Chaos
An Interdisciplinary Journal of Nonlinear Science

RESEARCH ARTICLE | AUGUST 08 2022

Time-series forecasting using manifold learning, radial basis function interpolation, and geometric harmonics

Panagiotis G. Papaioannou; Ronen Talmon ; Ioannis G. Kevrekidis ; Constantinos Siettos

Chaos 32, 083113 (2022)
https://doi.org/10.1063/5.0094887
CHORUS

Time-series forecasting using manifold learning, radial basis function interpolation, and geometric harmonics

Cite as: Chaos **32**, 083113 (2022); doi: 10.1063/5.0094887
Submitted: 6 April 2022 · Accepted: 5 July 2022 ·
Published Online: 8 August 2022

Panagiotis G. Papaioannou,[1] Ronen Talmon,[2] Ioannis G. Kevrekidis,[3] and Constantinos Siettos[4,a)]

AFFILIATIONS

[1]Dipartimento di Matematica e Applicazioni "Renato Caccioppoli," Università degli Studi di Napoli Federico II, Naples 80126, Italy
[2]Viterbi Faculty of Electrical and Computer Engineering, Technion, Israel Institute of Technology, Haifa 3200003, Israel
[3]Department of Chemical and Biomolecular Engineering, Department of Applied Mathematics and Statistics, and the School of Medicine, Johns Hopkins University, Baltimore, Maryland 21218, USA
[4]Dipartimento di Matematica e Applicazioni "Renato Caccioppoli" and Scuola Superiore Meridionale, Università degli Studi di Napoli Federico II, Naples 80126, Italy

[a)]**Author to whom correspondence should be addressed:** constantinos.siettos@unina.it.
URL: http://wpage.unina.it/constantinos.siettos

ABSTRACT

We address a three-tier numerical framework based on nonlinear manifold learning for the forecasting of high-dimensional time series, relaxing the "curse of dimensionality" related to the training phase of surrogate/machine learning models. At the first step, we embed the high-dimensional time series into a reduced low-dimensional space using nonlinear manifold learning (local linear embedding and parsimonious diffusion maps). Then, we construct reduced-order surrogate models on the manifold (here, for our illustrations, we used multivariate autoregressive and Gaussian process regression models) to forecast the embedded dynamics. Finally, we solve the pre-image problem, thus lifting the embedded time series back to the original high-dimensional space using radial basis function interpolation and geometric harmonics. The proposed numerical data-driven scheme can also be applied as a reduced-order model procedure for the numerical solution/propagation of the (transient) dynamics of partial differential equations (PDEs). We assess the performance of the proposed scheme via three different families of problems: (a) the forecasting of synthetic time series generated by three simplistic linear and weakly nonlinear stochastic models resembling electroencephalography signals, (b) the prediction/propagation of the solution profiles of a linear parabolic PDE and the Brusselator model (a set of two nonlinear parabolic PDEs), and (c) the forecasting of a real-world data set containing daily time series of ten key foreign exchange rates spanning the time period 3 September 2001–29 October 2020.

Forecasting is of utmost importance in a wide spectrum of fields ranging from finance and macro-economics to geomechanics and earthquake forecasting and from ecology and environmental engineering to neuroscience and epidemiology, to name just a few. A key task is the construction of reduced-order surrogate models that are able to generalize well. However, for high-dimensional time series, the "curse of dimensionality" is a stumbling block toward this aim. Here, we address a numerical scheme to perform out-of-sample predictions by first embedding the high-dimensional time series into a low-dimensional manifold, then train reduced-order surrogate models, and finally solve the pre-image problem to reconstruct the predictions in the ambient space. The efficiency of the proposed approach is demonstrated using (a) synthetic stochastic time series resembling electroencephalography (EEG) signals, (b) linear and nonlinear parabolic PDEs, and (c) a real data set of a financial problem and, in particular, that of a foreign exchange rate (FOREX) market.

I. INTRODUCTION

Forecasting methods include exponential smoothing and moving average models, ARIMA, Bayesian nonparametric models including Gaussian processes and other machine learning schemes,

such as forward and recurrent neural networks, deep learning, LSTMs, reinforcement learning, reservoir computing, and fuzzy systems.[1–11] An inherent problem of using such data-driven modeling/forecasting approaches, when dealing with temporal measurements in a high-dimensional feature space, is the "curse of dimensionality."[12] In such cases, training, i.e., the estimation of the values of the model parameters, requires a large number of snapshots which increases exponentially with the dimension of the feature space. Thus, a fundamental task in modeling and forecasting is that of dimensionality reduction. Out of the many features that can be measured in time, one has to first identify the intrinsic dimension of the possible low-dimensional subspace (manifold) and the corresponding vectors that span it, which actually parameterize the emergent/macroscopically observed dynamics of the system under study. Assuming that the emergent dynamics evolve on a smooth enough low-dimensional manifold, an arsenal of linear, such as singular value decomposition (SVD) and dynamic mode decomposition (DMD),[13–15] and nonlinear, such as kernel-PCA,[16] locally linear embedding (LLE),[17] ISOMAP,[18,19] Laplacian eigenmaps,[20] diffusion maps (DMs),[21–25] and Koopman operator,[26–28] related manifold learning algorithms can be exploited toward this direction.

For the twofold task of dimensionality reduction and modeling in the low-dimensional subspace, Bollt[29] used an ISOMAP to approximate the manifold of dynamical systems in the form of ODEs and PDEs and to construct reduced-order models. Chiavazzo *et al.*[30] constructed reduced kinetics models by extracting the slow dynamics on a manifold globally parameterized by truncated DMs. A comparison of the reconstructed and the original high-dimensional dynamics was also reported. The reconstruction was achieved using radial basis functions (RBF) interpolation and geometric harmonics.[23] Liu *et al.*[31] used DMs to identify coarse variables that govern the emergent dynamics of a particle-based model of animal swarming and based on these, they constructed a reduced stochastic differential equation. Dsilva *et al.*[32] used DMs with a Mahalanobis distance to parameterize the slow manifold of multiscale dynamical systems. Williams *et al.*[27] addressed an extension of DMD to compute approximations of the leading eigenvalues, eigenfunctions, and modes of the Koopman operator, thus showing that for large data sets, the procedure converges to the numerical approximation one would obtain from a Galerkin method. The performance of the method was tested via the unforced Duffing equation and a stochastic differential equation with a double-well potential that converges to a Fokker–Planck equation. Brunton *et al.*[33] used SVD to embed the dynamics of nonlinear systems in a linear manifold spanned by a few modes and constructed state-space linear models on the manifold to approximate the full dynamics. The so-called sparse identification of the nonlinear dynamics (SINDy) method was demonstrated using the Lorenz equations and the 2D Navier–Stokes equations. Bhattacharjee and Matouš[34] used ISOMAP to construct reduced-order models of heterogeneous hyperelastic materials in the 3D space, whose solutions are obtained by finite elements, while for the construction of the inverse map, they used RBF interpolation. Wand and Sapsis[35] proposed a methodology for forecasting and quantifying uncertainty in a reduced-dimensionality space based on Gaussian process regression. The efficiency of the proposed approach was validated using data series from the Lorenz 96 model, the Kuramoto–Sivashinsky, as well as a barotropic climate model in the form of a PDE. Kutz *et al.*[36] proposed a scheme based on the Koopman-operator theory to embed spatiotemporal dynamics of PDEs with the aid of DMD for approximating the evolution of the high-dimensional dynamics on the reduced manifold; the reconstruction of the states of the original system was achieved by a linear transformation. Chen and Ferguson[37] have proposed a molecular enhanced sampling method based on autoencoders[38] to discover the low-dimensional projection of molecular dynamics and then reconstructed back the atomic coordinates. Vlachas *et al.*[39] proposed an LSTM-based method to predict the high-dimensional dynamics from the information acquired in a low-dimensional space. The embedding space is constructed based on the Takens embedding theorem.[40] The method is demonstrated through time series obtained from the Lorenz 96 equations, the Kuramoto–Sivashinsky equation, and a barotropic climate model as in Wand and Sapsis.[35] Herzog *et al.*[41] used a convolutional autoencoder and extended conditional random fields for dimensionality reduction and prediction, respectively, for dealing with chaotic spatiotemporal series. The method was demonstrated via the Lorenz-96 system and the Kuramoto–Sivashinsky PDEs. Lee *et al.*[42] used DMs and automatic relevance determination to construct embedded PDEs by the aid of artificial neural networks (ANNs) and Gaussian processes. The methodology was illustrated with the lattice-Boltzmann simulations of the 1D FitzHugh–Nagumo model. Koronaki *et al.*[43] used DMs to embed the dynamics of a one-dimensional tubular reactor as modeled by a system of two PDEs on a 2D manifold and then constructed a feedforward ANN to learn the dynamics on the 2D manifold. The efficiency of the scheme was compared with the original high-dimensional PDE dynamics. Isensee *et al.*[44] used PCA for dimensionality reduction of spatiotemporal time series based on delay embedding nearest neighbor methods in the low-dimensinal space for prediction. The approach was demonstrated through noisy data produced from a Barkley model the Bueno–Orovio–Cherry–Fenton and the Kuramoto–Sivashinsky model. Recently, Lin and Lu[45] used the Koopman and the Mori–Zwanzig projection operator formalism to construct reduced-order models for chaotic and randomly forced dynamical systems, and then based on the Wiener projection, they derived nonlinear auto-regressive moving average with exogenous input models. The approach was demonstrated through the Kuramoto–Sivashinsky PDE and the viscous Burgers equation with stochastic forcing.

The performance of most of the above schemes was assessed in terms of interpolation; namely, the data were produced using dynamical models (ODEs, PDEs, SDEs, or kinetic models) that were simulated in some specific interval of the high-dimensional domain, then reduced-order models were constructed and trained based on the generated data, and finally, a reconstruction of the high-dimensional dynamics was implemented within the original interval of the high-dimensional domain.

In this work, we address a three-tier scheme to perform forecasting, i.e., out-of-sample extrapolation of high-dimensional time series by (i) embedding the high-dimensional time series into a low-dimensional manifold using nonlinear manifold learning, (ii) constructing and training reduced-order surrogate models on the low-dimensional manifold and, based on these, make predictions,

and finally, (iii) solve the pre-image problem by lifting the predictions in the high-dimensional space with geometric harmonics[23] or radial basis function (RBF) interpolation. The performance of this "embed–forecast–lift" scheme is assessed through three different families of problems: (a) the forecasting of three simplistic synthetic time series resembling EEG recordings generated by linear and weakly nonlinear discrete stochastic models, (b) the prediction/propagation of the solutions of two parabolic PDEs (a linear one and the Brusselator model giving rise to sustained oscillations) based on past spatiotemporal data, and (c) the forecasting of a real-world data set containing the time series of 10 key pairs of foreign exchange prices retrieved from the www.investing.com open API, spanning the period 3 September 2001–29 October 2020. For our computations, we used two well-established nonlinear manifold learning algorithms, namely, the LLE algorithm and DMs, two types of surrogate models, namely multivariate autoregressive (MVAR) and Gaussian process regression (GPR) models, and two methods for solving the pre-image problem, namely, geometric harmonics and radial basis function interpolation. We considered various combinations of the previous methods and compared them against the naïve random walk and MVAR and GPR surrogate models implemented directly in the original space. A comparison with the results obtained with PCA for the cases of the numerical solution of the parabolic PDEs and the FOREX forecasting problem is also provided. To the best of our knowledge, this is the first work that addresses such a numerical framework for the solution of the problem of forecasting of high-dimensional time series based on nonlinear manifold learning, thus providing a comparison of various embedding, modeling, and reconstruction approaches and assessing their performance on synthetic time series, PDEs, and a real-world FOREX data set.

II. METHODOLOGY AND PRELIMINARIES

Let us denote by $\boldsymbol{x}_i \in \mathbb{R}^D$ the vector containing the features at the ith snapshot of the time series and by $\boldsymbol{X} \in \mathbb{R}^{D\times N}$ the matrix having as columns the vectors spanning a time window of N observations. The assumption is that the time series/dynamics lie on a "smooth enough" low-dimensional (say, of dimension d) manifold that is embedded in the high-dimensional $\mathbb{R}^D$ space. Our aim is to exploit manifold learning algorithms to forecast the time series in the high-dimensional feature space. Thus, our purpose is to bypass the "curse of dimensionality" associated with the training of surrogate models in the original input space and, with the limited size that usually characterizes real-world data (such as financial data). Toward this aim, we propose a numerical framework that consists of three steps.

At the first step, we employ embedding algorithms (LLE and DMs) to construct nonlinear maps from the high-dimensional input space to a low-dimensional subspace that preserve as much as possible the intrinsic geometry of the manifold (i.e., maps assuring that neighborhoods of points in the high-dimensional space are mapped to neighborhoods of the same points in the low-dimensional manifold, and the notion of distance between the points is maintained as much as possible). In general, it is expected that the manifold learning algorithms will provide an approximation to the intrinsic embedding dimension, which will differ from the "true" one. This depends on the threshold that one puts for the selection of the number of embedded vectors that span the low-dimensional manifold. A central concept here is the Riemannian metric, which defines the properties of this map. At the second step, based on the resulting embedding features that span the low-dimensional manifold, we train a class of surrogate models (MVAR and GPR) to predict the evolution of the embedded time series on the manifold based on their past values. The final step implements the solution of the pre-image problem, i.e., a construction of a lifting operator. The aim here is to form the inverse map from the out-of-sample forecasted embedded points to the reconstruction of the features in the high-dimensional input space. On one hand, the existence of such an inverse map is theoretically guaranteed by the assumption that a low-dimensional "sufficiently smooth" manifold exists. At this point, we should note that the embedding generated using noisy data may produce a distorted and, therefore, unreliable geometry (see, e.g., the discussion in Gajamannage *et al.*[46]). On the other hand, the data-driven derivation of the corresponding inverse map is neither unique[47] nor trivial as, for example, when using PCA. In general, one generally has to solve a nonlinear least-squares problem requiring the minimization of an objective function that can be formed using different criteria for the properties of the neighborhood of points; this results in different inverse maps whose performance has to be validated numerically.

Next, and for the completeness and clarity of the presentation of the methodology, we briefly describe basic concepts of the manifold theory.

Manifold learning techniques can be viewed as unsupervised machine learning algorithms in the sense that they "learn" from the available data a low-dimensional representation of the original high-dimensional input space, thus providing an "optimal" (under certain assumptions) embedded subspace where the information available in the high-dimensional space is preserved as much as possible. Here, we briefly present some basic elements of the theory of manifolds and manifold learning.[47–50] Let us start with the definition of a d-dimensional manifold.

Definition 1 (Manifold): *A set $M \subset \mathbb{R}^n$ is called a d-dimensional manifold (of class C^∞) if for each point $\boldsymbol{p} \in M$, there is an open set $W \subset M$ such that there exists a C^∞ bijection $\boldsymbol{f}: U \to W$, $U \subset \mathbb{R}^d$ open, with C^∞ inverse $\boldsymbol{f}^{-1}: W \to U$ (i.e., W is C^∞-diffeomorphic to U). The bijection $\boldsymbol{f}$ is called a local parameterization of M, while $\boldsymbol{f}^{-1}$ is called a local coordinate mapping, and the pair $(W,\boldsymbol{f}^{-1})$ is called a chart (or neighborhood) of $\boldsymbol{p}$ on M.*

Thus, in a coordinate system $(W,\boldsymbol{f}^{-1})$, a point $\boldsymbol{p} \in W$ can be expressed by the coordinates $(f_1^{-1}, f_2^{-1}, \ldots, f_n^{-1})$, where f_i^{-1} is the ith element of $\boldsymbol{f}^{-1}$. A chart $(W,\boldsymbol{f}^{-1})$ is centered at $\boldsymbol{p}$ if and only if $\boldsymbol{f}^{-1}(\boldsymbol{p}) = \boldsymbol{0}$. Thus, we always assume that the manifold satisfies the Hausdorff separation axiom, stating that every pair of distinct points on M has disjoint open neighborhoods. Summarizing, a manifold is a Hausdorff (Separated/T_2) space, in which every point has a neighborhood that is diffeomorphic to an open subset in $\mathbb{R}^d$.

Definition 2 (Tangent space and tangent bundle to a manifold): *Let $M \subset \mathbb{R}^n$ be a manifold and $\boldsymbol{p} \in M$. The tangent space to M at $\boldsymbol{p}$ is the vector subspace $T_{\boldsymbol{p}}M$ formed by the tangent vectors at $\boldsymbol{p}$ defined by*

$$T_{\boldsymbol{p}}M = D\boldsymbol{f}_{\boldsymbol{u}}(T_{\boldsymbol{u}}\mathbb{R}^d), \quad \boldsymbol{f}(\boldsymbol{u}) = \boldsymbol{p},$$

where $T_{\boldsymbol{u}}\mathbb{R}^d$ is the tangent space at $\boldsymbol{u}$ and

$$Df_{\boldsymbol{u}}(T_{\boldsymbol{u}}\mathbb{R}^d) = \left[\frac{\partial f_i(\boldsymbol{u})}{\partial u_j}\right] = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots & \frac{\partial f_1}{\partial u_d} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \cdots & \frac{\partial f_2}{\partial u_d} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \frac{\partial f_n}{\partial u_2} & \cdots & \frac{\partial f_n}{\partial u_d} \end{bmatrix},$$

with $\text{rank}(Df) = d$. Thus, $S = \left\{\frac{\partial f}{\partial u_1}, \frac{\partial f}{\partial u_2}, \ldots, \frac{\partial f}{\partial u_d}\right\}$ is a basis for $T_{\boldsymbol{p}}M$. The union of tangent spaces

$$TM = \cup_{\boldsymbol{p}\in M} T_{\boldsymbol{p}}M$$

is called the tangent bundle of M.

Definition 3 (Riemannian manifold and metric): *A Riemannian manifold is a manifold M endowed with a positive definite inner product $g_{\boldsymbol{p}}$ defined on the tangent space $T_{\boldsymbol{p}}M$ at each point $\boldsymbol{p}$. The family of inner products $g_{\boldsymbol{p}}$ is called a Riemannian metric, and the Riemannian manifold is denoted by (M, g).*

The above definitions refer to the case of continuum limit—infinite number of points. However, for real data with a limited number of observations, one can only try to approximate the manifold; the analytic charts of the Riemannian metric that actually define the geometry of the manifold are simply not available. Thus, a consistent manifold learning algorithm constructs in a numerical way a map that approximates in a probabilistic way the continuous one when the sample size $N \to \infty$. Thus, a fundamental preprocessing step of a class of manifold learning algorithms, such as Laplacian maps, LLE, ISOMAP, and DMs, is the construction of a weighted graph, say, $\mathcal{G}(\boldsymbol{X}, E)$, where E denotes the set of weights between points. This construction is based on a predefined metric (such as the Gaussian kernel and the k-NN algorithm), which is used to appropriately connect each point $\boldsymbol{x} \in \mathbb{R}^D$ with all the others. This defines a weighted graph of neighborhoods of points. Theoretically, with an appropriate choice of the metric and its parameters, the graph is guaranteed to be connected; i.e., there is a path between every pair of points in $\boldsymbol{X}$.[47]

For the completeness of the presentation, in Secs. III–V, we briefly discuss the manifold learning algorithms, the regression models, and the lifting techniques used in this work.

III. MANIFOLD LEARNING ALGORITHMS

A. Locally linear embedding (LLE)

The LLE algorithm[51] is a nonlinear manifold learning algorithm, which constructs $\mathcal{G}(\boldsymbol{X}, E)$ based on the k-NN algorithm. It is assumed that the neighborhood forms a basis for the reconstruction of any point in the neighborhood itself. Thus, every point is written as a linear combination of its neighbors. The weights of all pairs are then estimated with least squares, minimizing the L_2 norm of the difference between the points and their neighborhood-based reconstruction. The same rationale is assumed for the low-dimensional subspace, keeping the same estimates of the weights in the high-dimensional space. In the low-dimensional subspace, one now seeks for the coordinates of the points that minimize the L_2 norm of the difference between the coordinate of the points on a d-dimensional manifold and the weighted neighborhood reconstruction. The minimization problem is represented as an eigenvalue problem, whose first d bottom non-zero eigenvectors provide an ordered set of orthogonal coordinates that span the manifold. It should be noted that in order to obtain a unique solution to the minimization problem, the number of k nearest neighbors should not exceed the dimension of the input space.[51]

The LLE algorithm can be summarized in the following three basic steps.[51]

1. Identify the k nearest neighbors for all $\boldsymbol{x}_i \in \mathbb{R}^D$, $i = 1, 2, \ldots, N$ (with $k \geq d + 1$). For each $\boldsymbol{x}_i$, this forms a set $K\{\boldsymbol{x}_i\} \subset \mathbb{R}^{k\times D}$ containing the k nearest neighbors of $\boldsymbol{x}_i$.
2. Write each $\boldsymbol{x}_i$ in terms of $K\{\boldsymbol{x}_i\}$ as

$$\boldsymbol{x}_i = \sum_{j\in K\{\boldsymbol{x}_i\}} w_{ij}\boldsymbol{x}_j \tag{1}$$

and find the matrix $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{N\times N}$ of the unknown weights by minimizing the objective function

$$\mathscr{L}(\boldsymbol{W}) = \sum_{i=1}^{N} \left\| \boldsymbol{x}_i - \sum_{j=1, j\neq i}^{N} w_{ij}\boldsymbol{x}_j \right\|_{L_2}^2, \tag{2}$$

with the constraint

$$\sum_{j=1}^{N} w_{ij} = 1. \tag{3}$$

3. Embed the points $\boldsymbol{x}_i \in \mathbb{R}^D$, $i = 1, 2, \ldots, N$, into a low-dimensional space with coordinates $\boldsymbol{y}_i \in \mathbb{R}^d$, $i = 1, 2, \ldots, N$, $d \ll D$. This step in LLE is accomplished by computing the vectors $\boldsymbol{y}_i \in \mathbb{R}^d$ that minimize the objective function,

$$\phi(\boldsymbol{Y}) = \sum_{i=1}^{N} \left\| \boldsymbol{y}_i - \sum_{j=1, j\neq i}^{N} w_{ij}\boldsymbol{y}_j \right\|_{L_2}^2, \tag{4}$$

where the weights w_{ij} are fixed at the values found by solving the minimization problem (2). The embedding vectors are required to be centered at the origin with an identity covariance matrix.[51] The embedded vectors $\boldsymbol{y}_i$ are constrained so that they have a zero mean and a unit covariance matrix.

The cost function (4) is quadratic and can be stated as

$$\phi(\boldsymbol{Y}) = \sum_{i=1}^{N}\sum_{j=1}^{N} q_{ij}\langle \boldsymbol{y}_i, \boldsymbol{y}_j\rangle, \tag{5}$$

involving the inner products of the embedding vectors and a symmetric square matrix $\boldsymbol{Q}$ with elements

$$q_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_k w_{ki}w_{kj}, \tag{6}$$

$\delta_{ij} = 1$ if $i = j$ and 0 otherwise. In a matrix form, $\boldsymbol{Q} \in \mathbb{R}^{N\times N}$ can be written as

$$\boldsymbol{Q} = (\boldsymbol{I} - \boldsymbol{W})^T(\boldsymbol{I} - \boldsymbol{W}). \tag{7}$$

In practice, this representation of $\boldsymbol{Q}$ gives rise to a significant reduction of the computational cost, especially when N is large, since left

multiplication by $\boldsymbol{Q}$ can be performed as

$$\boldsymbol{Q}\boldsymbol{v} = (\boldsymbol{v} - \boldsymbol{W}\boldsymbol{v}) - \boldsymbol{W}^T(\boldsymbol{v} - \boldsymbol{W}\boldsymbol{v}), \quad \boldsymbol{v} \in \mathbb{R}^N, \tag{8}$$

requiring just two multiplications by the sparse matrices $\boldsymbol{W}$ and $\boldsymbol{W}^T$.

Minimizing the cost function (4) can be performed via the eigenvalue decomposition of $\boldsymbol{Q}$. Specifically, the eigenvector associated with the smallest (zero) eigenvalue is the vector with all 1s, and it trivially minimizes (4);[51] it is disregarded because it leads to a constant (degenerate) embedding. The optimal embedding is, therefore, obtained by the d eigenvectors of $\boldsymbol{Q}$, denoted as $\boldsymbol{q}_k \in \mathbb{R}^N$, $k = 1, 2, \ldots, d$, corresponding to the next d smallest eigenvalues. Thus, the coordinates of $\boldsymbol{x}_i$, $i = 1, 2, \ldots, N$, in the embedded space are given by the vector

$$\mathscr{R}(\boldsymbol{x}_i) = \boldsymbol{y}_i = [q_{i1}, q_{i2}, \ldots, q_{id}]^T, \tag{9}$$

where q_{ij} denotes the jth element of eigenvector $\boldsymbol{q}_i$.

B. Diffusion maps (DMs)

Here, the construction of the affinity matrix is based on the computation of a random walk on the graph $\mathscr{G}(\boldsymbol{X}, E)$. The first step is to construct a graph using a kernel function, say, $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The kernel function can be chosen as a Riemannian metric so that it is symmetric and positive definite. Standard kernels, such as the Gaussian kernel, typically define a neighborhood of each point $\boldsymbol{x}_i$, i.e., a set of points $\boldsymbol{x}_j$ having strong connections with $\boldsymbol{x}_i$, namely, large values of $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

At the next step, one constructs a Markovian transition matrix, say, $\boldsymbol{P}$, whose elements correspond to the probability of jumping from one point to another in the high-dimensional space. This transition matrix defines a Markovian (i.e., memoryless) random walk X_t by setting

$$p_{ij} = p(\boldsymbol{x}_i, \boldsymbol{x}_j) = \text{Prob}(X_{t+1} = \boldsymbol{x}_j | X_t = \boldsymbol{x}_i).$$

For a graph constructed from a sample of finite size N, the weighted degree of a point (node) is defined by

$$\deg(\boldsymbol{x}_i) = \sum_{j=1}^{N} k(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{10}$$

and the volume of the graph is given by

$$vol(\mathscr{G}) = \sum_{i=1}^{N} d(\boldsymbol{x}_i).$$

Then, the random walk on such a weighted graph can be defined by the transition probabilities

$$p_{ij} = p(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\deg(\boldsymbol{x}_i)}. \tag{11}$$

Clearly, from the above definition, we have that $\sum_{j=1}^{N} p(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1$.

We note that in the continuum, the above can be described as a continuous Markov process on a probability space $(\Omega, \mathscr{H}, \mathscr{P})$, where Ω is the sample space, $\mathscr{H}$ is a σ-algebra of events in Ω, and $\mathscr{P}$ is a probability measure. Let μ be the density function of the points in the sample space Ω induced from the probability measure, $\mu : \mathscr{H} \to \mathbb{R}$ with $\mu(\Omega) = 1$. Then, using the kernel function k, the transition probability from a point $\boldsymbol{x} \in \Omega$ to another point $\boldsymbol{y} \in \Omega$ is given by

$$p(\boldsymbol{x}, \boldsymbol{y}) = \frac{k(\boldsymbol{x}, \boldsymbol{y})}{\int_{\Omega} k(\boldsymbol{x}, \boldsymbol{y}) d\mu(\boldsymbol{\Omega})}, \tag{12}$$

which is the continuous-space counterpart of (11).

The above procedure defines a row-stochastic transition matrix, $\boldsymbol{P} = [p_{ij}]$, which encapsulates the information about the neighborhoods of the points. Note that by raising $\boldsymbol{P}$ to the power of $t = 1, 2, \ldots$, we get the jumping probabilities in t steps. This way, the underlying geometry is revealed through high or low transition probabilities between the points, i.e., paths that follow the underlying geometry have a high probability of occurrence, while paths away from the "true" embedded structure have a low probability. Note that the t-step transition probabilities, say, $p_t(\boldsymbol{x}_i, \boldsymbol{x}_j)$, satisfy the Chapman–Kolmogorov equation,

$$p_{t_1+t_2}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{\boldsymbol{x}_k \in \boldsymbol{X}} p_{t_1}(\boldsymbol{x}_i, \boldsymbol{x}_k) p_{t_2}(\boldsymbol{x}_k, \boldsymbol{x}_j). \tag{13}$$

The Markov process defined by the probability matrix $\boldsymbol{P}$ has a stationary distribution given by

$$\pi(\boldsymbol{x}_i) = \frac{\deg(\boldsymbol{x}_i)}{\sum_{\boldsymbol{x}_j \in \boldsymbol{X}} \deg(\boldsymbol{x}_j)}, \tag{14}$$

and it is reversible, i.e.,

$$\pi(\boldsymbol{x}_i) p(\boldsymbol{x}_i, \boldsymbol{x}_j) = \pi(\boldsymbol{x}_j) p(\boldsymbol{x}_j, \boldsymbol{x}_i), \quad \forall \boldsymbol{x}_i, \boldsymbol{x}_i \in \boldsymbol{X}. \tag{15}$$

Furthermore, if the kernel is appropriately chosen so that the graph is connected, then the Markov chain is ergodic and irreducible, and its transition matrix has a stationary vector $\boldsymbol{\pi}$ such that

$$\boldsymbol{P}^T\boldsymbol{\pi} = \boldsymbol{K}\boldsymbol{D}^{-1}\boldsymbol{\pi} = \frac{\boldsymbol{K}\boldsymbol{D}^{-1}\boldsymbol{d}}{\mathbf{1}^T\boldsymbol{d}} = \frac{\boldsymbol{K}\mathbf{1}}{\mathbf{1}^T\boldsymbol{d}} = \frac{\boldsymbol{d}}{\mathbf{1}^T\boldsymbol{d}} = \boldsymbol{\pi}, \tag{16}$$

where $\boldsymbol{d}$ is the degree vector whose ith entry is $\deg(\boldsymbol{x}_i)$ and $\mathbf{1}$ is an all ones vector.

From the Perron–Frobenius theorem, we know that its geometric multiplicity is 1. It can be shown that all other eigenvalues have a magnitude smaller than 1.[22] From (11), we get

$$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{K}, \quad \boldsymbol{D} = \text{diag}\left(\sum_{j=1}^{N} k_{ij}\right), \tag{17}$$

where $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. We note that the transition matrix $\boldsymbol{P}$ is similar to the symmetric positive definite matrix $\boldsymbol{S} = \boldsymbol{D}^{-1/2}\boldsymbol{K}\boldsymbol{D}^{-1/2}$. Thus, the transition matrix $\boldsymbol{P}$ has a decomposition given by

$$\boldsymbol{P} = \sum_{i=1}^{N} \lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^T, \tag{18}$$

where $\lambda_i \in \mathbb{R}$ are the (positive) eigenvalues of $\boldsymbol{P}$, $\boldsymbol{u}_i \in \mathbb{R}^N$ are the left eigenvectors, and $\boldsymbol{v}_i \in \mathbb{R}^N$ are the right eigenvectors such that $\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle = \delta_{ij}$.

The set of right eigenvectors $\boldsymbol{v}_i$ establishes an orthonormal basis for the subspace $R(\boldsymbol{P}^T)$ of $\mathbb{R}^D$ spanned by the rows of $\boldsymbol{P}$. Row i represents the transition probabilities from point $\boldsymbol{x}_i$ to all the other points

of the graph. According to the Eckart–Young–Mirsky theorem,[52,53] the best d-dimensional low-rank approximation of the row space of $\boldsymbol{P}$ in the Euclidean space $\mathbb{R}^d$ is provided by its d right eigenvectors corresponding to the d largest eigenvalues.

It has been shown that asymptotically, when the number of data points uniformly sampled from a low dimensional manifold goes to infinity, the matrix $\frac{1}{\sigma}(\boldsymbol{I}-\boldsymbol{P})$ (where σ is the kernel function scale) approaches the Laplace–Beltrami operator of the underlying Riemannian manifold.[21,24] This allows one to consider the eigenvectors corresponding to the largest few eigenvalues of $\boldsymbol{P}$ as discrete approximations of the principal eigenfunctions of the Laplace–Beltrami operator. Since the principal eigenfunctions of the Laplace–Beltrami operator establish an accurate embedding of the manifold,[54] this result promotes the usage of the eigenvectors of $\boldsymbol{P}$ for practical (discrete) data embedding.

At this point, the so-called diffusion distance, which is an affinity distance related to the reachability between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, is given by

$$D_t^2(\boldsymbol{x}_i,\boldsymbol{x}_j) = ||p_t(\boldsymbol{x}_i,\cdot) - p_t(\boldsymbol{x}_j,\cdot)||^2_{L_{2,1/\deg}} \tag{19}$$

$$= \sum_{k=1}^{N} \frac{(p_t(\boldsymbol{x}_i,\boldsymbol{x}_k) - p_t(\boldsymbol{x}_j,\boldsymbol{x}_k))^2}{\deg(\boldsymbol{x}_k)}, \tag{20}$$

where $p_t(\boldsymbol{x}_i,\cdot)$ is the ith row of $\boldsymbol{P}^t$. The embedding of the t-step transition probabilities is achieved by forming a family of maps (DMs) of the N points $\boldsymbol{x}_i \in \boldsymbol{X}$ in the Euclidean subspace of $\mathbb{R}^d$ defined by

$$\mathscr{R}_t(\boldsymbol{x}_i) = \boldsymbol{y}_i = \left[\lambda_1^t \boldsymbol{v}_1(i), \lambda_2^t \boldsymbol{v}_2(i), \ldots, \lambda_d^t \boldsymbol{v}_d(i)\right]^T, i = 1,2,\ldots,N. \tag{21}$$

A useful property of DMs is that the Euclidean distance in the embedded space $||\mathscr{R}_t(\boldsymbol{x}_i) - \mathscr{R}_t(\boldsymbol{x}_j)||_{L_2}$ is the best d-dimensional approximation of the diffusion distance, given by (20), where the equality holds for $d = N$.

IV. FORECASTING WITH REGRESSION MODELS

Let us assume that the time series are being generated by a (nonlinear) law of the form

$$\boldsymbol{y}_t = \boldsymbol{\phi}(\boldsymbol{z},\boldsymbol{\mu}) + \boldsymbol{e}_t, \tag{22}$$

where $\boldsymbol{y}_t \in \mathbb{R}^d$ denotes the vector containing the measured values of the response variables at time t, $\boldsymbol{\phi} : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^d$ is a function encapsulating the law that governs the system dynamics, which contains the parameters and exogenous variables represented by the vector $\mu \in \mathbb{R}^q$, $\boldsymbol{z} \in \mathbb{R}^p$ is the vector containing the explanatory input variables (predictors), which may include past values of the response variables at certain times, say, $t-1, t-2, \ldots, t-m$, and $\boldsymbol{e}_t$ is the vector of the unobserved noise at time t.

In general, the forecasting problem (here in the low-dimensional space) using regression models can be written as a minimization problem of the following form:

$$\underset{\boldsymbol{g}\in G,\,\boldsymbol{\theta}\in\mathbb{R}^l}{\operatorname{argmin}} \mathscr{L}(\boldsymbol{y}_{t+k} - \boldsymbol{g}(\boldsymbol{z},\boldsymbol{\theta})), \tag{23}$$

where k is the prediction horizon, $\boldsymbol{g} : \mathbb{R}^p \times \mathbb{R}^l \to \mathbb{R}^d$ is the regression model,

$$\hat{\boldsymbol{y}}_{t+k} = \boldsymbol{g}(\boldsymbol{z},\boldsymbol{\theta}), \tag{24}$$

with parameters $\boldsymbol{\theta} \in \mathbb{R}^l$, G is the space of available models $\boldsymbol{g}$, $\hat{\boldsymbol{y}}_t$ is the output of the model at time t, and $\mathscr{L}$ is the loss function that determines the optimal forecast.

We note that the forecasting problem can be posed in two different modes:[55] the iterative and the direct one. In the iterative mode, one trains one-step ahead surrogate models based on the available time series and simulates/iterates the models to predict future values. In the direct mode, forecasting is performed in a sliding-window framework, where the model is retrained with the data contained within the sliding window to provide a multiperiod-ahead value of the dependent variables. Here, we aim at testing the performance of the proposed scheme for one-period ahead of time predictions (i.e., with $k = 1$) with both iterative and direct modes. For our illustrations, we used two surrogate models, namely, MVAR and GPR. A brief description of the models follows.

A. Multivariate autoregressive (MVAR) model

An MVAR model can then be written as

$$\boldsymbol{y}_t = \boldsymbol{\theta_0} + \sum_{j=1}^{m} \boldsymbol{y}_{t-j}\boldsymbol{\Theta}_j + \boldsymbol{e}_t, \tag{25}$$

where $\boldsymbol{y}_t = [y_{t1}, y_{t2}, \ldots, y_{td}]^T \in \mathbb{R}^d$ is the vector of the response time series at time t, m is the model order, i.e., the maximum time lag, $\boldsymbol{\theta_0} \in \mathbb{R}^d$ is the regression intercept, $\boldsymbol{\Theta}_j \in \mathbb{R}^{d\times d}$ is the matrix containing the regression coefficients of the MVAR model, and $\boldsymbol{e}_t = [e_t^{(1)}, e_t^{(2)}, \ldots, e_t^{(d)}]^T$ is the vector of the unobserved errors at time t, which are assumed to be uncorrelated random variables with zero mean and constant variance σ^2. In a more general form, in view of (22), the MVAR model can be written as

$$y_{ik} = \theta_{0k} + \sum_{j=1}^{m} \theta_{jk} z_{ij} + e_{ik}, \quad i = 1,2,\ldots,d, \quad k = 1,2,\ldots,N, \tag{26}$$

where y_{ik} is the model output for the ith variable at the kth time instant, $\theta_{0k} \in \mathbb{R}$ is the corresponding regression intercept and θ_{jk} the corresponding jth regression coefficient, and z_{ij} is the jth predictor of the ith response variable (e.g., the time-delayed time series). According to the Gauss–Markov theorem, the best unbiased linear estimator of the regression coefficients is the one that results from the solution of the least-squares (LS) problem,

$$\arg\min_{\theta_{0k},\theta_{jk}} \sum_{i=1}^{d}\sum_{k=1}^{N} \left(y_{ik} - \theta_{0k} - \sum_{j=1}^{m} \theta_{jk} z_{ij} - e_{ik} \right)^2,$$

given by

$$\hat{\boldsymbol{\Theta}} = (\boldsymbol{Z}^T\boldsymbol{Z})^{-1}\boldsymbol{Z}^T\boldsymbol{Y}, \tag{27}$$

where $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_d] \in \mathbb{R}^{N\times d}$ and $\boldsymbol{Z} = [\boldsymbol{1}_N, \boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m] \in \mathbb{R}^{N\times(m+1)}$. Assuming that the unobserved errors are i.i.d. normal random variables and then by the maximum likelihood estimate of the error covariance, one can also estimate the forecasting intervals of a new observation.

B. Gaussian process regression (GPR)

An introduction to the use of Gaussian processes for time-series forecasting can be found in Refs. 5 and 56. For the implementation of GPR, it is assumed that the unknown function $\boldsymbol{\phi}$ in (22) can be modeled by d single-output Gaussian distributions given by

$$P(\boldsymbol{\phi}_i|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{\phi}_i|\boldsymbol{\mu}, \boldsymbol{K}(\boldsymbol{z},\boldsymbol{z}|\boldsymbol{\theta})), \tag{28}$$

where $\boldsymbol{\phi}_i = [\phi_i(\boldsymbol{z}_1), \phi_i(\boldsymbol{z}_2), \ldots, \phi_i(\boldsymbol{z}_N)]$, ϕ_i is the ith component of $\boldsymbol{\phi}$, $\boldsymbol{\mu}(\boldsymbol{z})$ is the vector with the expected values of the function, and $\boldsymbol{K}(\boldsymbol{z},\boldsymbol{\theta})$ is a $N \times N$ covariance matrix formed by a kernel. The prior mean function is often set to $\boldsymbol{\mu}(\boldsymbol{z}) = \boldsymbol{0}$ with appropriate normalization of the data.

Predictions at a new point, say, $\boldsymbol{z}_*$, are made by drawing $\boldsymbol{\phi}_{i*}$ from the posterior distribution $P(\boldsymbol{\phi}_i|(\boldsymbol{Z},\boldsymbol{y}_i))$, $i = 1, 2, \ldots, d$, where $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N]^T \in \mathbb{R}^{N\times p}$, and by assuming a Gaussian distributed noise, and appropriate normalization of the data points is given by the joint multivariate normal distribution,

$$\begin{bmatrix} \boldsymbol{y}_i \\ \phi_i(\boldsymbol{x}_*) \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{Z},\boldsymbol{Z}|\boldsymbol{\theta}) + \sigma^2\boldsymbol{I} & \boldsymbol{k}(\boldsymbol{Z},\boldsymbol{z}_*\boldsymbol{\theta}) \\ \boldsymbol{k}(\boldsymbol{z}_*,\boldsymbol{Z}|\boldsymbol{\theta}) & k(\boldsymbol{z}_*,\boldsymbol{z}_*|\boldsymbol{\theta}) \end{bmatrix}\right). \tag{29}$$

It can be shown that the posterior conditional distribution

$$P(\phi_i(\boldsymbol{x}_*)|\boldsymbol{y}_i, \boldsymbol{Z}, \boldsymbol{z}_*) \tag{30}$$

can be analytically derived, and the expected value and covariance of the estimation are given by

$$\bar{\boldsymbol{\phi}}_i(\boldsymbol{z}_*) = \boldsymbol{k}(\boldsymbol{z}_*,\boldsymbol{Z}|\boldsymbol{\theta})\left(\boldsymbol{K}(\boldsymbol{Z},\boldsymbol{Z}|\boldsymbol{\theta}) + \sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{y}_i, \tag{31}$$

$$\sigma_*^2 = k(\boldsymbol{z}_*,\boldsymbol{z}_*|\boldsymbol{\theta}) - \boldsymbol{k}(\boldsymbol{z}_*,\boldsymbol{Z}|\boldsymbol{\theta})[\boldsymbol{K}(\boldsymbol{Z},\boldsymbol{Z}|\boldsymbol{\theta}) + \sigma^2\boldsymbol{I}]^{-1}\boldsymbol{k}(\boldsymbol{Z},\boldsymbol{z}_*|\boldsymbol{\theta}). \tag{32}$$

The hyperparameters in the above equations are estimated by minimizing the marginal likelihood that is given by

$$\begin{aligned}\ell &= -\log p(\boldsymbol{y}_i|\boldsymbol{Z},\boldsymbol{\theta}) \\ &= \frac{1}{2}\boldsymbol{y}_i^T[\boldsymbol{K}(\boldsymbol{Z},\boldsymbol{Z}|\boldsymbol{\theta}) + \sigma^2\boldsymbol{I}]^{-1}\boldsymbol{y}_i + \frac{1}{2}\log[\boldsymbol{K}(\boldsymbol{Z},\boldsymbol{Z}|\boldsymbol{\theta}) + \sigma^2\boldsymbol{I}] \\ &\quad + \frac{N}{2}\log 2\pi.\end{aligned}$$

Here, for the forecasting of the EEG and FOREX signals, we used a mixed kernel composed by (see, e.g., Corani *et al.*[57]), namely, a radial basis function kernel,

$$k(\boldsymbol{z}_i,\boldsymbol{z}_j) = \theta_1^2\exp\left(-\frac{||\boldsymbol{z}_i - \boldsymbol{z}_j||_{L_2}^2}{2\theta_2^2}\right); \tag{33}$$

a linear kernel,

$$k(\boldsymbol{z}_i,\boldsymbol{z}_j) = \theta_3^2 + \theta_4^2\langle\boldsymbol{z}_i,\boldsymbol{z}_j\rangle; \tag{34}$$

a periodic kernel,

$$k(\boldsymbol{z}_i,\boldsymbol{z}_j) = \theta_5^2\exp\left(-\frac{2\sin^2(\pi||\boldsymbol{z}_i - \boldsymbol{z}_j||_{L_2}/\theta_6)}{\theta_7^2}\right); \tag{35}$$

and a white noise kernel,

$$k(\boldsymbol{z}_i,\boldsymbol{z}_j) = \theta_8^2\delta_{ij}. \tag{36}$$

For the PDEs, we used a kernel of the Matérn class, and, in particular, the Matérn52 kernel given by[56]

$$k(\boldsymbol{z}_i,\boldsymbol{z}_j) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}||\boldsymbol{z}_i - \boldsymbol{z}_j||_{L_2}}{l}\right)^{\nu}K_\nu\left(\frac{\sqrt{2\nu}||\boldsymbol{z}_i - \boldsymbol{z}_j||_{L_2}}{l}\right), \tag{37}$$

where $\nu = \frac{5}{2}$, K_ν is the modified second kind Bessel function, and $l > 0$.

V. SOLUTION OF THE PRE-IMAGE PROBLEM

The final step is to solve the pre-image problem, i.e., "lift" the predictions made by the reduced-order surrogate models on the manifold back to the original high-dimensional space. In the case of PCA, this task is trivial, as the solution to the reconstruction problem, given by

$$\underset{\boldsymbol{U}_d\in\mathbb{R}^{D\times d}}{\operatorname{argmin}}\sum_{i=1}^{N}||\boldsymbol{x}_i - \boldsymbol{U}_d\boldsymbol{U}_d^T\boldsymbol{x}_i||_{L_2}^2, \quad \boldsymbol{U}_d^T\boldsymbol{U}_d = I, \tag{38}$$

is just a linear transformation, which maximizes the variance of the data on the linear manifold, and is given by the first d principal eigenvectors of the covariance matrix.

In the case of nonlinear manifold learning algorithms, such as LLE and DMs, we want to learn the inverse map (lifting operator),

$$\mathscr{L} \equiv \mathscr{R}^{-1} : \mathscr{R}(\boldsymbol{X}) \to \boldsymbol{X}, \tag{39}$$

for new samples on the manifold $\boldsymbol{y}_* \notin \mathscr{R}(\boldsymbol{X})$. This inverse problem is referred as the "pre-image" problem. The "out-of-sample extension" problem usually refers to the direct problem, i.e., that of learning the direct embedding map (i.e., the restrictions to the manifold) $\mathscr{R}(\boldsymbol{X}) : \boldsymbol{X} \to \mathscr{R}$ for new samples in the input space $\boldsymbol{x}_* \notin \boldsymbol{X}$. Toward this aim, a well-established methodology is the Nyström extension.[58]

In general, the solution of the pre-image problem can be posed as

$$\arg\min_{\boldsymbol{c}}||\boldsymbol{y}_* - \mathscr{R}(\mathscr{L}(\boldsymbol{y}_*)|\boldsymbol{c}))||, \tag{40}$$

subject to a constraint, where $\mathscr{L}(\cdot)|\boldsymbol{c}$ is the lifting operator depending on some parameters $\boldsymbol{c}$.

Below, we describe the reconstruction methods that we used in this work, namely, RBFs interpolation and geometric harmonics (GH), which provide a direct solution of the inverse problem, thus giving some insight about their implementation and pros and cons. For a review and a comparison of such methods in the framework of chemical kinetics, see Chiavazzo *et al.*[30]

A. Radial basis function (RBF) interpolation

The lifting operator is constructed with interpolation through RBFs among the corresponding set of say, k neighbors of the new point $\boldsymbol{y}_*$. The lifting operator is defined by[30]

$$\mathscr{L}(\boldsymbol{y}_*) = x_{i*} = \sum_{j=1}^{k}c_{ji}\psi(||\boldsymbol{y}_* - \boldsymbol{y}_j||), \quad i = 1, 2, \ldots, D, \tag{41}$$

where x_{i*} is the ith coordinate of $\boldsymbol{x}_*$, the $\boldsymbol{y}_j$'s are the neighbors of the unseen sample $\boldsymbol{y}_*$ on the manifold, and ψ is the radial basis function.

Similarly, the restriction operator can be written as

$$\mathscr{R}(\boldsymbol{x}_*) = y_{i*} = \sum_{j=1}^{k} c_{ji}\psi(||\mathscr{L}(\boldsymbol{y}_*) - \mathscr{L}(\boldsymbol{y}_j)||), \quad i = 1, 2, \ldots, D, \tag{42}$$

where $\mathscr{L}(\boldsymbol{y}_j) = \boldsymbol{x}_j$ is the known image of $\boldsymbol{y}_j$ (the neighbors of $\boldsymbol{y}_*$ in the ambient space).

Two of the most common RBFs used for this task are the Gaussian kernel defined as

$$\psi(\boldsymbol{y}_i, \boldsymbol{y}_j) = \exp\left(-\varepsilon||\boldsymbol{y}_i - \boldsymbol{y}_j||_{L_2}^p\right), \tag{43}$$

where ε is the so-called shape parameter and the so-called radial powers defined as

$$\psi(\boldsymbol{y}_i, \boldsymbol{y}_j) = ||\boldsymbol{y}_i - \boldsymbol{y}_j||_{L_2}^p, \tag{44}$$

where p is an odd integer.

Thus, the unknown coefficients c_{ji} of the lifting operator are given by the solution of the following linear system:

$$\boldsymbol{A}\begin{bmatrix} c_{1i} \\ c_{2i} \\ \vdots \\ c_{ki} \end{bmatrix} = \begin{bmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{ki} \end{bmatrix}, \quad i = 1, 2, \ldots, D, \tag{45}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 0 & \psi(\boldsymbol{y}_1 - \boldsymbol{y}_2) & \ldots & \psi(\boldsymbol{y}_1 - \boldsymbol{y}_k) \\ \psi(\boldsymbol{y}_2 - \boldsymbol{y}_1) & 0 & \ldots & \psi(\boldsymbol{y}_2 - \boldsymbol{y}_k) \\ \vdots & \vdots & \ldots & \vdots \\ \psi(\boldsymbol{y}_k - \boldsymbol{y}_1) & \psi(\boldsymbol{y}_k - \boldsymbol{y}_2) & \ldots & 0 \end{bmatrix}, \tag{46}$$

and x_{ji} is the ith coordinate of the jth point in the ambient space, whose restriction to the manifold is the jth nearest neighbor of $\boldsymbol{y}_*$. Then, (41) can be used to find the coordinates of $\boldsymbol{x}_*$ in the ambient space.

1. Convergence and efficiency

It has been proven (see Monning *et al.*[59] and the references therein) that for Gaussian RBFs, the approximation convergence as defined by the fill distance, say, h, in terms of L_{inf} to a function in the subspace spanned by the RBFs (also called native space) is exponential. However, for any practical purposes, Gaussian kernels result to an ill-conditioned matrix $\boldsymbol{A}$ when k is large,[59,60] relatively narrow native spaces of convergence, while the optimal selection of the optimal shape parameter imposes an extra burden. In such cases, one can resort to established techniques for the solution of (45) using, for example, the Moore–Penrose pseudoinverse of $\boldsymbol{A}$ via singular value decomposition or Tikhonov regularization. On the other hand, radial powers are better for the task of interpolation compared to Gaussian kernels, as they do not suffer from saturation error, thus result in a wider native space of convergence and in faster convergence rates.[59,61] Here, for our computations, we used radial powers with $p = 1$.

B. Geometric harmonics (GHs)

GHs are a set of functions that allow the extension of the embedding of new unseen points on the manifold $\boldsymbol{x}_* \notin \boldsymbol{X}$, which are not given in the set of points used for building the embedding.[23] Their derivation is based on the Nyström (or quadratic) extension method,[58] which has been used for the numerical solutions of integral equations,[62,63] and in particular, the Fredholm equation of a second kind, read as

$$f(t) = g(t) + \mu\int_a^b k(t, s)f(s)ds, \tag{47}$$

where $f(t)$ is the unknown function, while $k(t, s)$ and $g(t)$ are known. The Nyström method starts with the approximation of the integral, i.e.,

$$\int_a^b y(s)ds \approx \sum_{j=1}^{N} w_j y(s_j), \tag{48}$$

where s_j are N appropriately chosen collocation points and w_j are the corresponding weights, which are determined, e.g., by the Gauss–Jacobi quadrature rule. Then, by using (48) in (47) and evaluating f and g at the N collocation points, we get the following approximation:

$$(\boldsymbol{I} - \mu\tilde{\boldsymbol{K}})\hat{\boldsymbol{f}} = \boldsymbol{g}, \tag{49}$$

where the matrix $\tilde{\boldsymbol{K}}$ has elements $\tilde{k}_{ij} = k(s_i, s_j)w_j$. Based on the above, the solution of the homogenous Fredholm problem ($\boldsymbol{g} = 0$) is given by the solution of the eigenvalue problem,

$$\tilde{\boldsymbol{K}}\hat{\boldsymbol{f}} = \frac{1}{\mu}\hat{\boldsymbol{f}}, \tag{50}$$

i.e.,

$$\sum_{j=1}^{N} w_j k(s_i, s_j)\hat{f}_j = \frac{1}{\mu}\hat{f}_i, \quad i = 1, 2, \ldots, N, \tag{51}$$

where $\hat{f}_i = \hat{f}(s_i)$ is the ith component of $\hat{\boldsymbol{f}}$. The Nyström extension of $f(t)$, using a set of N sample (collocation) points, at an arbitrary point x in the full domain, is given by

$$\mathscr{E}(f(x)) = \hat{f}(x) = \mu\sum_{j=1}^{N} w_j k(x, s_j)\hat{f}_j. \tag{52}$$

Within the framework of DMs, we seek for the out-of-the-sample (filtered) extension of a real-valued function f defined at N sample points $\boldsymbol{x}_i \in \boldsymbol{X}$ to one or more unseen points $\boldsymbol{x}_* \notin \boldsymbol{X}$. The function f can be, for example, a DM coordinate $\lambda_j^t \boldsymbol{v}_j(\boldsymbol{x}_i)$, $j = 1, 2, \ldots, d$, or another function representing the output of a regression model.[64,65]

Recalling that the eigenvectors $\boldsymbol{v}_j$ form a basis, the extension is implemented[23] by first expanding $f(\boldsymbol{x}_i)$ in the first d parsimonious eigenvectors $\boldsymbol{v}_l$ of the Markovian matrix $\boldsymbol{P}^t$,

$$\hat{f}(\boldsymbol{x}_i) = \sum_{l=1}^{d} a_l \boldsymbol{v}_l(\boldsymbol{x}_i), \quad i = 1, 2, \ldots, N,$$

where $a_l = \langle \boldsymbol{v}_l, \boldsymbol{f}\rangle$ are the projection coefficients of the function on the first d parsimonious eigenvectors and $\boldsymbol{f} \in \mathbb{R}^N$ is the vector containing the values of the function at the N points $\boldsymbol{x}_i$. Then, one

computes the Nyström extension of $\hat{f}$ at $\boldsymbol{x}_*$ using the same projection coefficients as

$$\mathscr{E}(\hat{f}(\boldsymbol{x}_*)) = \sum_{l=1}^{d} a_l \hat{\boldsymbol{v}}_l(\boldsymbol{x}_*), \tag{53}$$

where

$$\hat{\boldsymbol{v}}_l(\boldsymbol{x}_*) = \frac{1}{\lambda_l^t} \sum_{j=1}^{N} k(\boldsymbol{x}_j, \boldsymbol{x}_*) \boldsymbol{v}_l(\boldsymbol{x}_j), \quad l = 1, 2, \ldots, d. \tag{54}$$

are the corresponding GHs. Scaling up the (filtered) extension of the function f to a set of, say, L new points can be computed using the following matrix product:[23,64]

$$\mathscr{E}(\hat{\boldsymbol{f}}) = \boldsymbol{K}_{L\times N} \boldsymbol{V}_{N\times d} \boldsymbol{\Lambda}_{d\times d}^{-1} \boldsymbol{V}_{d\times N}^{T} \boldsymbol{f}_{N\times 1}, \tag{55}$$

where $\boldsymbol{K}_{L\times N}$ is the corresponding kernel matrix, $\boldsymbol{V}_{N\times d}$ is the matrix with columns the d parsimonious eigenvectors $\boldsymbol{v}_l$, and $\boldsymbol{\Lambda}_{d\times d}$ is the diagonal matrix with elements λ_l^t.

The above direct approach provides a map from the ambient space to the reduced-order space (restriction) and vice versa (lifting).

VI. THE CASE STUDIES

For demonstrating the performance of the proposed numerical framework and comparing the various embedding, modeling, and reconstruction approaches, we used (a) synthetic time series resembling EEG recordings generated by simplistic linear and weakly nonlinear stochastic discrete models, (b) a linear parabolic PDE for which an analytical solution exists, and the celebrated Brusselator model introduced by Prigogine[66] consisting of a system of two nonlinear parabolic PDEs, and (c) a real-world data set of ten key FOREX pairs.

A. The synthetic EEG time series

Our synthetic stochastic signals that resemble EEG time series[67,68] are generated by simplistic linear and weakly nonlinear five-dimensional discrete stochastic models with white noise. Here, we note that specific low-dimensional synthetic EEG time series are tractable, transparent paradigms with simple/trivial dynamics, thus serving to illustrate a "computational proof of concept" of the proposed scheme, for which an MVAR model in the ambient space is expected for any practical means to perform accurately in the presence of low-amplitude noise.

The linear five-dimensional stochastic discrete model is given by the following equations:

$$\begin{aligned}
y_t^{(1)} &= 0.2y_{t-1}^{(1)} - 0.4y_{t-1}^{(2)} + w_t^{(1)},\\
y_t^{(2)} &= -0.5y_{t-1}^{(1)} + 0.15y_{t-1}^{(2)} + w_t^{(2)},\\
y_t^{(3)} &= -0.14y_{t-1}^{(2)} + w_t^{(3)},\\
y_t^{(4)} &= 0.5y_{t-1}^{(1)} - 0.25y_{t-1}^{(2)} + w_t^{(4)},\\
y_t^{(5)} &= 0.15y_{t-1}^{(1)} + w_t^{(5)},
\end{aligned} \tag{56}$$

where for the training process, the model order is assumed to be known (here equal to 1).

A second nonlinear model is given by the following equations (see, e.g., Nicolaou and Constandinou[68]):

$$\begin{aligned}
y_t^{(1)} &= 3.4y_{t-1}^{(1)}\left(1 - {y_{t-1}^{(1)}}^2\right)\exp\left(-{y_{t-1}^{(1)}}^2\right) + w_t^{(1)},\\
y_t^{(2)} &= 3.4y_{t-1}^{(2)}\left(1 - {y_{t-1}^{(2)}}^2\right)\exp\left(-{y_{t-1}^{(2)}}^2\right) + 0.5y_{t-1}^{(1)}y_{t-1}^{(2)} + w_t^{(2)},\\
y_t^{(3)} &= 3.4y_{t-1}^{(3)}\left(1 - {y_{t-1}^{(3)}}^2\right)\exp\left(-{y_{t-1}^{(3)}}^2\right) + 0.3y_{t-1}^{(2)}\\
&\quad + 0.5{y_{t-1}^{(1)}}^2 + w_t^{(3)},\\
y_t^{(4)} &= 0.5y_{t-1}^{(1)} - 0.25y_{t-1}^{(2)} + w_t^{(4)},\\
y_t^{(5)} &= 0.15y_{t-1}^{(1)} + w_t^{(5)}.
\end{aligned} \tag{57}$$

The proposed scheme was also validated through the time series produced by a linear stochastic model with a model order greater than 1, given by the following equations:

$$\begin{aligned}
y_t^{(1)} &= 0.1y_{t-1}^{(1)} - 0.6y_{t-3}^{(2)} + w_t^{(1)},\\
y_t^{(2)} &= -0.15y_{t-3}^{(1)} + 0.8y_{t-3}^{(2)} + w_t^{(2)},\\
y_t^{(3)} &= -0.45y_{t-3}^{(2)} + w_t^{(3)},\\
y_t^{(4)} &= 0.45y_{t-3}^{(1)} - 0.85y_{t-3}^{(2)} + w_t^{(4)},\\
y_t^{(5)} &= 0.95y_{t-2}^{(1)} + w_t^{(5)}.
\end{aligned} \tag{58}$$

In all the above models, the time series $w_t^{(i)}$, $i = 1, 2, 3, 4, 5$, are uncorrelated normally distributed noise with zero mean and unit standard deviation.

B. THE NUMERICAL CONTINUATION OF SOLUTIONS OF (PARABOLIC) PDEs

The proposed scheme can also be potentially used for constructing reduced-order surrogate models to learn and consequently predict the dynamics of high-dimensional systems as those resulting from the discretization of PDEs. Here, we illustrate this possibility by considering two problems. The first one is a linear parabolic PDE given by[69]

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u}{\partial x^2} - u. \tag{59}$$

This provides a simple paradigm that can be easily and successfully treated using PCA. Using Neumann boundary conditions in $[0 \quad \pi]$ and initial conditions $u(x,0) = 0.4\cos(2x) + 1.5$, the analytical solution is given by

$$u(x,t) = 0.4\exp(-5t)\cos(2x) + 1.5\exp(-t). \tag{60}$$

The second problem is the celebrated Brusselator system of nonlinear PDEs that gives rise to sustained oscillations (limit cycles). The model is given by the following coupled system of parabolic

PDEs:

$$\frac{\partial u(x,t)}{\partial t} = D_u \frac{\partial^2 u}{\partial x^2} + a - (1+b)u + vu^2, \tag{61}$$

$$\frac{\partial u(x,t)}{\partial t} = D_v \frac{\partial^2 v}{\partial x^2} + bu - vu^2, \tag{62}$$

with Dirichlet boundary conditions (BCs). Here, for our illustrations, we solved the above system with $a = 1$, $b = 3$, $D_u = D_v = \frac{1}{50}$ using central finite differences with 20 equidistant collocation points in (0 1) with BCs,

$$u(0,t) = u(1,t) = 1, v(0,t) = v(1,t) = 3,$$

and initial conditions (ICs)

$$u(x,0) = 1 + \sin(2\pi x), \quad v(x,0) = 3.$$

For the time integration of the resulting stiff system of 40 ODEs in the interval (0 33], we have used the `ode15s` stiff solver of the Matlab ODE suite[70] with absolute and relative tolerances set to 1×10^{-3} and 1×10^{-6}, respectively. In order to find (i) the optimal reduced-order GPR surrogate model, (ii) the optimal set of values of the hyperparameters (values of the kernel parameters, number of eigenpairs, number of nearest neighbors) for the embedding and the solution of the pre-image problem, and (iii) to assess the prediction performance of the proposed scheme, we split the dataset into training, validation, and test sets. Details on the splitting are given in Sec. VII.

C. The FOREX forecasting problem

We used the `investpy 0.9.14` module of Python[71] to download ten USD-based FOREX pairs daily spot closing prices from the www.investing.com open API, spanning the period 03/09/2001–29/10/2020: EUR/USD, GBP/USD, AUD/USD, NZD/USD, JPY/USD, CAD/USD, CHF/USD, SEK/USD, NOK/USD, and DKK/USD. The time period was selected to contain crucial market crashes like the 2008 Lehman related stock market crash, the 2010 sovereign debt crisis in the Eurozone, and even the most recent COVID-19 related market crash of 2020.

For our analysis, we computed the compounded daily returns of the FOREX pairs as

$$r_{i,t} = \log\left(\frac{S_{i,t}}{S_{i,t-1}}\right), \tag{63}$$

where $S_{i,t}$ is the spot price of the ith FOREX pair in the data set above and $r_{i,t}$ is the corresponding compounded return at date t. However, trading the FOREX markets is not only associated with the spot price movements but also with the so-called currency carry trading. The currency carry-trading involves first the identification of high interest-paying investment assets (e.g., bonds or short-term bank deposits) denominated in a country currency (e.g., Japanese yen). Then, the investment is carried on by borrowing money in another currency for which the paying interest rate is lower. Such a trading requires the approximation of the so-called interest-rate-differential (IRD) excess returns, which should be incorporated in the FOREX price movements (see, e.g., Menkhoff *et al.*[72]). Here, we used the short-term interest rates data retrieved from the OECD database as the proxy of these IRD excess returns. Since the short-term interest rates data are reported on a monthly basis and on an annual percentage format, we constructed daily approximations by linearly interpolating through the available downloaded records (using the `Pandas` module of Python[73]) and normalizing on the basis of an annual calendar period. After this pre-processing step, we denote as $\mathrm{IR}_{\mathrm{USA},t}$ the time series of the daily approximations of the USA short-term interest rates, and $\mathrm{IR}_{i,t}$ is the corresponding time series of each of the other $i = 1, \ldots, 10$ countries. Then, the interest rate differential (IRD) time series are given by

$$\mathrm{IRD}_{i,t} = \mathrm{IR}_{i,t} - \mathrm{IR}_{\mathrm{USA},t}. \tag{64}$$

Finally, the so-called carry-adjusted returns of the FOREX are given by (see also, e.g., Menkhoff *et al.*[72])

$$x_{i,t} = r_{i,t} + \mathrm{IRD}_{i,t}, \tag{65}$$

where $x_{i,t}$ is the ith FX pair "carry" adjusted return corresponding to its raw "unadjusted" market price return, $r_{i,t}$ is defined in (63) and $\mathrm{IRD}_{i,t}$.

For quantifying the potential excess returns (profits), we constructed a trading strategy based on the so-called risk parity rationale (see, e.g., Braga[74]) where each asset is allocated with a portfolio weight, which is proportional to its inverse risk. Thus, one assigns higher portfolio weights to less volatile assets and smaller portfolio weights to more risky assets. Thus, the risk is quantified using the volatility $\sigma_{i,t}$ (measured by the standard deviation of logarithmic returns over a specific time period up to the t trading day) of each asset i of the portfolio plus the total portfolio volatility. In our FOREX problem, the risk parity portfolio allocation practically means investing $1/\sigma_{i,t}$ at each of the $i = 1, 2, \ldots, 10$ FOREX pairs with corresponding carry-adjusted returns $x_{i,t}$. By performing one-step ahead predictions for each of carry-adjusted returns of the 10 pairs, denoted as $\hat{x}_{i,t+1}$, we create a "binary," or otherwise called "directional," trading signal of "buy" or "sell" for each one of the FOREX. Thus, the trading strategy reads as follows:

$$u_{i,t} = \mathrm{sign}(\hat{x}_{i,t+1}) = \begin{cases} 1, & \text{"buy,"} \\ -1, & \text{"sell."} \end{cases} \tag{66}$$

Based on the above, the profit or loss at the next day $(t+1)$ is given by

$$\Pi_{t+1} = \sum_{i=1}^{D} \frac{u_{i,t} x_{i,t+1}}{\sigma_{i,t}}, \tag{67}$$

where $x_{i,t+1}$ is the real ith FOREX pair return at time $t+1$, and Π_{t+1} is the risk parity portfolio return at time $t+1$.

VII. NUMERICAL RESULTS

For the implementation of the numerical algorithms, we used the `datafold`, `sklearn`, and `statsmodels` packages of Python.[75–77] The selection of the eigensolver was based on the sparsity and size of the input matrix: ARPACK was used if the size of the input matrix was greater than 200 and $n + 1 < 10$ (where n is the number of requested eigenvalues); otherwise, the "dense" option was used. The ARPACK package[78] implements implicitly

restarted Arnoldi methods, using a random default starting vector for each iteration, with a tolerance $tol = 10^{-6}$ and a maximum number of 100 iterations. This approach is implemented by the function `scipy.sparse.linalg.eigsh` of the `scipy` module (upon which the `sklearn` one depends).[79] The "dense" eigensolver is implemented by the function `eigh` of the `scipy` module and returns the eigenvalues and eigenvectors computed using the LAPACK routine `syevd` using the divide and conquer algorithm.[80,81] We used the default value of the tolerance of the Newton–Raphson iterations, which is of the order of the floating-point precision, and the maximum number of iterations was set to $30N$ iterations, where N is the size of the matrix.

The DM embedding on the training set of all data sets was performed using d parsimonious eigenvectors.[82] Here, for the construction of the graph at the first step, we used the standard Gaussian kernel, defined by

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{L_2}^2}{\sigma}\right), \tag{68}$$

where σ is a scaling parameter that controls the size of the neighborhood (or the connectivity of the graph). For its derivation, we follow the systematic approach provided by Singer *et al.*[83] The full kernel is used for the calculations.

The VAR models were trained using the VAR class of the `statsmodels.tsa.vector_ar.var_model` routine using the OLS default method for the parameter estimation. The corresponding LAPACK function used to solve the least-squares problem is the default `gelsd`, which exploits the QR or LQ factorization of the input matrix.

The hyperparameters of the GPR model were optimized using the (default) L-BFGS-B algorithm of the `scipy.optimize.minimize` method.[84,85] The gradient vector was estimated using forward finite differences with the numerical approximations of the Jacobian being performed with a default step size $eps = 10^{-8}$. We used the default values for tolerances and the maximum number of function evaluations and the maximum number of iterations (i.e., 15 000, as well as the default value for the maximum number of line searches per iteration, i.e., 20).

For the lifting task for the problems of EEG signals and FOREX, 50 nearest neighbors were considered for interpolation by all the methods. The underlying k-NN algorithm is based on the algorithm proposed in Maneewongvatana and Mount.[86] Using different values of the number of nearest neighbors within the range 20–100 did not change the outcomes of the analysis. In the case of RBF interpolation, the underlying linear system of equations was solved by the LAPACK `dgesv` routine from `scipy.linalg.lapack.dgesv`, which implements the default method of the LU decomposition with partial pivoting. For the GH approach, we used the Gaussian kernel. In effect, we are performing "double" DM here—computing diffusion maps on the leading retained Diffusion map components for the reduced embedding. This procedure, suggested in Chiavazzo *et al.*,[30] can actually be performed "once and for all" globally.[87]

The computations were performed using a system with an Intel Core i7-8750H CPU @2.20 GHz and 16GB of RAM.

A. Synthetic time series

For both the linear and nonlinear models, we produced 2000 points. We used 1500 points for learning the manifold and for training the various models and 500 points to test the forecasting performance of the various schemes. The forecasting performance was tested using the iterative mode, i.e., by training the models for one-step ahead predictions and then simulating the trained model iteratively to predict future values. The performance was measured using the root mean square error (RMSE) of the residuals, read as

$$\mathrm{RMSE} = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i)^2}, \tag{69}$$

where $\hat{\boldsymbol{x}}_i$ are the predictions and $\boldsymbol{x}_i$ the actual data. To quantify the forecasting intervals due to the stochasticity of the models, we performed 100 runs for each combination of manifold learning algorithms (DMs and LLE), models (MVAR and GPR), and lifting methods (RBFs and GHs), reporting the median and the 5th and 95th percentiles of the resulting RMSE. The RMSE values obtained with the naïve random walk model, as well as with the MVAR and GPR models trained in the original space, are also given for comparison purposes.

In Table I, we report the forecasting statistics of the time series produced with the linear model given by (56) as obtained over 100 runs. As it is shown, both the MVAR and GPR models trained in the original five-dimensional space outperform the naïve random walk. The RMSEs of the MVAR and GPR models suggest a good match with the stochastic model, with the residuals being approximately within one standard deviation of the distribution of the noise level. In the same table, we provide the corresponding forecasting statistics as obtained with the proposed "embed–forecast–lift" scheme for the various combinations of manifold learning algorithms, regression models, and lifting approaches. For our illustrations, we have chosen the first two parsimonious DM coordinates and the corresponding two LLE eigenvectors. As shown, the best performance is obtained with the GH lifting operator. Using GHs for lifting and any combination of the selected manifold learning algorithms and models outperforms all other combinations, thus resulting in practically the same RMSE values when compared with the predictions made in the original space. This suggests that the proposed "embed–forecast–lift" scheme applied in the 5D feature space provides a very good reconstruction of the predictions made in the original space. On the other hand, lifting with RBF interpolation with LU decomposition generally resulted in poor reconstructions of the high-dimensional space, thus, in many cases, giving wide forecasting intervals that contained the median value of the naïve random walk RMSE.

Next, in Table II, we report the forecasting statistics of the time series for the nonlinear stochastic model (57) as obtained over 100 runs. As in the case of the linear stochastic model, both MVAR and GPR trained in the original 5D space outperform the naïve random walk. The resulting RMSE values suggest a good match with the nonlinear stochastic model. Yet, the match is poorer than the one obtained for the linear model.

We also provide the corresponding forecasting statistics as obtained with the proposed "embed–forecast–lift" scheme. For

TABLE I. Linear stochastic model (56). RMSE statistics (median, 5th, and 95th percentiles over 100 runs) for each of the five variables as obtained by (a) training MVAR and GPR models with model order one in the original 5D space [MVAR(OS), GPR(OS)] and (b) the proposed "embed–forecast–lift" scheme for all combinations of the manifold learning algorithms (DMs and LLE) with two coordinates, models (MVAR and GPR), and lifting approaches (RBFs and GHs). For comparison purposes, the RMSEs obtained with the naïve random walk are also reported.

Model/variable	$y_t^{(1)}$	$y_t^{(2)}$	$y_t^{(3)}$	$y_t^{(4)}$	$y_t^{(5)}$
Random walk	1.374 (1.287,1.466)	1.446 (1.333,1.528)	1.427 (1.339,1.497)	1.551 (1.455,1.654)	1.425 (1.335,1.512)
MVAR(OS)	1.151 (1.077,1.218)	1.180 (1.113,1.269)	1.010 (0.966,1.062)	1.224 (1.156,1.281)	1.011 (0.960,1.063)
GPR(OS)	1.151 (1.077,1.218)	1.179 (1.113,1.272)	1.010 (0.966,1.061)	1.224 (1.154,1.284)	1.011 (0.959,1.064)
DM-GPR-GH	1.153 (1.082,1.220)	1.184 (1.117,1.273)	1.012 (0.966,1.061)	1.228 (1.160,1.287)	1.013 (0.959,1.065)
LLE-GPR-GH	1.155 (1.077,1.218)	1.182 (1.115,1.272)	1.011 (0.966,1.065)	1.230 (1.157,1.292)	1.014 (0.962,1.064)
DM-GPR-RBF	1.260 (1.111,2.201)	1.390 (1.149,1.980)	1.163 (0.997,2.331)	1.365 (1.173,1.939)	1.166 (0.997,1.853)
LLE-GPR-RBF	1.197 (1.103,1.452)	1.234 (1.131,1.461)	1.076 (0.993,1.582)	1.270 (1.175,1.468)	1.103 (0.990,1.553)
DM-MVAR-GH	1.151 (1.078,1.220)	1.180 (1.113,1.269)	1.011 (0.966,1.063)	1.225 (1.155,1.288)	1.011 (0.959,1.063)
LLE-MVAR-GH	1.155 (1.077,1.218)	1.182 (1.115,1.271)	1.011 (0.966,1.065)	1.229 (1.158,1.293)	1.014 (0.962,1.064)
DM-MVAR-RBF	1.289 (1.124,2.132)	1.312 (1.146,2.018)	1.135 (0.996,1.635)	1.341 (1.186,1.933)	1.106 (0.99,1.601)
LLE-MVAR-RBF	1.187 (1.106,1.477)	1.230 (1.14,1.477)	1.080 (0.995,1.602)	1.264 (1.177,1.479)	1.096 (0.989,1.558)

embedding in the reduced space, we have taken three (parsimonious) coordinates. Again, the best performance is obtained with the GH lifting operator for any combination of manifold learning algorithms and models. Importantly, the reconstruction errors between the forecasts with the "full" MVAR and GPR models trained directly in the original 5D space and the ones obtained with the proposed "embed–forecast–lift" scheme are negligible up to a three-digit accuracy for all five variables. As with the previous case, lifting with RBFs resulted in a poor reconstruction of the high-dimensional space, with forecasting intervals containing the median RMSE value of the naïve random walk.

Finally, in Table III, we report the RMSE statistics for the time series produced with the linear model with a model order three [see (58)] as obtained over 100 runs from (a) the naïve random walk model applied to the original 5D space, (b) the MVAR models trained in the original 5D data set with model orders one [MVAR(1)] and three [MVAR(3)], and (c) the proposed "embed–forecast–lift" method with the embedding applied to the original 5D data set using DM and LLE for embedding with three coordinates. In the reduced-order space, we have trained MVAR models with model orders 1 and 3 and used GHs for lifting. The best results were obtained when using the proposed scheme with DMs for embedding and a model order 3 for the training of the MVAR model in the corresponding manifold. Importantly, the implementation of DM-MVAR(3)-GH succeeds in reproducing quite well the results obtained by training MVAR with a maximum delay of three in the original space.

B. Propagation of solutions of PDEs

For both problems, we show the results obtained with PCA, DMs, and LLE for embedding the data on the low-dimensional manifold, GPR as a reduced-order surrogate model, and RBFs and GHs for solving the pre-image problem. For the linear PDE (59), we used the analytical solution [see Eq. (60)] with 100 equidistant spatial points to produce 5000 solution profiles in the time interval [0 0.25]. We used the first 4000 solution profiles to learn the manifold and to train the surrogate reduced-order GPR model, the next 500 solution profiles as a validation test to get the optimal set of values of the hyperparameters for the solution of the pre-image problem, thus resulting in the best prediction for this interval, and the last unseen 500 solution profiles as a test set to assess the prediction performance of the proposed scheme. In Fig. 1(a), we show the analytical solution of the PDE. We used the first 4000 solution profiles for training; blue lines depict the validation data and red lines the unseen data. In Fig. 1(b), we show the errors with respect to the analytical solution using the leading two principal components of PCA. We note that the PCA fails to adequately approximate the solution using just the first principal component. In Figs. 1(c) and 1(d),

TABLE II. Nonlinear stochastic model (57). RMSE statistics (median, 5th, and 95th percentiles over 100 runs) for each of the five variables as obtained by (a) training MVAR and GPR in the original 5D space [MVAR(OS), GPR(OS)] with model order one, (b) using the proposed "embed–forecast–lift" scheme for all the combinations of the manifold learning algorithms (DMs and LLE), models (MVAR and GPR), and lifting approaches (RBFs and GHs). For the embedding, three coordinates were used.

Model/variable	$y_t^{(1)}$	$y_t^{(2)}$	$y_t^{(3)}$	$y_t^{(4)}$	$y_t^{(5)}$
Random walk	1.711 (1.597,1.815)	2.085 (1.939,2.258)	2.292 (2.147,2.437)	1.731 (1.62,1.819)	1.435 (1.354,1.527)
MVAR(OS)	1.181 (1.123,1.234)	1.438 (1.359,1.555)	1.565 (1.463,1.648)	1.212 (1.151,1.272)	1.021 (0.967,1.076)
GPR(OS)	1.182 (1.123,1.233)	1.437 (1.360,1.555)	1.684 (1.540,1.799)	1.213 (1.151,1.272)	1.020 (0.966,1.076)
DM-GPR-GH	1.181 (1.123,1.234)	1.440 (1.363,1.56)	1.573 (1.477,1.654)	1.214 (1.151,1.276)	1.022 (0.968,1.077)
LLE-GPR-GH	1.182 (1.126,1.236)	1.451 (1.363,1.563)	1.579 (1.473,1.675)	1.214 (1.151,1.273)	1.023 (0.97,1.076)
DM-GPR-RBF	1.488 (1.172,8.88)	1.618 (1.392,9.95)	2.060 (1.642,6.102)	1.710 (1.203,16.868)	1.199 (1.012,9.108)
LLE-GPR-RBF	1.214 (1.133,1.557)	1.449 (1.365,1.561)	1.587 (1.502,1.692)	1.254 (1.159,1.446)	1.088 (0.989,1.5)
DM-MVAR-GH	1.181 (1.123,1.234)	1.438 (1.365,1.555)	1.571 (1.472,1.649)	1.213 (1.151,1.274)	1.022 (0.968,1.076)
LLE-MVAR-GH	1.182 (1.125,1.235)	1.452 (1.364,1.563)	1.586 (1.468,1.669)	1.214 (1.152,1.275)	1.023 (0.97,1.076)
DM-MVAR-RBF	1.387 (1.169,6.743)	1.534 (1.368,4.634)	1.902 (1.609,4.5)	1.560 (1.199,9.294)	1.131 (0.99,4.655)
LLE-MVAR-RBF	1.213 (1.135,1.507)	1.445 (1.366,1.561)	1.570 (1.484,1.657)	1.253 (1.159,1.621)	1.091 (0.998,1.674)

TABLE III. Linear stochastic model with model order three [see (58)]. RMSE statistics (median, 5th, and 95th percentiles over 100 runs) for 100 simulations for each of the five variables as obtained by (a) training MVAR(1) and MVAR(3) models in the original 5D feature space and (b) the proposed scheme with DMs and LLE, MVAR(1) and MVAR(3) models, and GHs for lifting. The embedding with DMs and LLE was implemented using three coordinates.

Model/variable	$y_t^{(1)}$	$y_t^{(2)}$	$y_t^{(3)}$	$y_t^{(4)}$	$y_t^{(5)}$
Random walk	2.138 (1.898,2.461)	2.910 (2.412,3.551)	1.930 (1.755,2.152)	3.499 (2.978,4.195)	2.477 (2.253,2.739)
MVAR(1)	1.629 (1.466,1.851)	2.121 (1.817,2.515)	1.382 (1.275,1.517)	2.567 (2.244,2.997)	1.840 (1.699,2.031)
MVAR(3)	1.611 (1.449,1.833)	2.092 (1.787,2.483)	1.371 (1.265,1.501)	2.531 (2.206,2.963)	1.824 (1.689,2.019)
DM-MVAR(1)-GH	1.630 (1.467,1.854)	2.121 (1.825,2.515)	1.383 (1.273,1.516)	2.569 (2.244,2.998)	1.843 (1.697,2.037)
LLE-MVAR(1)-GH	1.651 (1.472,1.905)	2.159 (1.832,2.567)	1.397 (1.281,1.545)	2.619 (2.269,3.080)	1.866 (1.705,2.084)
DM-MVAR(3)-GH	1.621 (1.455,1.839)	2.103 (1.791,2.484)	1.376 (1.267,1.509)	2.546 (2.217,2.972)	1.835 (1.694,2.028)
LLE-MVAR(3)-GH	1.649 (1.473,1.902)	2.149 (1.820,2.572)	1.393 (1.277,1.550)	2.608 (2.252,3.065)	1.862 (1.703,2.094)

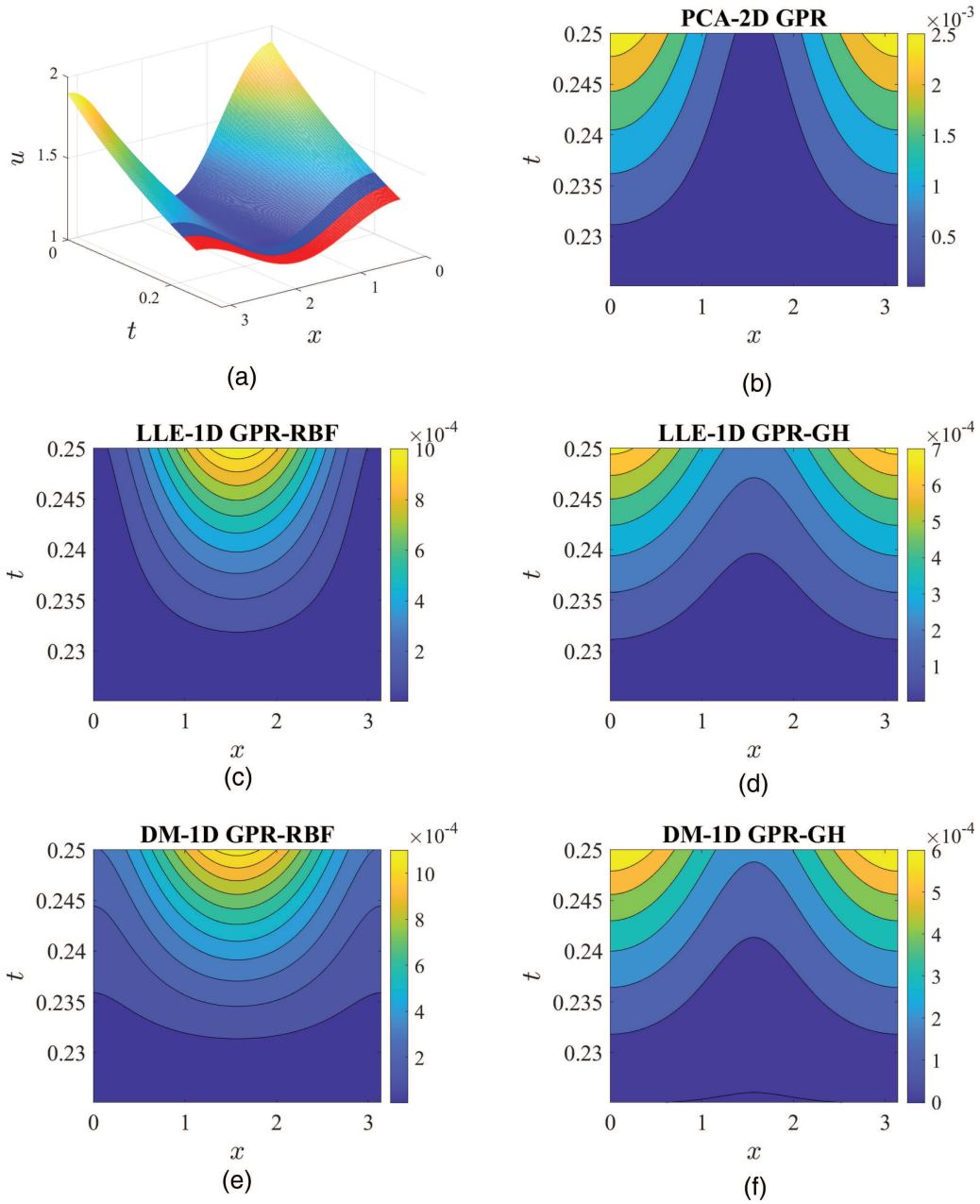

FIG. 1. Linear PDE (60). (a) Analytical solution on 100 equidistant spatial points in [0 π] in the time interval [0 0.25] tessellated in 5000 equidistant points; the blue mesh (500 solution profiles) corresponds to the validation set and the red mesh (last 500 solution profiles) depicts the unseen solution profiles that are used as a test set. (b)–(f) Spatiotemporal errors with respect to the analytical solution obtained with a GPR model trained at the embedded space using a: (b) 2D PCA, (c) 1D LLE for embedding and RBFs for lifting, (d) 1D LLE for embedding and GHs for lifting, (e) 1D DM for embedding and RBFs for lifting, and (f) 1D DM for embedding and GHs for lifting.

we show the spatiotemporal errors with respect to the analytical solution using the first LLE coordinate and RBF and GH, respectively, for lifting. Similarly, in Figs. 1(e) and 1(f), we show the spatiotemporal errors with respect to the analytical solution using the first DM for embedding and RBF and GH, respectively, for lifting. We see that for any practical purposes, the 1D LLE and 1D DM with both RBFs and GHs for lifting perform equivalently and both outperform the 2D PCA.

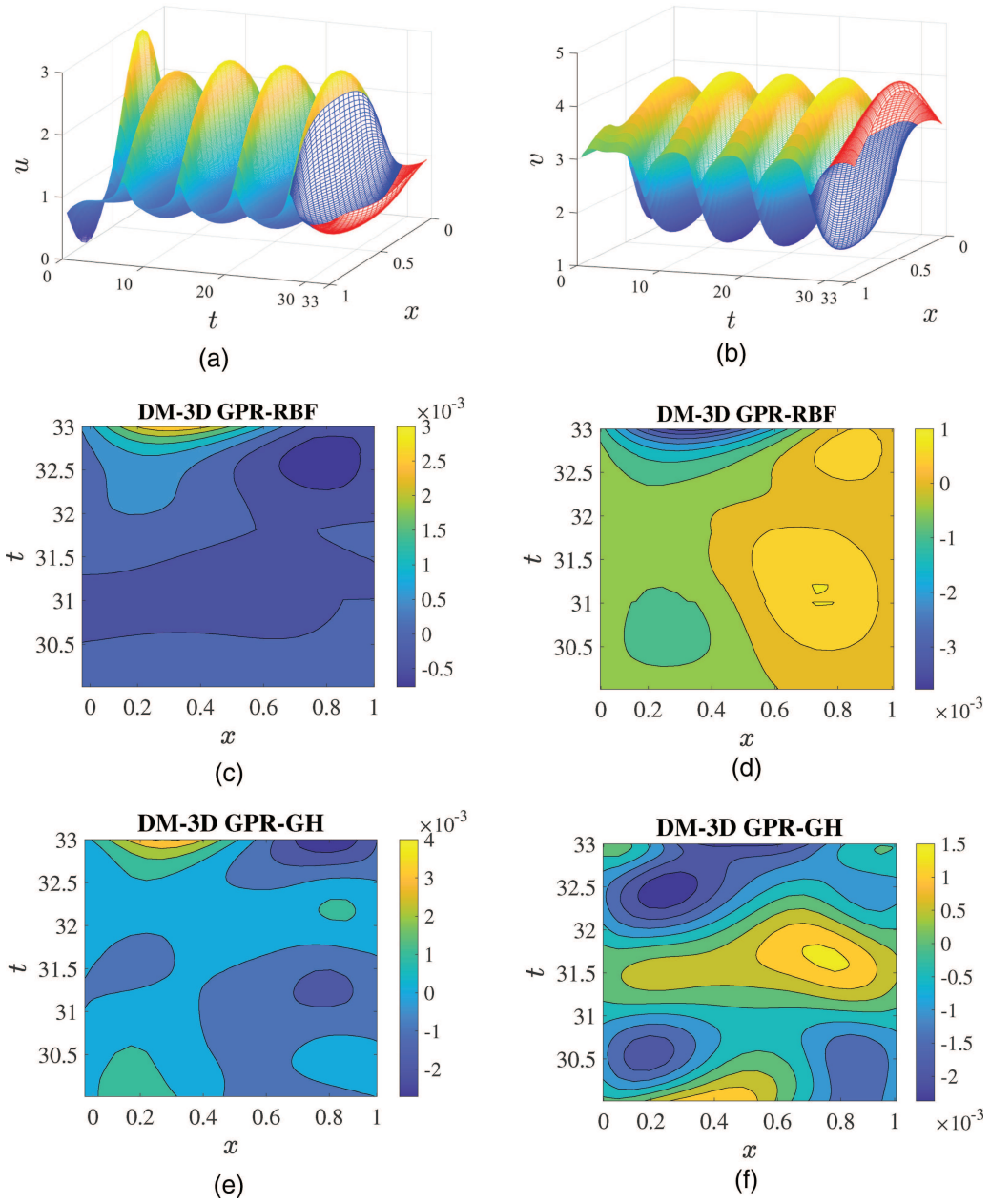

FIG. 2. Brusselator model given by Eqs. (61) and (62), (a) $u(x,t)$, (b) $v(x,t)$ profiles as derived from the numerical solution using central finite differences on 20 equidistant spatial points in (0 1) and the `ode15s` stiff solver of the Matlab ODE suite (with the absolute and relative tolerances set to 1×10^{-3} and 1×10^{-6}, respectively) for the time integration of the resulting stiff system of 40 ODEs in [0 33]; the blue mesh (250 solution profiles) depicts the validation set and the red mesh (last 300 solution profiles) depicts the unseen solution profiles that are used as a test set. (c)–(f) Spatiotemporal errors for $u(x,t)$ (c) and (e) and $v(x,t)$ (d) and (f) with respect to the reference numerical solution, obtained with a GPR model trained at the embedded space using 3D DMs for embedding and (c) and (d) RBF for lifting and (e) and (f) GH for lifting.

For the 1D Brusselator model of coupled nonlinear parabolic PDEs given by Eqs. (61) and (62), we sampled the solution in [0 33] every $DT = 0.01$, thus getting 3300 solution profiles; we used the first 2750 as a training set to learn the manifold and train the reduced-order surrogate GPR model, the next 250 points to find the optimal set of values of the hyperparameters for the solution of the pre-image problem, resulting in the best prediction performance for this interval, and finally, the last 300 unseen points as a test set

to assess the prediction performance scheme. In Figs. 2(a) and 2(b), we show the solution profiles for $u(x,t)$ and $v(x,t)$, respectively. In Figs. 2(c)–2(f), we show the spatiotemporal errors with respect to the numerical solution using three DM coordinates for embedding and RBFs [Figs. 2(c) and 2(d)] and GHs [Figs. 2(e) and 2(f)], respectively, for lifting. In Figs. 2(c) and 2(e), we show the spatiotemporal errors for $u(x,t)$ and in Figs. 2(d) and 2(f) the spatiotemporal errors for $v(x,t)$. For this nonlinear problem, the use of PCA fails to predict-reconstruct adequately the solution profiles even if one takes more than 20 principal components.

C. FOREX trading

Here, we assessed the performance of the proposed forecasting framework in the FOREX trading application described earlier, under the annualized Sharpe (SH) ratio[88] of the constructed risk parity portfolio. The returns are computed by (67). The basic formula for calculating the SH ratio is given by

$$\mathrm{SH} = \frac{\mu_\Pi - R_f}{\sigma_\Pi}, \tag{70}$$

where μ_Π is the sample average returns of the risk parity portfolio and σ_Π the corresponding volatility, while the R_f is the risk-free rate, usually set equal to the annual yield of the US Treasury Bonds. Here, for our analysis, we have set $R_f = 0$, which is a fair approximation of the reality.

The underlying dynamics of the FOREX market is in general non-autonomous, at least over a long time period. In principle, there are time-varying exogenous factors, including macroscopic economic indices, social interaction, and mimesis (see, e.g., Papaioannou *et al.*,[89] where machine learning has been used to forecast FOREX taking into account Twitter messages), but also seasonal factors and rare events (such as the recent COVID19 pandemic), which influence FOREX over time. This comes in contrast to the synthetic time series that we have examined here and also other autonomous models that have been considered in other studies that served as benchmarks to assess the performance of the various schemes (see also the discussion in the conclusions, Sec. VIII). Hence, to cope with such changes of the FOREX market and in general of financial assets, one would set up a sliding/rolling window, then train models within the rolling window, and perform (usually) one-day ahead forecasts for trading purposes.

For our illustrations, we assessed the performance of the proposed scheme based on the so-called risk parity trading strategy using a one-year (250 trading days) embedding rolling window and the last 50 or 100 days of the 250-day rolling window for training. The forecasting performance of the proposed scheme using DMs and LLE for embedding with three coordinates, MVAR and GPR for prediction, and GHs for lifting was compared against the naïve random walk and the full MVAR and GPR models trained and implemented directly in the original space. A comparison against the linear embedding provided by PCA with the same number of principal components was also performed.

Figure 3 depicts the SH ratios obtained with the various methods. As it is shown, the proposed schemes using the DM and LLE algorithms for embedding outperform all the other schemes when considering the same size of the training window. In particular, the

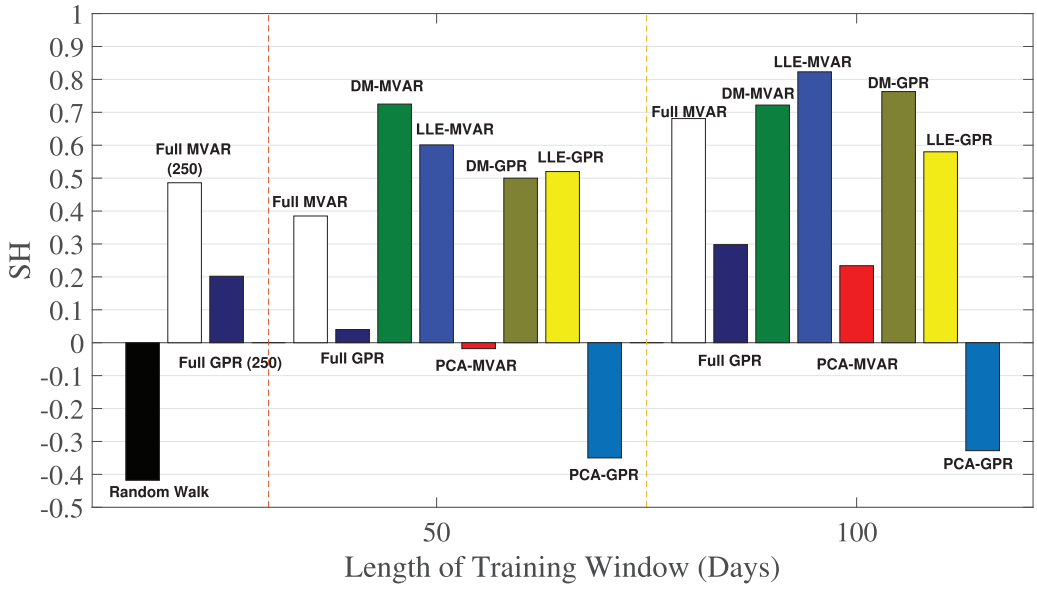

FIG. 3. FOREX trading. One-day-ahead predictions. Sharpe ratios obtained with the proposed framework (using DMs and LLE for embedding, MVAR and GPR for prediction at the embedded space, and GHs for lifting) as well as with PCA, random walk, and MVAR and GPR models implemented directly in the original space. A rolling window of 250 trading days and three vectors were used for the embedding, while the MVAR and GPR models in the embedded space were trained using the last 50 or 100 points of the rolling window.

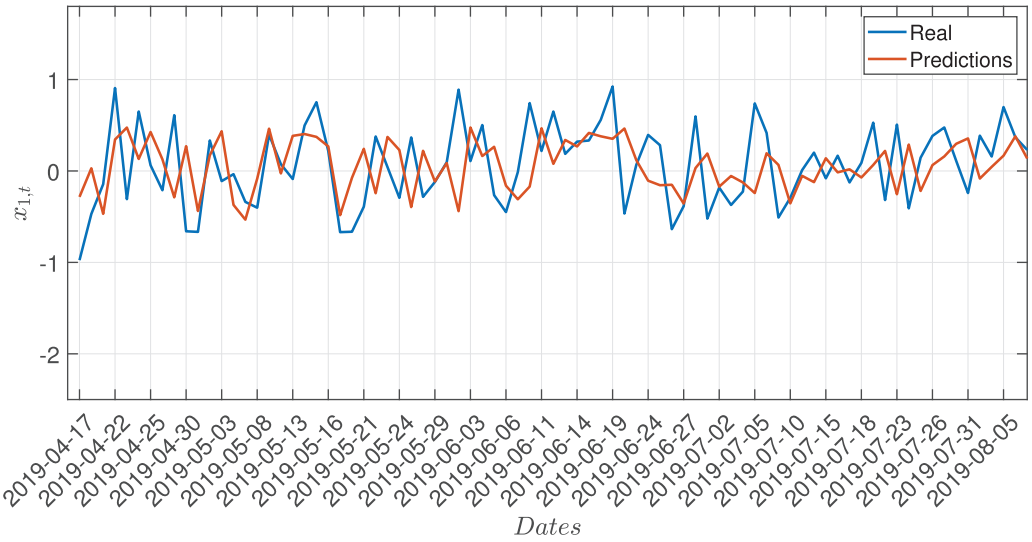

FIG. 4. Forecasting of the EURO/USD FOREX pair returns (solid blue line) using the LLE-MVAR(1)-GH scheme (red line) using a 100-day sliding training window.

highest SH ratios (∼0.83) are obtained with the combination of LLE and MVAR within the 100-day training window, followed by the combination of DMs and GPR in the 100-day training window, resulting in an SH ratio ∼0.76. The third (∼0.73) and fourth (∼0.72) larger SH ratios result again from the implementation of the DMs and MVAR within the 50- and 100-day training windows, respectively. On the other hand, the naïve random walk (black bar) resulted in a negative SH ratio (∼−0.42). Negative SH ratios (of the order of −0.35) resulted also from the implementation of PCA with GPR (yellow bars) for both sizes of the training window, while for the 50-day training window, the combination of PCA with MVAR resulted in an almost zero (but still negative) SH ratio. With PCA, a (small) positive SH value (∼0.23) was obtained with MVAR within the 100-day training window. The full MVAR and GPR models, trained and implemented directly in the original space, produced positive SH ratios, but still smaller than those of the DMs and LLE schemes. In particular, within the 50-day training window, the full MVAR (GPR) model resulted in an SH ratio of ∼0.39 (∼0.04), while within the 100-day training window, the full MVAR (GPR) model resulted in an SH ratio of ∼0.68 (∼0.3). Finally, we also tested the forecasting performance of the full MVAR and GPR models using the 250-day training window. Within this configuration, the MVAR model yielded an SH ratio of ∼0.49, while the GPR model an SH ratio of ∼0.2.

In Fig. 4, we depict indicatively the time series of the USD/EURO FOREX pair returns and the predictions using the full MVAR model (blue line), the LLE-MVAR(1)-GH scheme (red line) using a 100-day sliding training window.

VIII. CONCLUSIONS AND DISCUSSION

We proposed a data-driven numerical framework based on nonlinear manifold learning for forecasting high-dimensional time series, which is composed of three basic steps: (i) embedding of the original high-dimensional data in a low-dimensional manifold, (ii) construction of reduced-order surrogate models and forecasting on the manifold, and (iii) reconstruction of the predictions in the high-dimensional space by solving the pre-image problem.

The task of forecasting is different from that attempted for the reconstruction of high-dimensional models of dynamical systems based on interpolation in four main aspects. First, for real-world data sets, the existence of a relatively smooth low-dimensional manifold is not guaranteed as in the case of well-defined dynamical models (see, e.g., the discussion in Gajamannage *et al.*[46]). Second, non-stationary dynamics which, in general, are not an issue when dealing with the approximation of dynamical systems, pose a major challenge for a reliable/consistent forecasting. It should be noted that the stationarity assumption may be required to hold true even for interpolation problems. For example, the implementation of the MVAR models, but also Gaussian processes with a Gaussian kernel, requires the stationary covariance function assumption to be satisfied (see, e.g., the discussion in Rasmussen[90] and Cheng *et al.*[91]). Third, when dealing with real-world time series, such as financial time series, the number of available snapshots (even at the intraday frequency of trading) is limited in contrast to the size of temporal data that one can produce by model simulations. In such cases, the quest for beating the "curse of dimensionality" using the correct (parsimonious) embedding and modeling is stronger. Finally, in the case of smooth dynamical systems, as the dynamics emerges from the same physical laws as expressed by the underlying ODEs, PDEs, or SDEs, what is usually sought is a single global manifold (a single geometry). However, in many complex problems, such as those in finance, the best parameterization of the manifold (if it exists) may change over time. Thus, one would seek for a family of (sequential-in-time) submanifolds, which can be "identified" within a rolling window approach.

The performance of the scheme was assessed by implementing and comparing different combinations of nonlinear manifold learning algorithms (LLE and DMs), regression models (MVAR and GPR), and methods for solving the pre-image problem (geometric harmonics and radial basis functions) on various problems, including synthetic stochastic data series, the numerical propagation in time of the solution of linear and nonlinear parabolic PDEs, and a real-world data set of FOREX time series.

As discussed, the solution of the linear system associated with the RBF interpolation with Gaussian kernels may not be a good choice, as the system matrix (46) may be numerically rank deficient. In this case, one could compute the solution using, e.g., the Moore–Penrose pseudoinverse. Moreover, for large-scale almost-singular linear systems, one could use the GMRES method with preconditioning.[92–94] We intend to study the performance of such approaches in a future work.

At this point, we remark that there is no systematic way to choose which manifold learning technique to use (specifically, LLE or diffusion maps), and in practice, it often highly depends on the particular case. In the literature, one can find several empirical studies,[95] and the current understanding in the community is that diffusion maps may possibly perform better in modeling non-convex manifolds, data on the manifold that are non-uniformly sampled, and boundary effects. These empirical observations have been (partially) supported theoretically in a recent line of work.[96–98]

As a "restrict-run-lift" scheme, the proposed approach shares analogies with the equation-free approach and coarse-timestepping.[99–102] Furthermore, as we demonstrated for the case of the PDEs, the proposed scheme can be bridged with the concept of "gap-tooth"/"patch dynamics" schemes[103–105] for the numerical solution of high-dimensional PDEs; we will report on such a methodology in a future paper.

Finally, we note that in order to cope with the generalization property and the topological instability issues (see, e.g., Balasubramanian *et al.*[19]) that arise in implementing kernel-based manifold learning algorithms when the data set is not sufficiently dense on the manifold, and/or in the presence of "strong" stochasticity, one can resort to techniques, such as the constant-shift one to appropriately adjust the graph metric, and techniques for the removal of outliers from the data set and the construction of smooth geodesics.[46,50,106]

ACKNOWLEDGMENTS

This work was partially supported by the Italian program Fondo Integrativo Speciale per la Ricerca (FISR) (No. FISR2020IP-02893). The work of Ioannis G. Kevrekidis was partially supported by the U.S. Department of Energy and the U.S. Air Force Office of Scientific Research.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Panagiotis G. Papaioannou: Data curation (lead); Formal analysis (lead); Investigation (lead); Methodology (supporting); Software (lead); Validation (lead); Writing – original draft (supporting); Writing – review and editing (supporting). **Ronen Talmon:** Formal analysis (supporting); Methodology (supporting); Validation (equal); Writing – original draft (supporting); Writing – review and editing (equal). **Ioannis G. Kevrekidis:** Conceptualization (supporting); Methodology (supporting); Validation (supporting); Writing – original draft (supporting); Writing – review and editing (equal). **Constantinos Siettos:** Conceptualization (lead); Formal analysis (equal); Investigation (equal); Methodology (lead); Supervision (lead); Validation (equal); Writing – original draft (lead); Writing – review and editing (lead).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] C. W. J. Granger, "Investigating causal relations by econometric models and cross-spectral methods," Econometrica **37**, 424 (1969).
[2] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," Int. J. Forecast. **22**, 443–473 (2006).
[3] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," Comput. Sci. Rev. **3**, 127–149 (2009).
[4] F. Wyffels and B. Schrauwen, "A comparative study of reservoir computing strategies for monthly time series prediction," Neurocomputing **73**, 1958–1964 (2010).
[5] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain, "Gaussian processes for time-series modelling," Philos. Trans. Royal Soc. A: Math. Phys. Eng. Sci. **371**, 20110550 (2013).
[6] C. W. J. Granger and P. Newbold, *Forecasting Economic Time Series* (Academic Press, 2014).
[7] P. J. Brockwell, P. J. Brockwell, R. A. Davis, and R. A. Davis, *Introduction to Time Series and Forecasting* (Springer, 2016).
[8] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," IEEE Trans. Neural Netw. Learn. Syst. **28**, 2222–2232 (2016).
[9] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," Phys. Rev. Lett. **120**, 024102 (2018).
[10] R. S. Zimmermann and U. Parlitz, "Observing spatio-temporal dynamics of excitable media using reservoir computing," Chaos **28**, 043118 (2018).
[11] P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," Neural Netw. **126**, 191–217 (2020).
[12] R. Bellmann, *Dynamic Programming* (Princeton University Press, 1957), Vol. 8.
[13] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," J. Fluid Mech. **656**, 5–28 (2010).
[14] J. Mann and J. N. Kutz, "Dynamic mode decomposition for financial trading strategies," Quant. Finance **16**, 1643–1655 (2016).
[15] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems* (SIAM, 2016).
[16] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," Neural Comput. **10**, 1299–1319 (1998).
[17] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," Science **290**, 2323–2326 (2000).
[18] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," Science **290**, 2319–2323 (2000).

[19]M. Balasubramanian, E. L. Schwartz, J. B. Tenenbaum, V. de Silva, and J. C. Langford, "The Isomap algorithm and topological stability," Science **295**, 7 (2002).
[20]M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," Neural Comput. **15**, 1373–1396 (2003).
[21]R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps," Proc. Natl. Acad. Sci. U.S.A. **102**, 7426–7431 (2005).
[22]R. R. Coifman and S. Lafon, "Diffusion maps," Appl. Comput. Harmon. Anal. **21**, 5–30 (2006).
[23]R. R. Coifman and S. Lafon, "Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions," Appl. Comput. Harmon. Anal. **21**, 31–52 (2006).
[24]B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis, "Diffusion maps, spectral clustering and reaction coordinates of dynamical systems," Appl. Comput. Harmon. Anal. **21**, 113–127 (2006).
[25]R. R. Coifman, I. G. Kevrekidis, S. Lafon, M. Maggioni, and B. Nadler, "Diffusion maps, reduction coordinates, and low dimensional representation of stochastic systems," Multiscale Model. Simul. **7**, 842–864 (2008).
[26]I. Mezić, "Analysis of fluid flows via spectral properties of the Koopman operator," Annu. Rev. Fluid Mech. **45**, 357–378 (2013).
[27]M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition," J. Nonlinear Sci. **25**, 1307–1346 (2015).
[28]F. Dietrich, T. N. Thiem, and I. G. Kevrekidis, "On the Koopman operator of algorithms," SIAM J. Appl. Dyn. Syst. **19**, 860–885 (2020).
[29]E. Bollt, "Attractor modeling and empirical nonlinear model reduction of dissipative dynamical systems," Int. J. Bifurcation Chaos **17**, 1199–1219 (2007).
[30]E. Chiavazzo, C. W. Gear, C. J. Dsilva, N. Rabin, and I. G. Kevrekidis, "Reduced models in chemical kinetics via nonlinear data-mining," Processes **2**, 112–140 (2014).
[31]P. Liu, H. R. Safford, I. D. Couzin, and I. G. Kevrekidis, "Coarse-grained variables for particle-based models: Diffusion maps and animal swarming simulations," Comput. Part. Mech. **1**, 425–440 (2014).
[32]C. J. Dsilva, R. Talmon, C. W. Gear, R. R. Coifman, and I. G. Kevrekidis, "Data-driven reduction for multiscale stochastic dynamical systems," arXiv:1501.05195 (2015).
[33]S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," Proc. Natl. Acad. Sci. U.S.A. **113**, 3932–3937 (2016).
[34]S. Bhattacharjee and K. Matouš, "A nonlinear manifold-based reduced order model for multiscale analysis of heterogeneous hyperelastic materials," J. Comput. Phys. **313**, 635–653 (2016).
[35]Z. Y. Wan and T. P. Sapsis, "Reduced-space Gaussian process regression for data-driven probabilistic forecast of chaotic dynamical systems," Phys. D **345**, 40–55 (2017).
[36]J. Nathan Kutz, J. L. Proctor, and S. L. Brunton, "Applied Koopman theory for partial differential equations and data-driven modeling of spatio-temporal systems," Complexity **2018**, 6010634.
[37]W. Chen and A. L. Ferguson, "Molecular enhanced sampling with autoencoders: On-the-fly collective variable discovery and accelerated free energy landscape exploration," J. Comput. Chem. **39**, 2079–2102 (2018).
[38]M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," AIChE J. **37**, 233–243 (1991).
[39]P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos, "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks," Proc. R. Soc. A **474**, 20170844 (2018).
[40]F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980* (Springer, 1981), pp. 366–381.
[41]S. Herzog, F. Wörgötter, and U. Parlitz, "Convolutional autoencoder and conditional random fields hybrid for predicting spatial-temporal chaos," Chaos **29**, 123116 (2019).
[42]S. Lee, M. Kooshkbaghi, K. Spiliotis, C. I. Siettos, and I. G. Kevrekidis, "Coarse-scale PDEs from fine-scale observations via machine learning," Chaos **30**, 013141 (2020).
[43]E. Koronaki, A. Nikas, and A. Boudouvis, "A data-driven reduced-order model of nonlinear processes based on diffusion maps and artificial neural networks," Chem. Eng. J. **397**, 125475 (2020).
[44]J. Isensee, G. Datseris, and U. Parlitz, "Predicting spatio-temporal time series using dimension reduced local states," J. Nonlinear Sci. **30**, 713–735 (2020).
[45]K. K. Lin and F. Lu, "Data-driven model reduction, Wiener projections, and the Koopman-Mori-Zwanzig formalism," J. Comput. Phys. **424**, 109864 (2021).
[46]K. Gajamannage, R. Paffenroth, and E. M. Bollt, "A nonlinear dimensionality reduction framework using smooth geodesics," Pattern Recognit. **87**, 226–236 (2019).
[47]J. M. Lee, "Smooth manifolds," in *Introduction to Smooth Manifolds* (Springer, 2013), pp. 1–31.
[48]M. Berger and B. Gostiaux, *Differential Geometry: Manifolds, Curves, and Surfaces* (Springer Science & Business Media, 2012), Vol. 115.
[49]W. Kühnel, *Differential Geometry* (American Mathematical Society, 2015), Vol. 77.
[50]J. Wang, *Geometric Structure of High-Dimensional Data and Dimensionality Reduction* (Springer, 2012), Vol. 5.
[51]L. K. Saul and S. T. Roweis, "Think globally, fit locally: Unsupervised learning of low dimensional manifolds," J. Mach. Learn. Res. **4**, 119–155 (2003).
[52]C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," Psychometrika **1**, 211–218 (1936).
[53]L. Mirsky, "Symmetric gauge functions and unitarily invariant norms," Q. J. Math. **11**, 50–59 (1960).
[54]P. W. Jones, M. Maggioni, and R. Schul, "Manifold parametrizations by eigenfunctions of the Laplacian and heat kernels," Proc. Natl. Acad. Sci. U.S.A. **105**, 1803–1808 (2008).
[55]M. Marcellino, J. H. Stock, and M. W. Watson, "A comparison of direct and iterated multistep AR methods for forecasting macroeconomic time series," J. Econom. **135**, 499–526 (2006).
[56]C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (MIT Press, 2018).
[57]G. Corani, A. Benavoli, and M. Zaffalon, "Time series forecasting with Gaussian processes needs priors," arXiv:2009.08102 (2021).
[58]E. J. Nyström, *Über die Praktische Auflösung von Linearen Integralgleichungen mit Anwendungen auf Randwertaufgaben der Potentialtheorie* (Akademische Buchhandlung, 1929).
[59]N. D. Monnig, B. Fornberg, and F. G. Meyer, "Inverting nonlinear dimensionality reduction with scale-free radial basis function interpolation," Appl. Comput. Harmon. Anal. **37**, 162–170 (2014).
[60]B. Fornberg and J. Zuev, "The runge phenomenon and spatially variable shape parameters in RBF interpolation," Comput. Math. Appl. **54**, 379–398 (2007).
[61]E. Amorim, E. V. Brazil, J. Mena-Chalco, L. Velho, L. G. Nonato, F. Samavati, and M. C. Sousa, "Facing the high-dimensions: Inverse projection with radial basis functions," Comput. Graph. **48**, 35–47 (2015).
[62]L. M. Delves and J. L. Mohamed, *Computational Methods for Integral Equations* (CUP Archive, 1988).
[63]W. H. Press and S. A. Teukolsky, "Fredholm and Volterra integral equations of the second kind," Comput. Phys. **4**, 554–557 (1990).
[64]T. N. Thiem, M. Kooshkbaghi, T. Bertalan, C. R. Laing, and I. G. Kevrekidis, "Emergent spaces for coupled oscillators," Front. Comput. Neurosci. **14**, 36 (2020).
[65]N. Rabin, Y. Bregman, O. Lindenbaum, Y. Ben-Horin, and A. Averbuch, "Earthquake-explosion discrimination using diffusion maps," Geophys. J. Int. **207**, 1484–1492 (2016).
[66]I. Prigogine and R. Lefever, "Symmetry breaking instabilities in dissipative systems. II," J. Chem. Phys. **48**, 1695–1700 (1968).
[67]L. A. Baccalá and K. Sameshima, "Partial directed coherence: A new concept in neural structure determination," Biol. Cybern. **84**, 463–474 (2001).

[68]N. Nicolaou and T. G. Constandinou, "A nonlinear causality estimator based on non-parametric multiplicative regression," Front. Neuroinform. **10**, 19 (2016).
[69]P. G. Kevrekidis, C. I. Siettos, and Y. G. Kevrekidis, "To infinity and some glimpses of beyond," Nat. Commun. **8**, 1562 (2017).
[70]L. F. Shampine and M. W. Reichelt, "The MATLAB ODE suite," SIAM J. Sci. Comput. **18**, 1–22 (1997).
[71]A. B. del Canto, "investpy—Financial Data Extraction from Investing.com with Python," https://investpy.readthedocs.io/ (2020).
[72]L. Menkhoff, L. Sarno, M. Schmeling, and A. Schrimpf, "Carry trades and global foreign exchange volatility," J. Finance **67**, 681–718 (2012).
[73]W. McKinney, "Data structures for statistical computing in Python," in *Proceedings of the 9th Python in Science Conference*, Vol. 445 (Austin, TX, 2010), pp. 51–56.
[74]M. D. Braga, "Risk parity versus other μ-free strategies: A comparison in a triple view," Invest. Manag. Financ. Innov. **12**, 277–289 (2015).
[75]D. Lehmberg, F. Dietrich, G. Köster, and H.-J. Bungartz, "Datafold: Data-driven models for point clouds and time series on manifolds," J. Open Source Softw. **5**, 2283 (2020).
[76]F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res. **12**, 2825–2830 (2011).
[77]S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with Python," in *Proceedings of the 9th Python in Science Conference*, Vol. 57 (Austin, TX, 2010), p. 61.
[78]R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods* (SIAM, 1998).
[79]P. Virtanen, R. Gommers, T. E. Oliphant *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," Nat. Methods **17**, 261 (2020).
[80]J. J. M. Cuppen, "A divide and conquer method for the symmetric tridiagonal eigenproblem," Numer. Math. **36**, 177 (1980).
[81]E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. (SIAM, Philadelphia, PA, 1999).
[82]C. J. Dsilva, R. Talmon, R. R. Coifman, and I. G. Kevrekidis, "Parsimonious representation of nonlinear dynamical systems through manifold learning: A chemotaxis case study," Appl. Comput. Harmon. Anal. **44**, 759–773 (2018).
[83]A. Singer, R. Erban, I. G. Kevrekidis, and R. R. Coifman, "Detecting intrinsic slow variables in stochastic dynamical systems by anisotropic diffusion maps," Proc. Natl. Acad. Sci. U.S.A. **106**, 16090–16095 (2009).
[84]R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," SIAM J. Sci. Comput. **16**, 1190 (1995).
[85]C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," ACM Trans. Math. Softw. **23**, 550 (1997).
[86]S. Maneewongvatana and D. M. Mount, "Analysis of approximate nearest neighbor searching with clustered point sets," in *Data Structures, Near Neighbor Searches, and Methodology* (American Mathematical Society, 2002), Vol. 59, pp. 105–123.
[87]N. Evangelou, F. Dietrich, E. Chiavazzo, D. Lehmberg, M. Meila, and I. G. Kevrekidis, "Double diffusion maps and their latent harmonics for scientific computation in latent space," arXiv:2204.12536 (2022).
[88]W. F. Sharpe, "The sharpe ratio," J. Portf. Manag. **21**, 49 (1994).
[89]P. Papaioannou, L. Russo, G. Papaioannou, and C. I. Siettos, "Can social microblogging be used to forecast intraday exchange rates?," NETNOMICS: Econ. Res. Electron. Netw. **14**, 47 (2013).
[90]C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning* (Springer, 2003), pp. 63–71.
[91]C. Cheng, A. Sa-Ngasoongsong, O. Beyca, T. Le, H. Yang, Z. Kong, and S. T. Bukkapatnam, "Time series forecasting for nonlinear and non-stationary processes: A review and comparative study," IIE Trans. **47**, 1053–1071 (2015).
[92]Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm," SIAM J. Sci. Comput. **14**, 461–469 (1993).
[93]R. K. Beatson, J. B. Cherrie, and C. T. Mouat, "Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration," Adv. Comput. Math. **11**, 253–270 (1999).
[94]L. Eldén and V. Simoncini, "Solving ill-posed linear systems with GMRES and a singular preconditioner," SIAM J. Matrix Anal. Appl. **33**, 1369–1394 (2012).
[95]A. Schwartz and R. Talmon, "Intrinsic isometric manifold learning with application to localization," SIAM J. Imaging Sci. **12**, 1347–1391 (2019).
[96]H.-T. Wu and N. Wu, "Think globally, fit locally under the manifold setup: Asymptotic analysis of locally linear embedding," Ann. Stat. **46**, 3805–3837 (2018).
[97]H.-T. Wu and N. Wu, "When locally linear embedding hits boundary," arXiv:1811.04423 (2018).
[98]J. Malik, C. Shen, H.-T. Wu, and N. Wu, "Connecting dots: From local covariance to empirical intrinsic geometry and locally linear embedding," Pure Appl. Anal. **1**, 515–542 (2019).
[99]C. Theodoropoulos, Y.-H. Qian, and I. G. Kevrekidis, ""Coarse" stability and bifurcation analysis using time-steppers: A reaction-diffusion example," Proc. Natl. Acad. Sci. U.S.A. **97**, 9840–9843 (2000).
[100]I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, C. Theodoropoulos *et al.*, "Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level analysis," Commun. Math. Sci. **1**, 715–762 (2003).
[101]C. I. Siettos, M. D. Graham, and I. G. Kevrekidis, "Coarse Brownian dynamics for nematic liquid crystals: Bifurcation, projective integration, and control via stochastic simulation," J. Chem. Phys. **118**, 10149–10156 (2003).
[102]I. G. Kevrekidis, C. W. Gear, and G. Hummer, "Equation-free: The computer-aided analysis of complex multiscale systems," AIChE J. **50**, 1346–1355 (2004).
[103]C. W. Gear, J. Li, and I. G. Kevrekidis, "The gap-tooth method in particle simulations," Phys. Lett. A **316**, 190–195 (2003).
[104]G. Samaey, D. Roose, and I. G. Kevrekidis, "The gap-tooth scheme for homogenization problems," Multiscale Model. Simul. **4**, 278–306 (2005).
[105]G. Samaey, I. G. Kevrekidis, and D. Roose, "Patch dynamics with buffers for homogenization problems," J. Comput. Phys. **213**, 264–287 (2006).
[106]H. Choi and S. Choi, "Robust kernel Isomap," Pattern Recognit. **40**, 853–862 (2007).