ii}/(\overline{k}_{ip}\overline{k}_{iq}) - 1]_{p,q\neq i}$. Now we have $\boldsymbol{\Pi}_{-i} = \overline{k}_{ii}^{-1}\Box_{-i} \circ (\Delta_{-i} \otimes \mathbf{J}_k)$.

The computation of $\mathbf{S}_{-i}$ follows subsection 4.1. Since we are only concerned with the eigenvectors of $\mathbf{S}_{-i}$, with a little abuse of notation, we redefine $\boldsymbol{\Pi}_{-i}$ without the term $\overline{k}_{ii}^{-1}$. We describe the overall procedure in Algorithm 5.1.

Algorithm 5.1. LOOCV predictive error

```
Require: correlation function $k$; sample $(\boldsymbol{\theta}_i)_{i=1}^l$; preprocessing output $(\Box, \widetilde{\mathbf{C}} = \widetilde{\mathbf{R}}\widetilde{\mathbf{P}}^T)$.
Input: hyperparameters $\boldsymbol{\beta}$.
1: Construct inverse correlation matrix: $\overline{\mathbf{K}} \leftarrow \operatorname{solve}(\mathbf{K})$, where $k_{ij} \leftarrow k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j; \boldsymbol{\beta})$.
2: for $i$ in $1, \dots, l$ do
3:     Construct: $\boldsymbol{\Pi} \leftarrow [\boldsymbol{\Pi}_{pq}]_{p,q\neq i}$, where $\boldsymbol{\Pi}_{pq} \leftarrow \delta_{pq}\Box_{pq}$, $\delta_{pq} \leftarrow \overline{k}_{pq}\overline{k}_{ii}/(\overline{k}_{ip}\overline{k}_{iq}) - 1$.
4:     Construct: $\mathbf{S} \leftarrow \widetilde{\mathbf{L}}^T\widetilde{\mathbf{L}}$, where $\boldsymbol{\Pi} = \mathbf{L}\mathbf{L}^T$, $\widetilde{\mathbf{L}} \leftarrow \operatorname{solve}(\mathbf{L}, \widetilde{\mathbf{C}}_{-i}^T)$.
5:     Truncated EVD: $\mathbf{S} = \mathring{\mathbf{V}}\operatorname{diag}(\mathring{\boldsymbol{\lambda}})\mathring{\mathbf{V}}^T$, where $\mathring{\boldsymbol{\lambda}}$ has length $k$.
6:     Compute singular values: $\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma}(\widetilde{\mathbf{C}}_i^T\mathring{\mathbf{V}})$.
7:     Compute squared error: $\epsilon_i \leftarrow \sum_{j=1}^k \arccos(\sigma_j)^2$
8: end for
Output: LOOCV predictive error $L_{\text{LOO}} = \sum_{i=1}^l \epsilon_i$.
```

5.3. Computational cost. In terms of computation, Algorithm 5.1 is approximately l repetitions of Algorithm 4.2, so it costs about k^3l^4 flops per evaluation. This means that evaluating the LOOCV error takes about the same time as making l predictions. Because such evaluation needs to be repeated until numerical optimization converges, hyperparameter training may be a significant part of the overall cost. In practice, we recommend setting a very rough convergence threshold: for parameters with a range of one, a threshold of 0.01 is sufficient for the length-scale. If the problem has multiple parameters, they may be scaled into comparable ranges and share the same length-scale. If multiple hyperparameters are to be trained, gradient-based optimization methods (see section SM3) can be more efficient than just using the LOOCV error. To minimize the number of iterations, one may also set a restrictive range and, if applicable, a good initial value for the hyperparameters, for example, $\pm 30\%$ of the aforementioned rule-of-thumb length-scale, with initial value at the midpoint.

6. Application in model reduction.

In this section, we review the general setup of model reduction and compare the GPS with other methods for PROM.

6.1. Reduced order modeling.

To simplify the narrative, consider a system of ordinary differential equations (ODEs) that is first-order, linear, and time-invariant, with multiple input and output:

$$\Sigma : \begin{cases} \mathbf{E}\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \\ \mathbf{y} = \mathbf{C}\mathbf{x}. \end{cases} \tag{6.1}$$

With system dimension n, input dimension p, and output dimension q, this system is defined by constant matrices $\mathbf{E}, \mathbf{A} \in M_{n,n}$, $\mathbf{B} \in M_{n,p}$, and $\mathbf{C} \in M_{q,n}$. The state $\mathbf{x}$, input $\mathbf{u}$, and output $\mathbf{y}$ are all functions of time, with dimension n, p, and q, respectively. We assume $\mathbf{x}(0) = \mathbf{0}$; any fixed initial condition $\mathbf{x}_0$ can be included in the input as an impulse $\mathbf{x}_0\delta(t)$. In general, the ODE system Σ may represent a physical or artificial system modeled by a PDE system, which is discretized in space, and linearized around a stationary trajectory. The system dimension n typically scales with the size of a spatial grid, and for a large-scale problem, usually we have $n > 10^5$.

Projection-based model reduction constructs a reduced-order model (ROM) as

$$\Sigma_r : \begin{cases} \mathbf{E}_r\dot{\mathbf{x}}_r = \mathbf{A}_r\mathbf{x}_r + \mathbf{B}_r\mathbf{u}, \\ \mathbf{y}_r = \mathbf{C}_r\mathbf{x}_r. \end{cases} \tag{6.2}$$

Let $\mathbf{V}, \mathbf{W} \in V_{k,n}$ be orthonormal bases of k-dimensional subspaces, the reduced system matrices are defined as $\mathbf{E}_r = \mathbf{W}^T\mathbf{E}\mathbf{V}$, $\mathbf{A}_r = \mathbf{W}^T\mathbf{A}\mathbf{V}$, $\mathbf{B}_r = \mathbf{W}^T\mathbf{B}$, and $\mathbf{C}_r = \mathbf{C}\mathbf{V}$. Therefore we have $\mathbf{E}_r, \mathbf{A}_r \in M_{k,k}$, $\mathbf{B}_r \in M_{k,p}$, and $\mathbf{C}_r \in M_{q,k}$. If the reduced bases $\mathbf{V}$ and $\mathbf{W}$ are the same, this framework is called the Galerkin projection; otherwise, it is called the Petrov–Galerkin projection. Usually we would want a reduced system dimension $k \leq 50$. Because simulation time and model storage scale at least linearly with system dimension, they are reduced by several orders of magnitude via ROM.

6.2. Error measures.

To measure the error introduced by a ROM, one choice is the $\mathcal{L}_2$ state error for a given input. The $\mathcal{L}_2$ metric of square-integrable functions on the interval $[0, T]$, discretized into J parts of length δt, can be approximated as

$$\|\mathbf{x} - \hat{\mathbf{x}}\|_{\mathcal{L}_2}^2 = \int_0^T \|\mathbf{x}(t) - \hat{\mathbf{x}}(t)\|_2^2 \, dt \approx \sum_{i=1}^{J} \|\mathbf{x}(t_i) - \hat{\mathbf{x}}(t_i)\|_2^2 \, \delta t. \tag{6.3}$$

Relative $\mathcal{L}_2$ state error is the $\mathcal{L}_2$ error of the state of a ROM, divided by the $\mathcal{L}_2$ norm of the state of the original system. With approximated state $\hat{\mathbf{x}} = \mathbf{V}\mathbf{x}_r$, we have

$$e(\mathbf{x}, \mathbf{x}_r)_{\mathcal{L}_2} = \frac{\|\mathbf{x} - \mathbf{V}\mathbf{x}_r\|_{\mathcal{L}_2}}{\|\mathbf{x}\|_{\mathcal{L}_2}}. \tag{6.4}$$

Another error measure is the $\mathcal{H}_2$ metric, defined as the largest possible amplitude of the output error given any unit-energy input: with $\|\mathbf{y}\|_{\mathcal{L}_\infty} = \sup_{t \geq 0} \|\mathbf{y}(t)\|_\infty$,

$$\|\Sigma - \Sigma_r\|_{\mathcal{H}_2} = \sup_{\mathbf{u} \in \mathcal{L}_2} \frac{\|\mathbf{y} - \mathbf{y}_r\|_{\mathcal{L}_\infty}}{\|\mathbf{u}\|_{\mathcal{L}_2}}. \tag{6.5}$$

The $\mathcal{H}_2$ error of a ROM is, in a sense, more comprehensive than the $\mathcal{L}_2$ state error. Relative $\mathcal{H}_2$ error is the $\mathcal{H}_2$ error divided by the $\mathcal{H}_2$ norm of the original system:

$$e(\Sigma, \Sigma_r)_{\mathcal{H}_2} = \frac{\|\Sigma - \Sigma_r\|_{\mathcal{H}_2}}{\|\Sigma\|_{\mathcal{H}_2}}. \tag{6.6}$$

The $\mathcal{H}_2$ norms can be obtained analytically via the controllability Gramian, which can be computed by solving the Lyapunov equations [39].

6.3. Methods for ROM. To compute a reduced basis for the Galerkin projection, a widely used classic method is called the proper orthogonal decomposition (POD), originally proposed for turbulent flow analysis by [28]. This method takes a collection of system states $\mathbf{x}(t_i)$ at discrete times $\{t_i\}_{i=1}^m$, called snapshots, which may be obtained via simulation or experimental measurements. Let $\mathbf{X}$ be the matrix that stacks the snapshots as column vectors; then the POD basis $\mathbf{V}$ corresponds to the left singular vectors of $\mathbf{X}$ associated with the largest k singular values. This means that the POD basis minimizes the $\mathcal{L}_2$ error of snapshot reconstruction, which is an appealing property of POD. Besides providing a reduced basis, POD also associates each basis vector with the corresponding singular value, which can be used to determine basis dimension k. For large-scale systems, the number of snapshots required is far less than the system dimension, and usually $m = \mathcal{O}(10^3)$.

Another class of ROM methods are interpolatory [5], which approximate the transfer function of the original system using rational interpolation. The transfer function of the system Σ is defined as $\mathbf{H}(s) = \mathbf{C}(s\mathbf{E} - \mathbf{A})^{-1}\mathbf{B}$. Here, $\mathbf{H} : \mathbb{C} \mapsto M_{q,p}(\mathbb{C})$ is a complex matrix-valued function of a complex frequency variable. These methods interpolate the transfer function at an arbitrary number of points and up to an arbitrary number of derivatives along certain tangent directions. Among such methods, the iterative rational Krylov algorithm (IRKA) introduced by [20] has seen great success. It iteratively searches for an order-k rational function that approximates the transfer function, until it satisfies the tangential interpolation conditions. If IRKA converges, the converged point locally minimizes the $\mathcal{H}_2$ error in the space of order-k rational functions. IRKA constructs a ROM in state space via the two-sided Petrov–Galerkin projection, that is, the reduced bases $\mathbf{V}$ and $\mathbf{W}$ are different.

Besides POD and interpolatory methods, there are other ROM methods such as balanced truncation [31], most common in systems and control theory. There are effective ROM methods for systems more general than (6.1) as well, such as DEIM [9] for nonlinear systems and DMD [42] for black-box systems.

6.4. Methods for PROM. Our discussion so far assumes that the full model Σ in (6.1) is constant. In a more general class of problems, Σ is parametric, such that the system matrices $\mathbf{E}, \mathbf{A}, \mathbf{B}$, and $\mathbf{C}$ depend on a set of parameters $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^d$. This dependency can be nonlinear in general, and the dimension d of the parameter space varies greatly with the problem. There are many methods for PROM, and we refer the readers to [8] for a comprehensive review.

One approach is to construct a single basis that works well for the entire parametric set of systems. For example, given local reduced bases $(\mathbf{V}_i)_{i=1}^l$ obtained for a sample of the parameter space, one can concatenate them into a global basis $\mathbf{V} = [\mathbf{V}_1 \cdots \mathbf{V}_l]$. However, this increases the dimension of the reduced subspace, and therefore the size and simulation time of the ROM.

Another approach is to consider a projection-based ROM method as a mapping that associates each parameter point with a reduced subspace. Given local reduced subspaces at a parameter sample, one may approximate this subspace-valued mapping and predict reduced subspaces at other parameter points. This fits the problem in section 1 and includes subspace interpolation and our GPS method. Compared with using a global basis, this keeps the ROM small and often more reliable [3].

Instead of interpolating subspaces, [36] proposed a method that directly interpolates the reduced models: it first applies a congruence transformation to the local reduced models, and then interpolates the model matrices elementwise. We will refer to this approach as *matrix interpolation*. Influenced by this work, [4] proposed a method that interpolates the transformed matrices on a relevant matrix manifold, e.g., the general linear group, in a procedure analogous to subspace interpolation. We will refer to this approach as *manifold interpolation*.

An idea bridging the global and local approaches is parameter domain partitioning: one can partition the parameter space into small regions and apply a PROM method within each. This idea has been adopted in many papers; see, e.g., [2, 14].

6.5. Comparing PROM methods. Here we compare the GPS with other methods in model reduction, in terms of speed, accuracy, and property preservation.

6.5.1. Speed versus local bases. Our method is typically much faster than methods for computing local reduced bases. Consider the computation of a local POD basis given m snapshots at one parameter point. The cost is dominated by a truncated SVD of the n-by-m snapshot matrix, which takes $\mathcal{O}(nmk)$ time. To compare the costs, take the rocket injector example in [29], where $n \approx 10^5$, $m = 10^3$, $k = 45$, $l = 30$. We have $(nmk)/(k^3 l^3) \approx 1.83$. Considering the constant factor in truncated SVD, in this case our method is about an order of magnitude faster than computing a local POD basis. Because the cost of computing snapshots dominates the overall POD procedure, this implies a clear advantage in using our method to approximate local POD bases.

The cost of computing a pair of local IRKA bases is less straightforward to analyze [5]. Every iteration needs to solve $2k$ systems of linear equations, each with a different coefficient matrix of order n that cannot be reused across iterations. The number of iterations depends on the initial values provided to the algorithm, and the algorithm needs to be restarted if it does not converge after a predefined maximum number of iterations. Depending on the problem, IRKA can take longer than the POD procedure.

6.5.2. Speed versus interpolatory methods. Subspace interpolation [3] uses the Riemannian exponential and logarithm of the Grassmann manifold, both involving a thin SVD of an n-by-k matrix, which scales with $\mathcal{O}(nk^2)$. Since its prediction does not have a special factorization structure (as the GPS does), it takes another $2nk^2 + kT_{\text{mult}}$ flops to compute a reduced matrix, where T_{mult} denotes the cost of a matrix-vector multiplication. The prediction cost can be greatly reduced if the problem has only one parameter and one uses linear interpolation [43]. In general, the prediction scales with n and is slow for large-scale problems.

Matrix interpolation [36] and manifold interpolation [4] directly interpolate local ROMs so their prediction costs do not depend on n, and therefore they are considered as suitable for online computation.

In comparison, our algorithm turns the truncated EVD of the order-n matrix $\mathbf{\Sigma}$ into one of the order-kl matrix $\mathbf{S}$, and the prediction cost is instead dominated by the construction of $\mathbf{S}$, which is carried out efficiently via matrix decomposition and linear solvers. Thus, the prediction cost also does not depend on n.

Table 1
Interpolatory methods for PROM: flop counts of the dominant terms.

	Preprocess	Subspace	ROM	Training	Reference
GPS	$5nk^2l^2$	k^3l^3	$2k^3l^2$	k^3l^4	this paper
Subspace-Int	$10nk^2l^2$	$8nk^2$	$2nk^2$	†	[3]
Matrix-Int	$6nk^2l^2$	-	$2k^2l$	†	[36]
Manifold-Int	nk^2l^2	-	$\mathcal{O}(k^3l)^*$	†	[4]

* Coefficient usually on the scale of 50 due to matrix logarithm/exponential, which can be numerically unstable [23].
† Optimal choice of reference ROM and interpolation scheme is an open problem.

Table 1 compares the computational costs of these methods in detail. This table does not include the generation of reduced bases at a sample of the parameter space, a step required by all these methods. Generating a reduced basis can be computationally expensive depending on the ROM method in use, which limits the sample size l.

6.5.3. Accuracy. All three interpolation methods lack a clear rule for model selection, i.e., selecting the reference point, other interpolation points, and the interpolation scheme. This often leads to model misspecification, which undermines accuracy. Moreover, interpolation on tangent spaces of Riemannian manifolds, such as subspace and manifold interpolation, are extrinsic to the underlying manifolds. As explained in section SM6, when points further away from the reference point are used, the true mapping becomes more distorted on the tangent space and thus harder to approximate. A similar concern is addressed in [50, sect. 3]. Therefore, these methods cannot use more than a handful of points at a time and have limited potential to extend to higher-dimensional parameter spaces.

Our method has specific model selection criteria which make it data efficient, so a small sample size is enough to give accurate results. Besides, the GPS is intrinsic to the Grassmann manifold, so it does not incur extra approximation error and its accuracy improves with sample size.

6.5.4. Preservation of properties. Another important issue in ROM is the preservation of system properties, such as stability, passivity, and contractivity. Although stability is not guaranteed for the reduced models generated by our method, from section 7 we will see that it is still observed in most cases, simply because our method can accurately approximate the subspace map of local ROMs.

7. Numerical experiments.

7.1. Visualization of GP subspace prediction. The simplest type of subspace-valued functions have the form $f : \mathbb{R} \mapsto G_{1,2}$, which maps a real number to a one-dimensional linear subspace in the plane. The Grassmann manifold $G_{1,2}$ can be identified as the unit circle, treating antipodal points as equivalent (Figure 1(a)). Therefore, such a function f can be plotted on the surface of a cylinder (Figure 1(b)), which helps us visualize the posterior process of the GPS model.

Specifically, let f be a covering map such that $f(\theta)$ is the subspace with angle $\alpha = \theta \bmod \pi$. This can be plotted as a double helix on the cylinder. To approximate this function with the proposed GPS model, suppose we observe sample points $\theta_i = c_i\pi$, where c_i are seven equal-distanced points between 0.2 and 1.8. For the correlation function k, we use the SE kernel and set the length-scale β by minimizing the LOOCV predictive error. In this example, $\beta = 2.8 \approx 0.9\pi$. To visualize predictive uncertainty, we plot the 95% posterior predictive intervals from Theorem 3.1. We also include

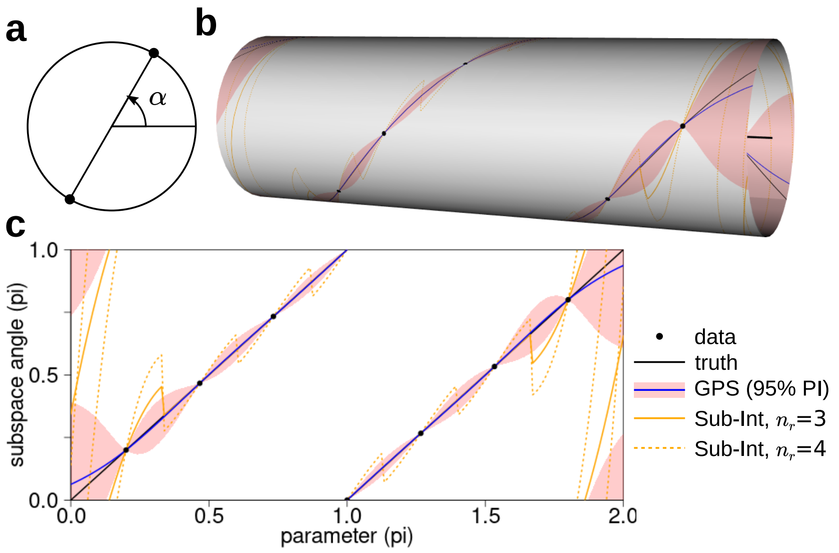

FIG. 1. *Visualization of the GPS model.* (a) *Every one-dimensional subspace in the plane can be uniquely identified by either a pair of antipodal points on a circle, or an angle* $\alpha \in [0, \pi)$. (b) *Posterior process of the GPS model on the surface of a cylinder.* (c) *Same as* (b) *but as a two-dimensional plot. True function (black line), data (black points), GPS predictive mean (blue curve), 95% predictive interval (red shade). Orange curves are predictions from subspace interpolation:* $n_r = 3$ *(solid),* $n_r = 4$ *(dotted).*

results from subspace interpolation for comparison. As suggested by the authors of [3], for every target parameter we use the nearest n_r sampled points for the interpolation (where $n_r = 3$ and 4 in Figure 1), among which the nearest sampled point is used as the reference point. We use Lagrange interpolation for the tangent vectors.

We see that, with only seven data points, the predictive mean function of GPS closely tracks the true function within the range of sampled parameter points. Furthermore, the uncertainties from our model also well cover the truth: the posterior predictive intervals contain the true subspace values for all $\theta \in [0, 2\pi]$. Note that as the target point moves away from the sample points, the predictive distribution degenerates to the prior, the uniform distribution on $G_{1,2}$. Subspace interpolation, on the other hand, yields noticeably poorer predictions compared to GPS for both $n_r = 3$ and $n_r = 4$. As a deterministic interpolation approach, it also does not provide a quantification of interpolation uncertainty. This shows that, for this example, the proposed GPS model uses sample data more effectively to yield better predictions with uncertainty quantification.

7.2. Anemometer: Approximating local POD bases. Here we consider a benchmark problem for PROM known as the anemometer [32], a type of microelectromechanical system device that measures the flow speed of its surroundings. Such a device needs to be calibrated under different flow conditions for its temperature response. However, an accurate representation of the device needs to resolve the coupled fluid and thermodynamics and can be very time-consuming to compute. It is therefore useful to apply PROM methods.

Specifically, a convection-diffusion equation is discretized into a linear ODE system as (6.1), with system dimension $n = 29{,}008$ and input and output dimensions $p = q = 1$. The matrix $\mathbf{A}$ depends on one parameter $\theta \in [0, 1]$ representing fluid velocity and is not symmetric in general, while $\mathbf{E}, \mathbf{B}, \mathbf{C}$ are constants. The input map $\mathbf{B}$ represents a heat source, and the output map $\mathbf{C}$ gives the temperature difference of two nodes.

To build a parametric reduced-order model (PROM), we first construct local POD bases at a sample of the parameter space, and then use the mean prediction of GPS to estimate the reduced subspaces at other parameter points. As before, we use the SE kernel, with a length-scale that minimizes the LOOCV predictive error. The subspace-valued mappings being approximated in this problem have very high dimensional codomains: because the dimension of $G_{k,n}$ is $k(n-k)$, with $k = 20$ and $k = 40$, the manifold dimensions here are 579,760 and 1,158,720, respectively.

For comparison, we also estimate the reduced subspaces using subspace interpolation, with the same setup as in the visualization example. For manifold interpolation [4], we use the same setup for subspace interpolation. For matrix interpolation [36], we use the nearest sampled point as the reference point and, as suggested by the authors, we use linear interpolation for the reduced system matrices. We include results for local POD bases as a reference level we would like to match.

Figure 2(a) shows the relative $\mathcal{H}_2$ errors using these methods, with subspace dimension $k = 20$. Here we use a sample of seven equal-distanced points from 0 to 1. GPS uses a length-scale $\beta = 0.36$, selected via LOOCV. The results for subspace and manifold interpolation use $n_r = 3$; the results are similar for $n_r = 4$ or 5. We see that the three existing interpolation methods perform similarly, and the errors tend to blow up in between sample points. In comparison, the proposed GPS model yields much lower errors: the relative $\mathcal{H}_2$ error is comparable to that for the local POD (the reference level). Note that the goal here is not to perfectly match the error curve of the local POD, but to keep the error as low as possible; in this sense, the GPS model appears to provide noticeable improvements over existing methods.

Figure 2(b) shows the results for $k = 40$. Here we use a sample of 11 equal-distanced points from 0 to 1. GPS uses a length-scale $\beta = 0.25$. Setup for the interpolation methods are unchanged. We see that, even with the increased sample size, all three interpolation methods fail to keep a low error level. While matrix inter-

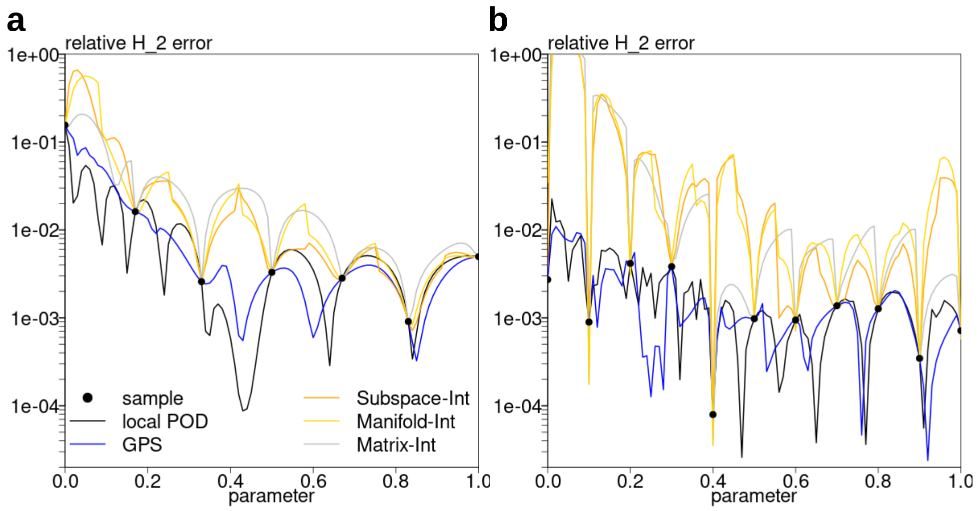

Fig. 2. *Anemometer, relative $\mathcal{H}_2$ error:* (a) $k = 20$*;* (b) $k = 40$*. Training data shown as points. The $\mathcal{H}_2$ error curve of local POD is wiggly because it minimizes the $\mathcal{L}_2$ state error.*

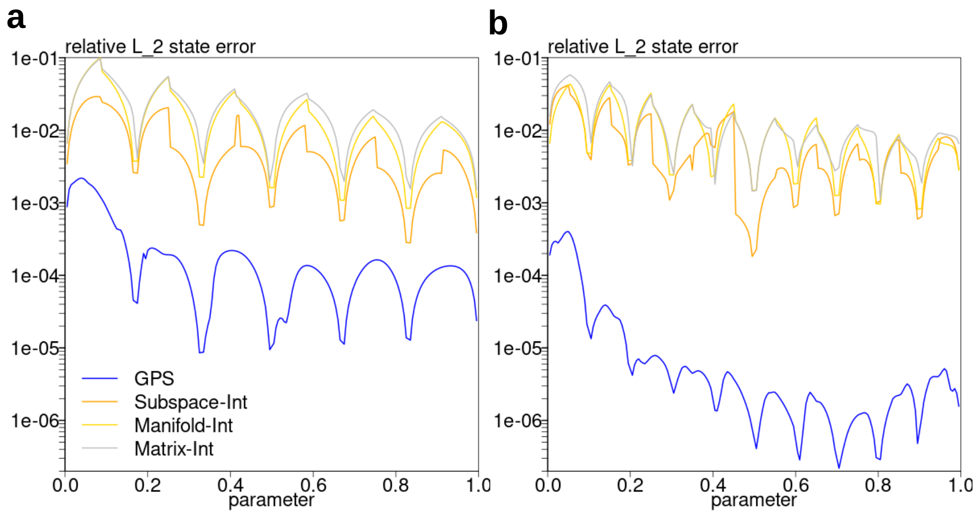

FIG. 3. *Anemometer, relative $\mathcal{L}_2$ state error:* (a) $k = 20$*;* (b) $k = 40$.

polation occasionally does better than the other two, this is probably not generalizable due to the linear interpolation scheme. In comparison, our method again yields much lower errors, and maintains a similar level of accuracy as the local POD.

Figure 3 shows the relative $\mathcal{L}_2$ state errors using these methods. The local POD is omitted from these plots since its relative $\mathcal{L}_2$ state error is practically zero. The error curves of the three interpolation methods are qualitatively similar, with subspace interpolation better than manifold interpolation, which is in turn better than matrix interpolation. In comparison, the GPS again yields much lower errors: for $k = 20$, the average error is about two orders of magnitude lower than that of subspace interpolation; for $k = 40$, it is about three orders of magnitude lower. This improvement can be attributed to the more flexible and intrinsic nature of the GPS model, which allows for more effective use of sample data.

Measured computation time for this problem is provided in section SM5.

7.3. Microthruster: Approximating local IRKA bases. Here we consider another benchmark problem for PROM known as the microthruster [34], an array of solid propellant microthrusters on a chip. To find an optimal design of array geometry and driving circuit, many simulations need to be carried out, which can be prohibitive with large-scale models. The use of PROM is therefore justified.

Specifically, the numerical model discretizes a heat transfer equation into a linear ODE system as (6.1), with system dimension $n = 4{,}257$, input dimension $p = 1$, and output dimension $q = 7$. The input $\mathbf{B}$ represents the electrical circuit, and the output $\mathbf{C}$ gives the temperature at seven nodes. The convection boundary conditions are parameterized into three parameters, each within the range $[1, 10^4]$, and affect the symmetric system matrix $\mathbf{A}$ on the diagonal. To simplify comparison, we fix the three parameters to always be the same, and take the base-10 logarithm of their original values, so we have one parameter $\theta \in [0, 4]$.

For this problem, we use IRKA to construct reduced bases at the sample points. Because IRKA uses two different bases $\mathbf{V}$ and $\mathbf{W}$, for a parametric system this means that each parameter is associated with a pair of subspaces, and we may construct a PROM by approximating a mapping for the form $(\mathfrak{V}, \mathfrak{W})(\boldsymbol{\theta})$. Since our proposed

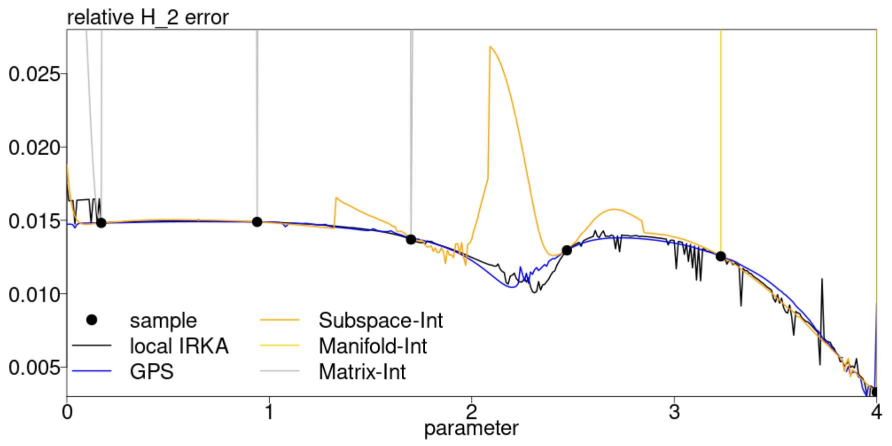

FIG. 4. *Microthruster, relative $\mathcal{H}_2$ error. $k = 10$. Training data are shown as points. The error curve of local IRKA is more level than local POD in Figure 2 because it minimizes the $\mathcal{H}_2$ error.*

method only handles mappings that output one subspace, we proceed by modeling the pair of subspaces separately. This inevitably leaves some information in the data unused, and there may be methods that can improve upon this work-around. Setup for the interpolation methods is the same as in the anemometer example.

Figure 4 shows the relative $\mathcal{H}_2$ errors using these methods, with subspace dimension $k = 10$. Here we use a sample of six points: $\theta = 0.17, 0.94, 1.7, 2.47, 3.23, 4$. GPS uses a length-scale $\beta = 1.4$ for basis $\mathbf{V}$ and $\beta = 2.56$ for basis $\mathbf{W}$. The result for subspace interpolation uses $n_r = 3$; the other values of n_r give results with larger errors. We see that while subspace interpolation matches the error curve of local IRKA (the reference level) quite well in some parts of the parameter space, its error blows up in an unsmooth region in between. These errors are noticeably larger for manifold and matrix interpolation, so we cropped them out of the plot. To contrast, the proposed GPS method instead tracks the local IRKA error curve smoothly across the parameter space, yielding much lower errors than existing interpolation methods.

For this problem, many of the ROMs generated by manifold interpolation are complex-valued, due to the matrix logarithm that computes the tangent vectors. Moreover, many ROMs generated by manifold and matrix interpolation are unstable, which means that the $\mathcal{H}_2$ errors are infinite. Although our method and subspace interpolation do not guarantee the stability of reduced models, because they seem to accurately approximate the reduced subspaces, unstable ROMs appear less often. We discuss issues specific to approximating IRKA bases in section SM7.

7.4. Anemometer: three-parameter case. To compare the methods in a PROM problem with multiple parameters, here we consider the three-parameter version of the anemometer [32]. The parameters include specific heat $c \in [0, 1]$, thermal conductivity $\kappa \in [1, 2]$, and fluid velocity $v \in [0.1, 2]$. The system matrices have the form $\mathbf{E} = \mathbf{E}_s + c\mathbf{E}_f$ and $\mathbf{A} = \mathbf{A}_{d,s} + \kappa\mathbf{A}_{d,f} + cv\mathbf{A}_c$, while $\mathbf{B}$ and $\mathbf{C}$ are constant. Other aspects of the problem are unchanged.

To sample the parameter space, we first use the maximin Latin hypercube sampling (LHS) to obtain a training set, and then use the sequential maximin design to obtain a testing set; see, e.g., [17, Chap. 4]. Maximin LHS generates a random set

Table 2
Mean relative $\mathcal{H}_2$ error for three-parameter anemometer, $k = 20$, varying sample size. Relative errors to local POD are shown in parentheses.

	$l = 14$		$l = 18$		$l = 21$	
local POD	5.55%	(1)*	5.46%	(1)	5.69%	(1)
GPS	6.49%	(1.169)	5.80%	(1.062)	5.14%	(0.903)
Subspace-Int	8.34%	(1.503)	7.38%	(1.352)	6.19%	(1.148)
Manifold-Int	16.6%	(2.986)	13.8%	(2.524)	12.7%	(2.232)
Matrix-Int	49.7%	(8.962)	44.2%	(8.104)	45.5%	(8.003)

Table 3
Mean relative $\mathcal{L}_2$ state error for three-parameter anemometer, $k = 20$, varying sample size. Relative errors to subspace interpolation are shown in parentheses.

	$l = 14$		$l = 18$		$l = 21$	
local POD	7.98e-13	(0)*	8.36e-13	(0)	8.77e-13	(0)
GPS	1.24e-2	(0.437)	6.42e-3	(0.273)	5.55e-3	(0.250)
Subspace-Int	2.85e-2	(1)	2.35e-2	(1)	2.22e-2	(1)

of points that are spread out in the parameter space and well-distanced from each other. Sequential maximin design generates another with similar properties, but also well-distanced from the given training set.

The setup for the PROM methods remain unchanged from the one-parameter case, except the interpolation scheme for the three interpolation methods. Since Lagrange and linear interpolations do not apply to multiple parameters, we use the radial basis function (RBF) method described in [2, p. 278]. Specifically, a multiquadric RBF is applied entrywise to interpolate the tangent vectors in subspace and manifold interpolation as well as the matrices in matrix interpolation. For subspace interpolation, horizontal projection is applied to maintain validity of the interpolated tangent vector.

Table 2 compares the mean relative $\mathcal{H}_2$ errors, with training sample sizes $l = 14, 18$, or 21, and testing sample size 100. For each training sample, GPS uses a lengthscale $\beta = 1.05, 0.85$, or 0.7, respectively. Notice that, with subspace dimension $k = 20$, the mean relative $\mathcal{H}_2$ error of local POD is about 5.5%, which is not particularly low. It is clear that our method is able to maintain the error level of local POD with as few as 18 training points. In comparison, the error increase in subspace interpolation is several times higher in all cases. Manifold interpolation is much less accurate than the previous two methods, while matrix interpolation is the least accurate.

Similarly, Table 3 compares the mean relative $\mathcal{L}_2$ state errors. Manifold and matrix interpolation are excluded because they cannot reconstruct the state vector. Since local POD minimizes the $\mathcal{L}_2$ state error by construction, its error level is practically zero. With $l = 14$, our method has a relative error of about 1%, less than half that of subspace interpolation. This ratio drops as sample size gets larger. Overall, the GPS method is much more data efficient than subspace interpolation in this multiparameter setting, again owing to its flexibility and intrinsic nature.

Measured computation time for this problem is provided in section SM5.

8. Concluding remarks.

In this paper we propose a new GP model for probabilistic approximation of subspace-valued functions. A key application of this model is parametric reduced order modeling. We show that the GPS model gives accurate predictions even with small sample sizes, and because its prediction cost does not depend on system dimension n, it is typically faster than subspace interpolation in PROM problems. In the following, we discuss several topics on the use of the GPS.

Prediction speed. Since the prediction cost of our method is cubic in subspace dimension k and sample size l, it is best to keep them small for fast computation. To keep k small, one needs to choose a ROM method that is best suited for the relevant error measure. For example, POD is optimal in $\mathcal{L}_2$ error of snapshot reconstruction, while IRKA is locally optimal in $\mathcal{H}_2$ error. To keep l small, one needs to choose an efficient method for parameter sampling. One may consider adaptive sampling and sparse grids [8], or experimental design methods in statistics [41, 17].

Handling higher-dimensional parameter spaces. When parameter dimension d is large, even with the $l = 10d$ rule of thumb for GP models [27], l can quickly become very large. Fortunately, there are some methods to cap the l^3 scaling. One approach is to use local approximate GP [18], where for each target point only a subsample of mostly nearby points are used in the prediction. Another approach is covariance tapering [15] or compactly supported kernels [24], where the kernel becomes zero beyond a certain distance, so that the covariance matrix is sparse and sparse matrix algorithms can be used to speed up computation. Both are in a similar spirit to parameter domain partitioning.

Prediction uncertainty. The uncertainty in subspace predictions, quantified by the eigenvalues of $\mathbf{\Sigma}$, serves as a diagnostic tool for prediction confidence. It can also guide parameter sampling: one can put extra sample points in regions with high prediction uncertainty. This could lead to efficient adaptive sampling methods [17].

Variation of subspace dimension. In some cases it can be desirable to let k vary with the parameters, e.g., to attain a fixed ROM accuracy. Since the Grassmann manifold requires a fixed k, it acts as a Procrustean bed and limits all methods based on it, including the GPS. We recommend setting k to the highest value in the sample.

Acknowledgments. The authors thank Johan Guilleminot, Akil Narayan, Daniel Tartakovsky, and the anonymous reviewers for helpful feedback during this work. We also thank the INFORMS QSR community for their recognition.

REFERENCES

[1] B. Afsari, *Riemannian L^p center of mass: Existence, uniqueness, and convexity*, Proc. Amer. Math. Soc., 139 (2011), pp. 655–673, https://doi.org/10.1090/S0002-9939-2010-10541-5.

[2] D. Amsallem, *Interpolation on Manifolds of CFD-Based Fluid and Finite Element-Based Structural Reduced-Order Models for On-Line Aeroelastic Predictions*, Ph.D. thesis, Stanford University, 2010, https://searchworks.stanford.edu/view/8652541.

[3] D. Amsallem and C. Farhat, *Interpolation method for adapting reduced-order models and application to aeroelasticity*, AIAA J., 46 (2008), pp. 1803–1813, https://doi.org/10.2514/1.35374.

[4] D. Amsallem and C. Farhat, *An online method for interpolating linear parametric reduced-order models*, SIAM J. Sci. Comput., 33 (2011), pp. 2169–2198, https://doi.org/10.1137/100813051.

[5] A. C. Antoulas, C. A. Beattie, and S. Gugercin, *Interpolatory Methods for Model Reduction*, SIAM, Philadelphia, 2020, https://doi.org/10.1137/1.9781611976083.

[6] H. Babaee, *An observation-driven time-dependent basis for a reduced description of transient stochastic systems*, Proc. A, 475 (2019), https://doi.org/10.1098/rspa.2019.0506.

[7] T. Bendokat, R. Zimmermann, and P. A. Absil, *A Grassmann Manifold Handbook: Basic Geometry and Computational Aspects*. https://arxiv.org/abs/2011.13699, 2020.

[8] P. Benner, S. Gugercin, and K. Willcox, *A survey of projection-based model reduction methods for parametric dynamical systems*, SIAM Rev., 57 (2015), pp. 483–531, https://doi.org/10.1137/130932715.

[9] S. Chaturantabut and D. C. Sorensen, *Nonlinear model reduction via discrete empirical interpolation*, SIAM J. Sci. Comput., 32 (2010), pp. 2737–2764, https://doi.org/10.1137/090766498.

[10] J. Chen, S. Mak, V. R. Joseph, and C. Zhang, *Function-on-function kriging, with applications to three-dimensional printing of aortic tissues*, Technometrics, 63 (2021), pp. 384–395.

[11] Y. Chikuse, *Statistics on Special Manifolds*, Springer-Verlag, New York, 2003, https://doi.org/10.1007/978-0-387-21540-2.

[12] P. G. Constantine, *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies*, SIAM, Philadelphia, PA, 2015, https://doi.org/10.1137/1.9781611973860.

[13] A. Edelman, T. A. Arias, and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 303–353, https://doi.org/10.1137/s0895479895290954.

[14] J. L. Eftang, A. T. Patera, and E. M. Rønquist, *An "hp" certified reduced basis method for parametrized elliptic partial differential equations*, SIAM J. Sci. Comput., 32 (2010), pp. 3170–3200, https://doi.org/10.1137/090780122.

[15] R. Furrer, M. G. Genton, and D. Nychka, *Covariance tapering for interpolation of large spatial datasets*, J. Comput. Graph. Statist., 15 (2006), pp. 502–523, https://doi.org/10.1198/106186006X132178.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 2013, https://jhupbooks.press.jhu.edu/title/matrix-computations.

[17] R. B. Gramacy, *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*, Chapman & Hall/CRC, Boca Raton, FL, 2020, https://bobby.gramacy.com/surrogates/.

[18] R. B. Gramacy and D. W. Apley, *Local Gaussian process approximation for large computer experiments*, J. Comput. Graph. Statist., 24 (2015), pp. 561–578, https://doi.org/10.1080/10618600.2014.914442.

[19] P. Grohs, *Quasi-interpolation in Riemannian manifolds*, IMA J. Numer. Anal., 33 (2013), pp. 849–874, https://doi.org/10.1093/imanum/drs026.

[20] S. Gugercin, A. C. Antoulas, and C. Beattie, *$\mathcal{H}_2$ model reduction for large-scale linear dynamical systems*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 609–638, https://doi.org/10.1137/060666123.

[21] R. Guhaniyogi and D. B. Dunson, *Compressed Gaussian process for manifold regression*, J. Mach. Learn. Res., 17 (2016), pp. 1–26, http://jmlr.org/papers/v17/14-230.html.

[22] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288, https://doi.org/10.1137/090771806.

[23] N. J. Higham, *The scaling and squaring method for the matrix exponential revisited*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 1179–1193, https://doi.org/10.1137/04061101x.

[24] C. G. Kaufman, D. Bingham, S. Habib, K. Heitmann, and J. A. Frieman, *Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology*, Ann. Appl. Stat., 5 (2011), pp. 2470–2492, https://doi.org/10.1214/11-AOAS489.

[25] L. Lin, N. Mu, P. Cheung, and D. Dunson, *Extrinsic Gaussian processes for regression and classification on manifolds*, Bayesian Anal., 14 (2019), pp. 887–906, https://doi.org/10.1214/18-BA1135.

[26] L. Lin, B. S. Thomas, H. Zhu, and D. B. Dunson, *Extrinsic local regression on manifold-valued data*, J. Amer. Statist. Assoc., 112 (2017), pp. 1261–1273, https://doi.org/10.1080/01621459.2016.1208615.

[27] J. L. Loeppky, J. Sacks, and W. J. Welch, *Choosing the sample size of a computer experiment: A practical guide*, Technometrics, 51 (2009), pp. 366–376, https://doi.org/10.1198/tech.2009.08040.

[28] J. L. Lumley, *The structure of inhomogeneous turbulent flows*, in Atmospheric Turbulence and Radio Wave Propagation, Elsevier, Amsterdam, 1967, pp. 166–178.

[29] S. Mak, C.-L. Sung, X. Wang, S.-T. Yeh, Y.-H. Chang, V. R. Joseph, V. Yang, and C. F. J. Wu, *An efficient surrogate model for emulation and physics extraction of large eddy simulations*, J. Amer. Statist. Assoc., 113 (2018), pp. 1443–1456, https://doi.org/10.1080/01621459.2017.1409123.

[30] A. Mallasto and A. Feragen, *Wrapped Gaussian process regression on Riemannian manifolds*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, https://openaccess.thecvf.com/content_cvpr_2018/html/Mallasto_Wrapped_Gaussian_Process_CVPR_2018_paper.html.

[31] B. C. Moore, *Principal component analysis in linear systems: Controllability, observability,*

and model reduction, IEEE Trans. Automat. Control, 26 (1981), pp. 17–32, https://doi.org/10.1109/TAC.1981.1102568.

[32] *Anemometer*, MORwiki Community, Model Order Reduction Wiki, 2018, http://modelreduction.org/index.php/Anemometer.

[33] M. Niu, P. Cheung, L. Lin, Z. Dai, N. Lawrence, and D. Dunson, *Intrinsic Gaussian processes on complex constrained domains*, J. R. Stat. Soc. Ser. B. Stat. Methodol., 81 (2019), pp. 603–627, https://doi.org/10.1111/rssb.12320.

[34] *Thermal model*, Oberwolfach Benchmark Collection, Model Order Reduction Wiki, 2018, http://modelreduction.org/index.php/Thermal_Model.

[35] M. Oulghelou, C. Allery, and R. Mosquera, *Parametric reduced order models based on a Riemannian barycentric interpolation*, Internat. J. Numer. Methods Engrg., 22 (2021), pp. 6623–6640, https://doi.org/10.1002/nme.6805.

[36] H. Panzer, J. Mohring, R. Eid, and B. Lohmann, *Parametric model order reduction by matrix interpolation*, Automatisierungstechnik, 58 (2010), pp. 475–484, https://doi.org/10.1524/auto.2010.0863.

[37] A. Petersen and H.-G. Müller, *Fréchet regression for random objects with Euclidean predictors*, Ann. Statist., 47 (2019), pp. 691–719, https://doi.org/10.1214/17-AOS1624.

[38] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA, 2006, http://www.gaussianprocess.org/gpml/.

[39] J. Saak, M. Köhler, and P. Benner, *M-M.E.S.S. - The Matrix Equation Sparse Solver Library*, Zenodo, 2021, https://doi.org/10.5281/zenodo.4719688.

[40] O. Sander, *Geodesic finite elements of higher order*, IMA J. Numer. Anal., 36 (2016), pp. 238–266, https://doi.org/10.1093/imanum/drv016.

[41] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*, Springer, New York, 2018, https://doi.org/10.1007/978-1-4939-8847-1.

[42] P. J. Schmid, *Dynamic mode decomposition of numerical and experimental data*, J. Fluid Mech., 656 (2010), pp. 5–28, https://doi.org/10.1017/s0022112010001217.

[43] N. T. Son, *A real time procedure for affinely dependent parametric model order reduction using interpolation on Grassmann manifolds*, Internat. J. Numer. Methods Engrg., 93 (2013), pp. 818–833, https://doi.org/10.1002/nme.4408.

[44] Y. Yang and D. B. Dunson, *Bayesian manifold regression*, Ann. Statist., 44 (2016), pp. 876–905, https://doi.org/10.1214/15-AOS1390.

[45] K. Ye and L.-H. Lim, *Schubert varieties and distances between subspaces of different dimensions*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 1176–1197, https://doi.org/10.1137/15m1054201.

[46] R. Zhang, *Newton Retraction as Approximate Geodesics on Submanifolds*. https://arxiv.org/abs/2006.14751, 2020.

[47] R. Zhang and R. Ghanem, *Normal-bundle bootstrap*, SIAM J. Math. Data Sci., 3 (2021), pp. 573–592, https://doi.org/10.1137/20m1356002.

[48] R. Zhang, P. Wingo, R. Duran, K. Rose, J. Bauer, and R. Ghanem, *Environmental economics and uncertainty: Review and a machine learning outlook*, in Oxford Research Encyclopedia of Environmental Science, Oxford University Press, 2020, https://doi.org/10.1093/acrefore/9780199389414.013.572.

[49] R. Zimmermann, *Manifold Interpolation and Model Reduction*. https://arxiv.org/abs/1902.06502, 2019.

[50] R. Zimmermann, *Hermite interpolation and data processing errors on Riemannian matrix manifolds*, SIAM J. Sci. Comput., 42 (2020), pp. A2593–A2619, https://doi.org/10.1137/19m1282878.