Online learning of quadratic manifolds from streaming data for nonlinear dimensionality reduction and nonlinear model reduction

Paul Schwerdtner* Prakash Mohan[†] Aleksandra Pachalieva[‡] Julie Bessac[†] Daniel O'Malley[‡] Benjamin Peherstorfer*

September 2024

This work introduces an online greedy method for constructing quadratic manifolds from streaming data, designed to enable in-situ analysis of numerical simulation data on the Petabyte scale. Unlike traditional batch methods, which require all data to be available upfront and take multiple passes over the data, the proposed online greedy method incrementally updates quadratic manifolds in one pass as data points are received, eliminating the need for expensive disk input/output operations as well as storing and loading data points once they have been processed. A range of numerical examples demonstrate that the online greedy method learns accurate quadratic manifold embeddings while being capable of processing data that far exceed common disk input/output capabilities and volumes as well as main-memory sizes.

 $\textbf{Keywords:} \ \ \text{model reduction, surrogate modeling, quadratic manifolds, closure modeling}$

1 Introduction

Dimensionality reduction is an important building block in computational science and engineering, including model reduction [49, 10, 1, 39, 42, 36, 21] and closure modeling [52, 57, 20, 19, 58, 25, 60, 41]. We focus on nonlinear dimensionality reduction with quadratic manifolds, which can be interpreted as nonlinearly correcting linear approximations obtained with principal components [24, 6, 17, 55].

^{*}Courant Institute of Mathematical Sciences, New York University, New York (paul.schwerdtner@nyu.edu)

[†]Computational Science Center, National Renewable Energy Laboratory, Golden, Colorado

[‡]Los Alamos National Laboratory, Los Alamos, New Mexico

The motivation for focusing on quadratic approximations is that quadratic terms arise in the modeling of many physics phenomena [34, 51, 8, 45, 47, 7, 28, 54] and also have been shown to achieve accurate approximations of latent dynamics in model reduction [22, 6, 26, 9, 56, 59, 48, 35, 53, 27].

Several methods for learning quadratic manifolds from data have been introduced [24, 6, 22, 23, 55]. All of these methods operate in the batch setting, which means that all data points are available at the beginning of the computational process and the algorithms make multiple passes over the data points. However, in many science and engineering applications, loading all data points into the main memory, or even storing them on disk, is intractable. In this work, we introduce an online learning method that incrementally constructs quadratic manifolds from streaming data in one pass, which means that data points are processed immediately as they are received one by one or in small chunks. This avoids multiple passes over the data and having to store the whole data set to disk or even loading all data points into the main memory at once. We first demonstrate that the online learning method constructs quadratic manifolds that can accurately approximate the data. We then leverage the online capabilities of our approach to use it as an in-situ data analysis approach that processes data as they are generated from numerical simulations without having to store data on disk. The online learning capabilities allows us to train quadratic manifolds on Petabyte-scale data, which is impractical with batch methods due to memory and storage limitations as well as high costs of disk input/output operations.

We briefly review the literature on constructing quadratic manifolds. We distinguish between three different batch methods for constructing quadratic manifolds from data. First, there is the approach introduced in [24, 6] that computes the singular value decomposition (SVD) of the data matrix and subsequently fits a weight matrix to the difference between the data points and the reduced data points in the subspace spanned by the leading left-singular values. As recognized in [22, 23, 55], relying on the leading left-singular vectors alone can lead to poor approximation capabilities of the learned quadratic manifolds. The second approach for constructing quadratic manifolds is via an alternating-minimization problem introduced in [22, 23]. The approach solves a non-convex optimization problems, which leads to quadratic manifolds that can more accurately approximate the data points but at the same time the computational costs of the training process can be high due to the non-convex optimization problem. The third approach is the method introduced in [55], which greedily constructs quadratic manifolds. It extends the approach based on the SVD and the linear least-squares problem from [24, 6] to compute multiple linear least-squares problems to better fit the quadratic manifolds. All of the three approaches, however, assume that all data points are available at the beginning of the process and that the algorithms can make multiple passes over the data points.

Besides batch methods for constructing quadratic manifolds, an important building block for us will be methods for incrementally constructing the SVD [2, 15, 3, 13, 12, 18, 37] and dominated subspaces [5, 4, 44] from streaming data. There also are randomized methods [16, 40, 29] that can learn principal components [33, 32] from streaming data. Incrementally constructing the SVD has been a key building block for model reduction

from streaming data [43, 46] and streaming dynamic mode decomposition [30, 61, 38].

In this work, we introduce an online greedy method for constructing quadratic manifolds that builds on [55] but is applicable when data points are streamed and have to be processed in one pass without storing them for subsequent post-processing. We first show that the batch greedy method from [55] can act directly on the left- and right-singular vectors as well as the singular values of the data, without having to explicitly assemble the data matrix. We then argue that the greedy method remains valid even on a truncated SVD of the data matrix, which we then propose to compute incrementally from streaming data. For the incremental SVD, we build on the algorithm introduced in [2]. Numerical experiments with data from wave propagation problems and computational fluid dynamics demonstrate that the online greedy method can handle Petabyte-scale data, can be used as in-situ data analysis tool within numerical simulations, and constructs quadratic manifolds that accurately approximate simulation data.

This manuscript is organized as follows. We provide preliminaries and a problem formulation in Section 2. The online greedy method is introduced in Section 3, together with an algorithmic description. Numerical experiments are shown in Section 4. We draw conclusions in Section 5.

2 Preliminaries

2.1 Linear dimensionality reduction with encoder and decoder functions

Dimensionality reduction typically can be formulated via an encoder $f: \mathbb{R}^N \to \mathbb{R}^n$ that maps a high-dimensional data point $s \in \mathbb{R}^N$ of dimension N to a reduced point \hat{s} of low dimension $n \ll N$, and a decoder $g: \mathbb{R}^n \to \mathbb{R}^N$ that lifts a reduced data point $\hat{s} \in \mathbb{R}^n$ back into the high-dimensional space \mathbb{R}^N . The singular value decomposition (SVD) provides a classical approach to dimensionality reduction with linear encoder and decoder functions. Let $s_1, \ldots, s_M \in \mathbb{R}^N$ be data samples that are collected into the data matrix $S = [s_1, \ldots, s_M]$. The SVD of S gives the decomposition $S = \Phi \Sigma \Psi^\top$, with the orthonormal matrices $\Phi \in \mathbb{R}^{N \times r_{\text{max}}}, \Psi \in \mathbb{R}^{M \times r_{\text{max}}}$ and diagonal matrix $\Sigma \in \mathbb{R}^{r_{\text{max}} \times r_{\text{max}}}$, with the rank r_{max} of the matrix S. The singular values, i.e., the diagonal elements of Σ , are denoted as $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{r_{\text{max}}} \geq 0$. Selecting the first n left-singular vectors ϕ_1, \ldots, ϕ_n of S into the basis matrix $V = [\phi_1, \ldots, \phi_n] \in \mathbb{R}^{N \times n}$ leads to the linear encoder function $f_V(s) = V^\top s$ and linear decoder function $g_V(\hat{s}) = V\hat{s}$. The subspace that is spanned by the columns of V is denoted as $V \subset \mathbb{R}^N$. The encoding and subsequent decoding $g_V \circ f_V$ of a data points s corresponds to the orthogonal projection of s with respect to the Euclidean inner product onto the subspace V. We denote the projection onto V as $P_V : \mathbb{R}^N \to V$, $s \mapsto VV^\top s$.

2.2 Dimensionality reduction with quadratic manifolds

For nonlinear dimensionality reduction, the linear decoder function $g_{\mathbf{V}}$ can be augmented with a nonlinear correction term as

$$g_{\mathbf{V},\mathbf{W}}(\hat{\mathbf{s}}) = \mathbf{V}\hat{\mathbf{s}} + \mathbf{W}h(\hat{\mathbf{s}}), \tag{1}$$

where $h: \mathbb{R}^n \to \mathbb{R}^p$ is a nonlinear feature map that provides $p \gg n$ features for a reduced data point \hat{s} and $\mathbf{W} \in \mathbb{R}^{N \times p}$ is a weight matrix; see [34, 51, 24, 6, 23, 22, 55]. Given a basis matrix \mathbf{V} of a subspace \mathcal{V} , a weight matrix \mathbf{W} , and a feature map h, the linear encoder $f_{\mathbf{V}}$ and the nonlinear decoder $g_{\mathbf{V},\mathbf{W}}$ induce a manifold

$$\mathcal{M}_n(\mathbf{V}, \mathbf{W}, h) = \{g_{\mathbf{V}, \mathbf{W}}(\hat{\mathbf{s}}) \mid \hat{\mathbf{s}} \in \mathbb{R}^n\} \subset \mathbb{R}^N.$$

Because h is nonlinear, the manifold \mathcal{M}_n can include points in \mathbb{R}^N that are outside of the subspace \mathcal{V} . In particular, if h is a quadratic polynomial

$$h_{\text{quad}}: \mathbb{R}^n \to \mathbb{R}^{n(n+1)/2}, \boldsymbol{x} \mapsto \begin{bmatrix} x_1 x_1 & x_1 x_2 & \dots & x_1 x_n & x_2 x_2 & \dots & x_n x_n \end{bmatrix}^\top,$$
 (2)

then we call the corresponding manifold \mathcal{M}_n a quadratic manifold. Quadratic manifolds are widely used; see, e.g., [34, 51, 24, 6, 23, 22, 55].

References [24, 6] propose setting the matrix V to have the leading n left-singular vectors ϕ_1, \ldots, ϕ_n as columns and constructing W via a linear regression problem as

$$\min_{\boldsymbol{W} \in \mathbb{R}^{N \times p}} \| \boldsymbol{P}_{\mathcal{V}} \boldsymbol{S} - \boldsymbol{S} + \boldsymbol{W} h(f_{\boldsymbol{V}}(\boldsymbol{S})) \|_F^2 + \gamma \| \boldsymbol{W} \|_F^2,$$
 (3)

where $\|\cdot\|_F$ denotes the Frobenius norm. We overload the notation of h in (3) to allow h to be evaluated column-wise on the matrix $f_{\mathbf{V}}(\mathbf{S}) = \mathbf{V}^{\top} \mathbf{S}$ to obtain $h(f_{\mathbf{V}}(\mathbf{S})) \in \mathbb{R}^{p \times M}$. The regularization term is controlled by $\gamma > 0$ and can prevent overfitting of the weight matrix \mathbf{W} to data in \mathbf{S} .

2.3 Greedy construction of quadratic manifolds from data

Selecting the n leading left-singular vectors ϕ_1, \ldots, ϕ_n as columns of V that span \mathcal{V} can lead to poor results when constructing quadratic manifolds with the optimization problem (3): Notice that the nonlinear correction term in the decoder (1) acts on the projection of the data point onto the first leading left-singular vectors that can miss information that are necessary for the quadratic term to be effective, which is discussed in detail in [55]. This is why the work [55] proposes a greedy method for selecting n left-singular vectors $\phi_{j_1}, \ldots, \phi_{j_n}$ with indices $j_1, j_2, \ldots, j_n \in \{1, \ldots, q\}$ from the first $q \gg n$ left-singular vectors ϕ_1, \ldots, ϕ_q . Importantly, the selected left-singular vectors $\phi_{j_1}, \ldots, \phi_{j_n}$ with indices j_1, \ldots, j_n are not necessarily the leading n left-singular vectors with indices j_1, \ldots, j_n .

At each greedy iteration i = 1, ..., n, the index j_i is determined as a minimizer of

$$\min_{j_i=1,\dots,m} \min_{\boldsymbol{W} \in \mathbb{R}^{N \times p}} J(\boldsymbol{\phi}_{j_i}, \boldsymbol{V}_{i-1}, \boldsymbol{W}). \tag{4}$$

The matrix $\boldsymbol{V}_{i-1} = [\boldsymbol{\phi}_{j_1}, \dots, \boldsymbol{\phi}_{j_{i-1}}]$ contains as columns the left-singular vectors $\boldsymbol{\phi}_{j_1}, \dots, \boldsymbol{\phi}_{j_{i-1}}$ with indices j_1, \dots, j_{i-1} that were selected in the previous greedy iterations $1, \dots, i-1$. The objective function is

$$J(\boldsymbol{v}, \boldsymbol{V}, \boldsymbol{W}) = \|\boldsymbol{P}_{\mathcal{V} \oplus \text{span}\{\boldsymbol{v}\}} \boldsymbol{S} - \boldsymbol{S} + \boldsymbol{W} h(f_{[\boldsymbol{V}, \boldsymbol{v}]}(\boldsymbol{S}))\|_F^2 + \gamma \|\boldsymbol{W}\|_F^2,$$
 (5)

where $P_{\mathcal{V}\oplus \text{span}\{v\}}$ denotes the orthogonal projection operator onto the subspace $\mathcal{V}\oplus \text{span}(v)$ spanned by the columns of V and the vector v. After n iterations, the indices j_1,\ldots,j_n are selected and the matrix $V=[\phi_{j_1},\ldots,\phi_{j_n}]$ is assembled. The weight matrix W is then fitted via (3).

2.4 Problem formulation

Current methods [24, 6, 55] for constructing quadratic manifolds operate under the assumption that the data points s_1, \ldots, s_M are available all at once at the beginning of the computations, which is typically referred to as batch setting. Furthermore, the algorithms are allowed to make multiple passes over the data samples s_1, \ldots, s_M . In contrast, we are interested in online learning of quadratic manifolds, which means that data points are streamed and have to be processed in one pass as they are received.

Formally, our setup is that at each iteration $k = 0, 1, 2, 3, \ldots$, we have access to a chunk $\mathbf{S}_k = [\mathbf{s}_{km+1}, \ldots, \mathbf{s}_{(k+1)m}]$ of $m \ll M$ data samples. In particular, we have access to \mathbf{S}_k only at iteration k and thus the chunk \mathbf{S}_k has to be processed immediately because it cannot be stored and processed at later iterations.

3 Online learning of quadratic manifolds

We introduce an online greedy approach that constructs quadratic manifolds from streaming data. The key insight is that all steps of the batch greedy method presented in [55] can be reformulated to directly operate on the factors of the SVD of the data matrix rather than on the data points themselves. Thus, it is sufficient to have access to the SVD of the streamed data samples, instead of requiring direct access to the data samples. Building on this insight, we incrementally update a truncated SVD of the data matrix, to which we then apply the reformulated greedy method to construct a quadratic manifold.

3.1 Greedy construction from the SVD of the data matrix

Recall that $S = \Phi \Sigma \Psi^{\top}$ is the SVD of the data matrix S. We now show that the greedy iterations described in Section 2.3 can be performed with the SVD of S given by the matrices Φ, Σ, Ψ , without having to assemble the data matrix S.

At greedy iteration i = 1, ..., n, we define the index sets $\mathcal{I} = \{j_1, ..., j_i\}$ and $\check{\mathcal{I}} = \{1, ..., \min(N, M)\} \setminus \mathcal{I}$, where $j_1, ..., j_i$ are the indices of the left-singular vectors that have been selected by the greedy algorithm up to iteration i. Based on the index sets \mathcal{I} and $\check{\mathcal{I}}$, we permute the SVD as

$$S = \mathbf{\Phi} \mathbf{\Sigma} \mathbf{\Psi}^{\top} = \begin{bmatrix} \mathbf{\Phi}_{\mathcal{I}} & \mathbf{\Phi}_{\check{\mathcal{I}}} \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_{\mathcal{I}} & 0 \\ 0 & \mathbf{\Sigma}_{\check{\mathcal{I}}} \end{bmatrix} \begin{bmatrix} \mathbf{\Psi}_{\mathcal{I}} & \mathbf{\Psi}_{\check{\mathcal{I}}} \end{bmatrix}^{\top}, \tag{6}$$

where $\Phi_{\mathcal{I}} = [\phi_{j_1}, \dots, \phi_{j_i}]$, $\Sigma_{\mathcal{I}}$ contains $\sigma_{j_1}, \dots, \sigma_{j_i}$ on its diagonal, and $\Psi_{\mathcal{I}} = [\psi_{j_1}, \dots, \psi_{j_i}]$. Analogously, the matrices $\Phi_{\check{\mathcal{I}}}, \Psi_{\check{\mathcal{I}}}$, and $\Sigma_{\check{\mathcal{I}}}$ contain the singular vectors and singular values with indices in $\check{\mathcal{I}}$. Let now $V_{\mathcal{I}} = [\phi_{j_1}, \dots, \phi_{j_i}]$ be a basis matrix that spans the

subspace $\mathcal{V}_{\mathcal{I}} \subset \mathbb{R}^N$. We can then write the orthogonal projection of S onto $\mathcal{V}_{\mathcal{I}}$ as

$$oldsymbol{P}_{\mathcal{V}_{\mathcal{I}}} oldsymbol{S} = oldsymbol{\Phi}_{\mathcal{I}} oldsymbol{\Sigma}_{\mathcal{I}} oldsymbol{\Psi}_{\mathcal{I}}^{ op}$$
 .

Furthermore, because the left-singular vectors are orthonormal, we obtain that the projection error of S onto $\mathcal{V}_{\mathcal{I}}$ is

$$P_{\mathcal{V}_{\mathcal{I}}}S - S = \Phi_{\check{\mathcal{I}}}\Sigma_{\check{\mathcal{I}}}\Psi_{\check{\mathcal{T}}}^{\top}. \tag{7}$$

Finally, we can represent the reduced data points of S given by the encoder $f_{\mathcal{V}_{\mathcal{I}}}$ as $f_{\mathcal{V}_{\mathcal{I}}}(S) = \Sigma_{\mathcal{I}} \Psi_{\mathcal{I}}$.

Equipped with these reformulations, we can state the the least-squares problem (3) using only factors of the SVD of S as

$$\min_{\boldsymbol{W} \in \mathbb{R}^{N \times p}} \|\boldsymbol{\Phi}_{\boldsymbol{\mathcal{I}}} \boldsymbol{\Sigma}_{\boldsymbol{\mathcal{I}}} \boldsymbol{\Psi}_{\boldsymbol{\mathcal{I}}}^{\top} + \boldsymbol{W} h(\boldsymbol{\Sigma}_{\boldsymbol{\mathcal{I}}} \boldsymbol{\Psi}_{\boldsymbol{\mathcal{I}}}^{\top}) \|_F^2 + \gamma \|\boldsymbol{W}\|_F^2.$$
 (8)

Similarly, because the greedy iterations evaluate the objective function (5) only at left-singular vectors, we can represent (5) as

$$J'(j, \boldsymbol{V}_{\mathcal{I}}, \boldsymbol{W}) = \|\boldsymbol{\Phi}_{\boldsymbol{\check{\mathcal{I}}}\setminus\{j\}} \boldsymbol{\Sigma}_{\boldsymbol{\check{\mathcal{I}}}\setminus\{j\}} \boldsymbol{\Psi}_{\boldsymbol{\check{\mathcal{I}}}\setminus\{j\}}^{\top} + \boldsymbol{W}h(\boldsymbol{\Sigma}_{\mathcal{I}\cup\{j\}} \boldsymbol{\Psi}_{\mathcal{I}\cup\{j\}}^{\top}) \|_F^2 + \gamma \|\boldsymbol{W}\|_F^2, \quad (9)$$

where the first argument is now the index j of the left-singular vector $\boldsymbol{\phi}_j$ instead of a generic vector \boldsymbol{v} as in (5). Notice that $\boldsymbol{\mathcal{I}} \setminus \{j\}$ is the set $\boldsymbol{\mathcal{I}}$ with the element j removed. Because each greedy iteration just minimizes (5) over the left-singular vectors $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_q$ according to the optimization problem given in (4), the objective J in (5) can be replaced with the objective J' given in (9) that uses the factors $\boldsymbol{\Phi}, \boldsymbol{\Sigma}, \boldsymbol{\Psi}$ of the SVD of \boldsymbol{S} only.

3.2 Greedy construction from a truncated SVD of the data matrix

Let us now consider a truncated SVD of the data matrix. Recall the full SVD in (6) that is permuted based on the index sets \mathcal{I} and $\check{\mathcal{I}}$. We introduce a third index set that we denote as $\hat{\mathcal{I}}$ and that we define as $\hat{\mathcal{I}} = \{1, \ldots, q\} \setminus \mathcal{I}$. The three index sets $\mathcal{I}, \hat{\mathcal{I}}$, and $\check{\mathcal{I}} \setminus \hat{\mathcal{I}}$ lead to the SVD of S as

$$S = \mathbf{\Phi} \mathbf{\Sigma} \mathbf{\Psi}^{\top} = \begin{bmatrix} \mathbf{\Phi}_{\mathcal{I}} & \mathbf{\Phi}_{\hat{\mathcal{I}}} & \mathbf{\Phi}_{\check{\mathcal{I}} \setminus \hat{\mathcal{I}}} \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_{\mathcal{I}} & 0 & 0 \\ 0 & \mathbf{\Sigma}_{\hat{\mathcal{I}}} & 0 \\ 0 & 0 & \mathbf{\Sigma}_{\check{\mathcal{I}} \setminus \hat{\mathcal{I}}} \end{bmatrix} \begin{bmatrix} \mathbf{\Psi}_{\mathcal{I}} & \mathbf{\Psi}_{\hat{\mathcal{I}}} & \mathbf{\Psi}_{\check{\mathcal{I}} \setminus \hat{\mathcal{I}}} \end{bmatrix}^{\top}.$$
 (10)

Leaving out the singular vectors and singular values with index in $\check{\mathcal{I}} \setminus \hat{\mathcal{I}}$ leads to a truncated SVD

$$\mathbf{\Phi}_{q} \mathbf{\Sigma}_{q} \mathbf{\Psi}_{q}^{\top} = \begin{bmatrix} \mathbf{\Phi}_{\mathcal{I}} & \mathbf{\Phi}_{\hat{\mathcal{I}}} \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_{\mathcal{I}} & 0 \\ 0 & \mathbf{\Sigma}_{\hat{\mathcal{I}}} \end{bmatrix} \begin{bmatrix} \mathbf{\Psi}_{\mathcal{I}} & \mathbf{\Psi}_{\hat{\mathcal{I}}} \end{bmatrix}^{\top}, \tag{11}$$

which includes the singular vectors and singular values with index $1, \ldots, q$ only that are in $\mathcal{I} \cup \hat{\mathcal{I}}$. The truncated SVD (11) has at most rank q. We sometimes refer to q as the truncation dimension.

Using only the truncated SVD in the least-squares problem (8) based on the set $\hat{\mathcal{I}}$ instead of $\check{\mathcal{I}}$ leads to

$$\min_{\boldsymbol{W} \in \mathbb{R}^{N \times p}} \|\boldsymbol{\Phi}_{\hat{\mathcal{I}}} \boldsymbol{\Sigma}_{\hat{\mathcal{I}}} \boldsymbol{\Psi}_{\hat{\mathcal{I}}}^{\top} + \boldsymbol{W} h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}}^{\top}) \|_F^2 + \gamma \|\boldsymbol{W}\|_F^2.$$
 (12)

By leaving out the difference

$$\mathbf{\Phi}_{\breve{\mathcal{I}} \setminus \hat{\mathcal{I}}} \mathbf{\Sigma}_{\breve{\mathcal{I}} \setminus \hat{\mathcal{I}}} \mathbf{\Psi}_{\breve{\mathcal{I}} \setminus \hat{\mathcal{I}}}^{\top} = \mathbf{\Phi}_{\breve{\mathcal{I}}} \mathbf{\Sigma}_{\breve{\mathcal{I}}} \mathbf{\Psi}_{\breve{\mathcal{I}}}^{\top} - \mathbf{\Phi}_{\hat{\mathcal{I}}} \mathbf{\Sigma}_{\hat{\mathcal{I}}} \mathbf{\Psi}_{\hat{\mathcal{I}}}^{\top}, \tag{13}$$

from the objective of (8), it is sufficient to have available the truncated SVD (11) of S. Similarly, we can approximate the objective J' given in (9) as

$$\hat{J}'(j, \boldsymbol{V}_{\mathcal{I}}, \boldsymbol{W}) = \left\| \boldsymbol{\Phi}_{\hat{\mathcal{I}} \setminus \{j\}} \boldsymbol{\Sigma}_{\hat{\mathcal{I}} \setminus \{j\}} \boldsymbol{\Psi}_{\hat{\mathcal{I}} \setminus \{j\}}^{\top} + \boldsymbol{W} h(\boldsymbol{\Sigma}_{\mathcal{I} \cup \{j\}} \boldsymbol{\Psi}_{\mathcal{I} \cup \{j\}}^{\top}) \right\|_{F}^{2} + \gamma \|\boldsymbol{W}\|_{F}^{2}, \quad (14)$$

which relies on the truncated SVD (11) only.

In summary, by using the objective (14) for selecting the V via the greedy construction and the minimization problem (12) to find W, we rely on the truncated SVD (11) of the data matrix S only. It is important to note that when solving (12) and evaluating (14), the low-rank matrices $\Phi_{\hat{\mathcal{I}}} \Sigma_{\hat{\mathcal{I}}} \Psi_{\hat{\mathcal{I}}}^{\top}$ and $\Phi_{\hat{\mathcal{I}} \setminus \{j\}} \Sigma_{\hat{\mathcal{I}} \setminus \{j\}} \Psi_{\hat{\mathcal{I}} \setminus \{j\}}^{\top}$, respectively, are not multiplied to avoid having to assemble large matrices and instead the corresponding least-squares problems are solved using the factorized forms. Moreover, in [55], it is shown that \hat{J}' can be evaluated without computing W leading to lower computational costs.

The following proposition bounds the difference between the minimizer of (8) and the minimizer when the truncated SVD is used and thus (12) is solved instead.

Proposition 3.1. The Frobenius norm of the difference between the minimizers of (8) and (12) is bounded by $\alpha \|\mathbf{\Sigma}^{\check{\mathcal{I}}\setminus\hat{\mathcal{I}}}\|_{F}$, where

$$\alpha = \frac{\sigma_{\max}(h(\mathbf{\Sigma}_{\mathcal{I}}\mathbf{\Psi}_{\mathcal{I}}^{\top})))}{\sigma_{\min}\left(h(\mathbf{\Sigma}_{\mathcal{I}}\mathbf{\Psi}_{\mathcal{I}}^{\top})h(\mathbf{\Sigma}_{\mathcal{I}}\mathbf{\Psi}_{\mathcal{I}}^{\top})^{\top} + \gamma I\right)}.$$

Proof. The minimizers $W^{(3.3)}$ and $W^{(3.7)}$ of (8) and (12), respectively are unique for $\gamma > 0$ and given by

$$\boldsymbol{W}^{(3.3)} = \left(h(\boldsymbol{\Sigma}_{\mathcal{I}}\boldsymbol{\Psi}_{\mathcal{I}}^{\top})h(\boldsymbol{\Sigma}_{\mathcal{I}}\boldsymbol{\Psi}_{\mathcal{I}}^{\top})^{\top} + \gamma I\right)^{-1}h(\boldsymbol{\Sigma}_{\mathcal{I}}\boldsymbol{\Psi}_{\mathcal{I}}^{\top})\boldsymbol{\Psi}_{\check{\mathcal{I}}}\boldsymbol{\Sigma}_{\check{\mathcal{I}}}\boldsymbol{\Phi}_{\check{\mathcal{I}}}^{\top},$$
$$\boldsymbol{W}^{(3.7)} = \left(h(\boldsymbol{\Sigma}_{\mathcal{I}}\boldsymbol{\Psi}_{\mathcal{I}}^{\top})h(\boldsymbol{\Sigma}_{\mathcal{I}}\boldsymbol{\Psi}_{\mathcal{I}}^{\top})^{\top} + \gamma I\right)^{-1}h(\boldsymbol{\Sigma}_{\mathcal{I}}\boldsymbol{\Psi}_{\mathcal{I}}^{\top})\boldsymbol{\Psi}_{\hat{\mathcal{I}}}\boldsymbol{\Sigma}_{\hat{\mathcal{I}}}\boldsymbol{\Phi}_{\hat{\mathcal{I}}}^{\top}.$$

Therefore, the difference $\Delta_{W} = W^{(3.3)} - W^{(3.7)}$ is given by

$$\Delta_{\boldsymbol{W}} = \left(h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}}^{\top}) h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}}^{\top})^{\top} + \gamma I \right)^{-1} h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}}^{\top}) \left(\boldsymbol{\Psi}_{\boldsymbol{I}} \boldsymbol{\Sigma}_{\boldsymbol{I}} \boldsymbol{\Phi}_{\boldsymbol{I}}^{\top} - \boldsymbol{\Psi}_{\hat{\mathcal{I}}} \boldsymbol{\Sigma}_{\hat{\mathcal{I}}} \boldsymbol{\Phi}_{\hat{\mathcal{I}}}^{\top} \right)
= \left(h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}}^{\top}) h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}}^{\top})^{\top} + \gamma I \right)^{-1} h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}}^{\top}) \boldsymbol{\Phi}_{\boldsymbol{I} \setminus \hat{\mathcal{I}}} \boldsymbol{\Sigma}_{\boldsymbol{I} \setminus \hat{\mathcal{I}}} \boldsymbol{\Psi}_{\boldsymbol{I} \setminus \hat{\mathcal{I}}}^{\top},$$

which has norm

$$\|\boldsymbol{\Delta}_{\boldsymbol{W}}\|_{F} = \left\| \left(h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}}^{\top}) h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}}^{\top})^{\top} + \gamma I \right)^{-1} h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}}^{\top}) \boldsymbol{\Phi}_{\check{\mathcal{I}} \backslash \hat{\mathcal{I}}} \boldsymbol{\Sigma}_{\check{\mathcal{I}} \backslash \hat{\mathcal{I}}} \boldsymbol{\Psi}_{\check{\mathcal{I}} \backslash \hat{\mathcal{I}}}^{\top} \right\|_{F}$$

$$\leq \frac{\sigma_{\max}(h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}})))}{\sigma_{\min}(h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}}) h(\boldsymbol{\Sigma}_{\mathcal{I}} \boldsymbol{\Psi}_{\mathcal{I}})^{\top} + \gamma I)} \|\boldsymbol{\Sigma}^{\check{\mathcal{I}} \backslash \hat{\mathcal{I}}}\|_{F}.$$

$$(15)$$

The bound (15) given in Proposition 3.1 scales with the sum of the discarded singular values $\Sigma^{\check{\mathcal{I}}\setminus\hat{\mathcal{I}}}$, which are all smaller than any of the singular values that are retained after truncation.

3.3 Incremental updates to truncated SVD of the data matrix

We now discuss an incremental updated version of the truncated SVD of the data matrix as data chunks S_0, S_1, S_2, \ldots are received over the iterations $k = 0, 1, 2, 3, \ldots$; see problem formulation in Section 2.4. Each chunk S_k is of size $N \times m$. At iteration k = 0, we have available an initial truncated SVD (11), which we denote with $\Phi_q^{(-1)} \in \mathbb{R}^{N \times q}, \Sigma_q^{(-1)} \in \mathbb{R}^{q \times q}, \Psi_q^{(-1)} \in \mathbb{R}^{0 \times q}$.

At iteration $k=0,1,2,3,\ldots$, we update the matrices $\boldsymbol{\Phi}_q^{(k-1)},\boldsymbol{\Sigma}_q^{(k-1)},\boldsymbol{\Psi}_q^{(k-1)}$ with a new data chunk \boldsymbol{S}_k to obtain $\boldsymbol{\Phi}_q^{(k)},\boldsymbol{\Sigma}_q^{(k)},\boldsymbol{\Psi}_q^{(k)}$. To perform the update, we follow the approach introduced in [2]. The first step is to process \boldsymbol{S}_k by computing the QR decomposition

$$Q^{(k)}R^{(k)} = [\Phi_q^{(k-1)}\Sigma_q^{(k-1)}, S_k] \in \mathbb{R}^{N \times (q+m)},$$
(16)

with the factors $Q^{(k)}$ and $R^{(k)}$. Furthermore, the matrix $\Psi_q^{(k-1)}$ of right-singular vectors is expanded as

$$\hat{\boldsymbol{\Psi}}_{q}^{(k-1)} = \begin{bmatrix} \boldsymbol{\Psi}_{q}^{(k-1)} & 0\\ 0 & \boldsymbol{I}_{m} \end{bmatrix}, \tag{17}$$

where I_m is the $m \times m$ identity matrix. The second step takes the SVD of $\mathbf{R}^{(k)}$ to obtain

$$\mathbf{R}^{(k)} = \mathbf{G}_{\Phi}^{(k)} \mathbf{G}_{\Sigma}^{(k)} {\mathbf{G}_{\Psi}^{(k)}}^{\top}.$$
 (18)

The matrix of left- and right-singular vectors of the SVD (18) act as transformations to compute

$$\mathbf{\Phi}_{q}^{(k)} = [\mathbf{Q}^{(k)} \mathbf{G}_{\Phi}^{(k)}]_{1:q}, \qquad \mathbf{\Psi}_{q}^{(k)} = [\hat{\mathbf{\Psi}}_{q}^{(k-1)} \mathbf{G}_{\Psi}^{(k)}]_{1:q}, \qquad (19)$$

where $[]_{1:q}$ means that the first q columns are used only. The diagonal matrix of the updated singular values $\Sigma_q^{(k)}$ is set to the diagonal matrix that has as diagonal elements the first q diagonal elements of $G_{\Sigma}^{(k)}$.

The incremental updating of the truncated SVD introduces errors. Roughly speaking, the error incurred by the incremental updating of the SVD is proportional to the smallest positive singular value of the truncated SVD. Thus, choosing q large enough can keep the

error of the incremental update under control; see [15, 3, 2] for a priori and a posteriori error bounds.

We now discuss the computational costs and storage requirements of the steps for updating the matrices $\Phi_q^{(k-1)} \in \mathbb{R}^{N \times q}, \Sigma_q^{(k-1)} \in \mathbb{R}^{q \times q}, \Psi_q^{(k-1)} \in \mathbb{R}^{km \times q}$ to $\Phi_q^{(k)} \in \mathbb{R}^{N \times q}, \Sigma_q^{(k)} \in \mathbb{R}^{q \times q}, \Psi_q^{(k)} \in \mathbb{R}^{km \times q}$ for a chunk S_k . The costs of computing the QR decomposition (16) scale as $\mathcal{O}(N(q+m)^2)$. The SVD of $\mathbf{R}^{(k)}$ computed in (18) has costs that scale as $\mathcal{O}((q+m)^3)$. At iteration k, the costs of the matrix multiplications in (19) scale as $\mathcal{O}(Nq^2)$ and $\mathcal{O}(kmq^2)$, respectively. Thus, at iteration k, the costs scale linearly in the dimension N of the data points and linearly in km but we avoid costs that scale as $\mathcal{O}(\max(N,km)\times\min(N,km)^2)$. The storage requirement is dominated by the matrices $\Phi_q^{(k)}$ and $\Psi_q^{(k)}$, since the full matrix S is never used and the chunks S_k are not accumulated in memory. Therefore, at iteration k, the storage requirement only scales linearly in N and the number of data samples (k+1)m received up to iteration k, that is our storage requirement scales as $\mathcal{O}(kmq+Nq)$. This is in stark contrast to storing the data matrix at iteration k, which scales as $\mathcal{O}(Nkm)$, whereas our approach has storage requirements that avoid the multiplicative scaling of dimension N and number of data points (k+1)m received up to iteration k and instead achieves an additive scaling as ((k+1)m+N)q.

3.4 Online greedy construction of quadratic manifolds from streaming data

Building on incremental updates to a truncated SVD of the data matrix allow us to derive Algorithm 1 for constructing quadratic manifolds from streaming data. The algorithm takes as input a data chunk S_k , the current SVD $\Phi_q^{(k-1)}$, $\Sigma_q^{(k-1)}$, $\Psi_q^{(k-1)}$, the truncation dimension q, the latent dimension n, as well as the regularization parameter γ , and the feature map h. The algorithm processes the incoming data chunk as follows. As the new data chunk S_k is received, first the SVD is updated (lines 4-7) and after that the greedy construction of a quadratic manifold using the updated SVD is performed. Note that the quadratic manifold does not have to be updated at each step. In our numerical experiments, we only update the SVD as we process the data chunks and compute the quadratic manifold only once after all chunks have been processed.

In Algorithm 1, we optimize for $\phi_{j_i}^{(k)}$ that minimizes (14) over all $\phi_{j_1}^{(k)}, \ldots, \phi_{j_q}^{(k)}$, that is, over all computed left singular vectors. In [55], we limit the search to only a small subset of the singular vectors, to trade off runtime and accuracy. Since in the streaming variant, the set of computed singular vectors q is already much smaller than N and M, we search all over all q computed left singular vectors.

4 Numerical experiments

We demonstrate the online greedy method with three numerical experiments, ranging from Hamiltonian wave problems to the Kelvin-Helmholtz instability. In the last example, we train a quadratic manifold on more than one Petabyte of data. We run the numerical experiments on Xeon Sapphire Rapids 52-core processors with 256GB of main

Algorithm 1 Online greedy construction of quadratic manifolds from streaming data

```
1: procedure StreamingQM(S_k, \Phi_q^{(k-1)}, \Sigma_q^{(k-1)}, \Psi_q^{(k-1)}, q, n, \gamma, h)
             // Update SVD
             \mathbf{Q}^{(k)}, \mathbf{R}^{(k)}, \hat{\mathbf{\Psi}}_{q}^{(k-1)} using (16) and (17).
 3:
             Compute G_{\Phi}^{(k)} and G_{\Psi}^{(k)} as left and right singular vectors of R^{(k)}.
 4:
             Set \Phi_q^{(k)} = Q^{(k)} G_{\Phi}^{(k)}, \ \Psi_q^{(k)} = \hat{\Psi}_q^{(k-1)} G_{\Psi}^{(k)}, \ \text{and} \ \Sigma_q^{(k)} = G_{\Phi}^{(k)}^{\top} R^{(k)} G_{\Psi}^{(k)}.
 5:
             Truncate \Phi_q^{(k)}, \Psi_q^{(k)}, and \Sigma_q^{(k)} to keep a rank q approximation. // Optional: update quadratic manifold
 6:
 7:
             Set \mathcal{I}_0 = \{\}, \check{\mathcal{I}}_0 = \{1, \dots, q\}, V_0 = []
 8:
             for i=1,\ldots,n do
 9:
                   Compute \phi_{j_i}^{(k)} that minimizes (14) over all \phi_{j_1}^{(k)}, \dots, \phi_{j_q}^{(k)} and \boldsymbol{W} \in \mathbb{R}^{N \times p}
10:
                   Set \mathcal{I}_i = \{j_1, \dots, j_i\} and \check{\mathcal{I}}_i = \{1, \dots, M\} \setminus \mathcal{I}_i
Set \mathbf{V}_i = [\boldsymbol{\phi}_1^{(k)}, \dots, \boldsymbol{\phi}_{j_k}^{(k)}]
11:
12:
             end for
13:
             end for Set V^{(k)} = [\phi_{j_1}^{(k)}, \dots, \phi_{j_r}^{(k)}] Compute W^{(k)} via the regularized least-squares problem (12).
14:
15:
16: end procedure
```

memory. The simulation and data-processing of our last example is carried out on an NVIDIA H-100 GPU with 80GB of memory. A python implementation using jax [11] is available at github.com/Algopaul/sqm_demo_gpu.

4.1 Hamiltonian interacting pulse

We consider data samples that are numerical solutions of the acoustic wave equation in Hamiltonian form,

$$\partial_t \rho(t, \mathbf{x}) = -\nabla v(t, \mathbf{x}),\tag{20}$$

$$\partial_t v(t, \mathbf{x}) = -\nabla \rho(t, \mathbf{x}),\tag{21}$$

$$\rho(0, \mathbf{x}) = \rho_0(\mathbf{x}; \mu), \tag{22}$$

$$v(0, \boldsymbol{x}) = 0, \tag{23}$$

where $\rho: [0,T] \times \Omega \to \mathbb{R}$ denotes the density and $v: [0,T] \times \Omega \to \mathbb{R}^2$ the velocity field. The time interval is [0,T] with T=8 and the spatial domain is $\Omega=[-4,4)^2$. We impose periodic boundary conditions. The initial condition ρ_0 depends on a parameter $\mu \in [0,1]$,

$$\rho_0(\mathbf{x}, \mu) = \exp\left(-(\mu + 6)^2 \left((x_1 - 2)^2 + (x_2 - 2)^2\right)\right),\tag{24}$$

where $\boldsymbol{x} = [x_1, x_2]^T \in \Omega$. We employ a central second order finite difference scheme with 600 grid points in each spatial direction, resulting in a state-space dimension of N = 1,080,000. Time is discretized with a fourth-order Runge-Kutta method with

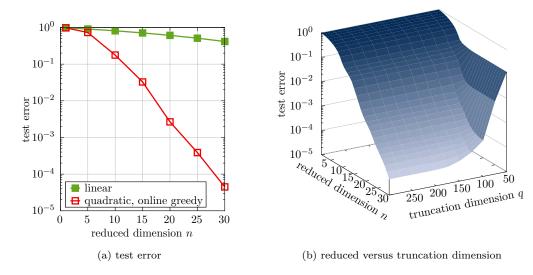


Figure 1: Hamiltonian wave: (a) The proposed online greedy method constructs a quadratic manifold from streaming data that achieves orders of magnitude low approximation errors (25) on test data than linear dimensionality reduction with the SVD. (b) As the reduced dimension n of the quadratic manifold is increased, the truncation dimension q has to be increased too so that the online greedy method can utilize higher-order left-singular vectors of the data matrix.

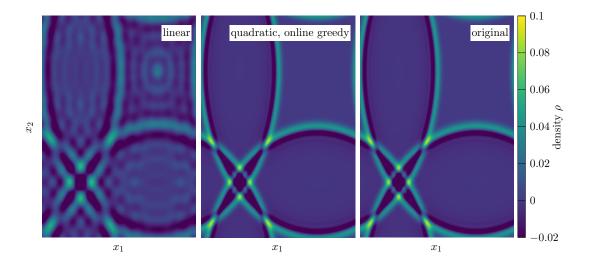


Figure 2: Hamiltonian wave: The online greedy method constructs quadratic manifolds from streaming data that achieve visibly better approximations of unseen test data than linear dimensionality reduction. Approximations shown for reduced dimension n=20 and time t=6.

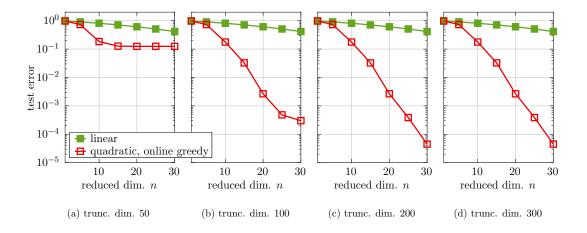


Figure 3: Hamiltonian wave: The plots show that the truncation dimension q and the reduced dimension n have to increase in tandem so that the online greedy method can leverage higher-order left-singular vector for larger n.

time-step size $\delta t = 5 \times 10^{-3}$. A data point s_i corresponds to the solution fields ρ and v evaluated at the N grid points at one of the $T/\delta t = 1601$ time points that are equidistantly distributed in [0, T].

We generate 101 trajectories corresponding to the initial conditions with parameters $\mu \in \{0, 1/100, 2/100, \dots, 1\}$. We divide the 101 trajectories into 99 training trajectories that lead to the data points in S, one validation trajectory corresponding to $\mu = 1/4$ that we collect in S^{valid} and one test trajectory corresponding to $\mu = 3/4$ that we store in S^{test} . The size of the training data S is approximately 1.37 TB.

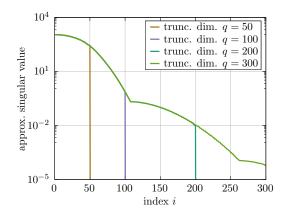
We process the training data with the online greedy method over the iterations $k = 0, 1, 2, 3, \ldots$, where we receive chunks $S_0, S_1, S_2, S_3, \ldots$ with size m = 347. We learn quadratic manifolds of dimensions $n \in \{1, 5, 10, 15, \ldots, 30\}$. For now, we set the truncation dimension to q = 300 because the largest considered reduced dimension n = 30 is multiplied by a factor 10, which is the convention also used in the work [55]; we will study the effect of q in more detail below. The test error that we report is the relative error of approximating test trajectories

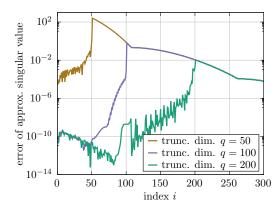
$$e(\mathbf{S}^{\text{test}}) = \frac{\|\hat{\mathbf{S}} - \mathbf{S}^{\text{test}}\|_F^2}{\|\mathbf{S}^{\text{test}}\|_F^2},$$
(25)

where $S^{\text{test}} = [s_1^{\text{test}}, s_2^{\text{test}}, \dots]$ contains the test trajectory and $\hat{S} = g(f(S^{\text{test}}))$ is the test trajectory approximated with the encoder function f and the decoder function g. The validation trajectory S^{valid} is used to select the regularization parameter γ over a sweep $10^{-8}, 10^{-6}, \dots, 10^{0}$ that leads to the lowest relative error that is analogously defined to (25) but for the validation data S^{val} . Based on this procedure, we select $\gamma = 10^{-8}$ in this example.

Figure 1a shows the relative error (25) of approximating the test trajectory. We compare the quadratic manifold to linear dimensionality reduction that uses the encoder and decoder function given by the SVD truncated at n as discussed in Section 2.1. Note that the linear encoder and decoder also use the incremental SVD as described in Section 3.3. The results in Figure 1a show that the quadratic manifold achieves orders of magnitude lower test errors (25) than linear dimensionality reduction. The results are in agreement with the plots shown in Figure 2, where the linear dimensionality reduction leads to visible artifacts, while the quadratic-manifold approximation is visually indistinguishable from the original density field.

We now investigate how the truncation dimension q of the incremental SVD influences the error of the online greedy method. Figure 1b shows the test error (25) over reduced dimensions $n \in \{1, 2, ..., 30\}$ and truncation dimensions $q \in \{50, 100, ..., 300\}$. A large reduced dimension n combined with a low truncation dimension q leads to a large error, which decreases as q is increased. This indicates that the truncation dimension q and the reduced dimension n have to increase in tandem so that the online greedy method can leverage higher-order left-singular vectors. Figure 3 shows slices of Figure 1b that further emphasize that q and n depend on each other. The reason that a low truncation dimension q leads to larger errors is twofold. First, by truncating too early, the truncated terms (13) still contain information that are important for fitting the quadratic manifold.





- (a) approximated singular values given by incremental SVD
- (b) absolute error of approximated singular values

Figure 4: Hamiltonian wave: Because the leading singular values are approximated well by the incremental SVD, it is sufficient to truncate the SVD early in the online greedy method. Increasing the truncation dimension q reduces the error of the approximated singular values.

Second, the truncated incremental SVD underlying the online greedy method incurs errors over the online iterations $k=0,1,2,3,\ldots$; see discussion in Section 3.3. The errors are typically of the order of the first singular value that is truncated, which means that truncating too early leads to quick error accumulation over the iterations. Figure 4 shows the approximations of singular values and their errors obtained with the incremental truncated SVD for $q \in \{50,100,200,300\}$. The absolute errors of the singular values are computed with respect to the singular value obtained with q=300. The leading singular values are computed to a high accuracy while the error increases for later singular values. Furthermore, the results show that the error in the approximations of the singular values depends on the truncation dimension q, which is in agreement with the discussion above.

4.2 Turbulent channel flow at a high Reynolds numbers

We now consider data samples that represent three-dimensional velocity fields of a turbulent channel flow, which are computed using AMR-Wind [14]. We use a wall-modeled large eddy simulation at a friction Reynolds number 5200, discretized with a staggered finite volume method into a grid of $384 \times 192 \times 32$ cells. The dimension of the data points is $N=3\times384\times192\times32=7,077,888$. We have a total of M=10000 data points, which we uniform randomly split into 9000 training data samples in \boldsymbol{S} , 500 validation data samples in $\boldsymbol{S}^{\text{valid}}$, and 500 test samples in $\boldsymbol{S}^{\text{test}}$. The training data \boldsymbol{S} is of size of about 510 GB. We receive data samples via chunks of size m=52. The setup of computing the quadratic manifold is the same as in Section 4.1.

Figure 5 shows the relative error (25) on the test data for the quadratic mani-

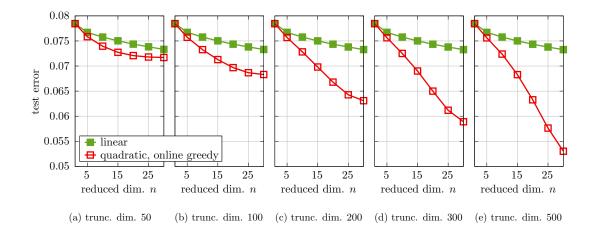


Figure 5: Channel flow: As the truncation dimension q is increased, the online greedy method constructs quadratic manifolds from streaming data that achieve about 40% lower test errors (25) than linear dimensionality reduction with the SVD.

fold with dimension $n \in \{1, 5, 10, 15, 20, 25, 30\}$ over the truncation dimension $q \in \{50, 100, 200, 300, 500\}$. Compared to the linear dimensionality with encoder and decoder corresponding to the SVD, the quadratic manifold achieves a lower test error (25). The difference between the error obtained with the quadratic manifold and linear dimensionality reduction is lower for this example than for the Hamiltonian wave example from Section 4.1. However, when comparing the velocity fields in Figure 6, it can be seen that the quadratic manifold approximates some of the high-frequency features of the velocity field with higher accuracy than linear dimensionality reduction. In Figure 7, we study the error in the approximation of the singular values of the online greedy method. Analogously to the results presented for the Hamiltonian wave problem in Section 4.1, the approximation accuracy of the singular values increases with the truncation dimension q.

4.3 Petabyte-scale data of Kelvin-Helmholtz instability

We now use the online greedy method as an in-situ data analysis tool to construct quadratic manifolds from streaming data as they are generated by a numerical simulation. This avoids having to store the data on disk because they are processed as they are generated.

Let us consider the Euler equations for gas dynamics in a two-dimensional domain $\Omega = [-1, 1)^2$ with periodic boundary conditions,

$$\frac{\partial}{\partial t} \begin{bmatrix} \rho \\ \rho v_{x_1} \\ \rho v_{x_2} \\ E \end{bmatrix} + \frac{\partial}{\partial x_1} \begin{bmatrix} \rho v_{x_1} \\ \rho v_{x_1}^2 + p \\ \rho v_{x_1} v_{x_2} \\ v_{x_1}(E+p) \end{bmatrix} + \frac{\partial}{\partial x_2} \begin{bmatrix} \rho v_{x_2} \\ \rho v_{x_1} v_{x_2} \\ \rho v_{x_2}^2 + p \\ v_{x_2}(E+p) \end{bmatrix} = 0,$$
(26)

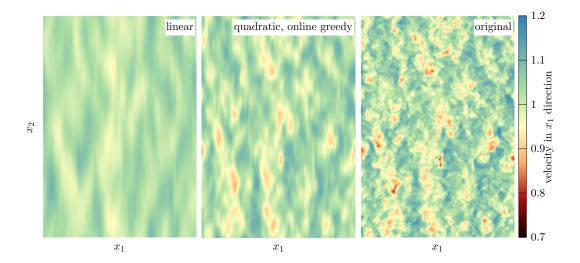


Figure 6: Channel flow: The online greedy method learns quadratic manifolds that capture fine-scale structure of the turbulent field that are missed by linear dimensionality reduction.

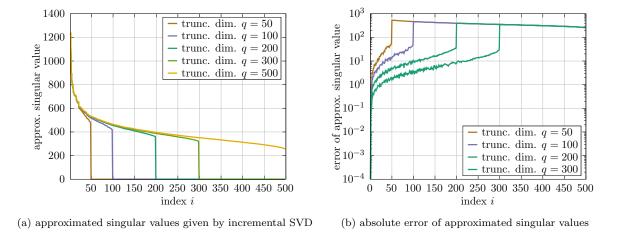


Figure 7: Channel flow: As the truncation dimension q is increased, the approximation accuracy of the singular values increases, which means that the online greedy method can leverage higher-order left-singular vectors for constructing more accurate quadratic manifolds; see also Figures 5–6.

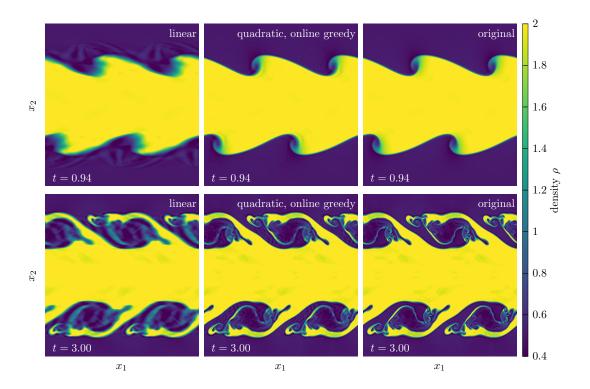


Figure 8: Kelvin-Helmholtz instability: The online greedy method processes more than one Petabyte of streaming data for constructing a quadratic manifold of the Kelvin-Helmholtz instability in this example. The online greedy method is run alongside the numerical simulation that produces the data in an in-situ fashion so that only small data chunks have to be kept in the main memory and costly disk input/output operations are avoided.

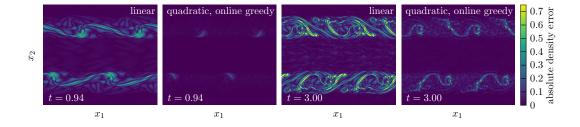
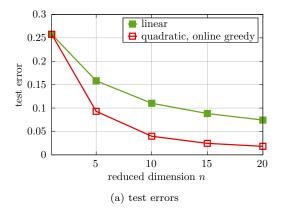
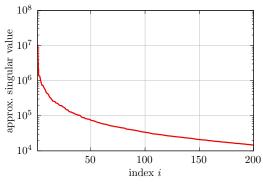


Figure 9: Kelvin-Helmholtz instability: The quadratic manifold obtained from more than one Petabyte of streaming data achieves lower errors than linear dimensionality reduction, which is visible in the fine-scale structure as the instability emerges. Results are shown for a quadratic manifold of dimension n=20.





(b) approximated singular value

Figure 10: Kelvin-Helmholtz instability: The plots report the error (25) of approximating test trajectories on the quadratic manifold learn from the Kelvin-Helmholtz instability. While the test error (25) corresponding to the quadratic manifold is only slightly lower than the error corresponding to linear dimensionality reduction, fine-scale features are more accurately captures by the quadratic manifold as can be seen in Figure 9.

where $\rho: [0,T] \times \Omega \to \mathbb{R}$, $v_{x_1}: [0,T] \times \Omega \to \mathbb{R}$, $v_{x_2}: [0,T] \times \Omega \to \mathbb{R}$, $E: [0,T] \times \Omega \to \mathbb{R}$, and $p: [0,T] \times \Omega \to \mathbb{R}$ denote the density, velocities in x_1 and x_2 direction, energy, and pressure, respectively. The equations are closed by the ideal gas law

$$p(t, \boldsymbol{x}) = (\Gamma - 1) \left(E(t, \boldsymbol{x}) - \frac{1}{2} \rho(t, \boldsymbol{x}) (v_{x_1}(t, \boldsymbol{x})^2 + v_{x_2}(t, \boldsymbol{x})^2) \right), \quad \forall \boldsymbol{x} \in \Omega, t \in [0, T],$$
(27)

with the gas-constant $\Gamma = 1.4$. To trigger the Kelvin-Helmholtz instability, we initialize the system based on the setup in [50] via

$$\rho(0, x_1, x_2) = 0.5 + b(x_1, x_2), \tag{28}$$

$$v_{x_1}(0, x_1, x_2) = \alpha(b(x_1, x_2) - 1) + 0.5, (29)$$

$$v_{x_2}(0, x_1, x_2) = 0.1\sin(2\pi x_1),\tag{30}$$

$$b(x_1, x_2) = 0.75(\tanh(\beta(x_2 + 0.5)) - \tanh(\beta(x_2 - 0.5)), \tag{31}$$

with a fixed $\beta=80$ and we set the initial pressure to 1.0 in the entire spatial domain. We vary α to cover different initial velocity profiles, which lead to changes in the resulting vortex shape. The equations of motion are discretized on a grid of size 1024×1024 using a Harten-Lax-van Leer flux as described in [31, Chapter 6.2.3]. We simulate the system until the final time T=3.

Our goal is learning a quadratic manifold from the data generated by 700 simulation runs that correspond to varying the parameter α equidistantly in the interval [0.495, 0.505]. One data point consists of the vectorized density, velocities in x_1 and x_2 directions, and the energy, such that the full dimension is $N = 4 \times 1024^2 = 4,194,304$. The 700 simulation runs correspond to $M = 3.19 \times 10^7$ data points. Storing all data

points for a post-processing data analysis steps would require 1.07 Petabytes of disk storage. Instead, we use our online greedy method and run it alongside the simulation in an in-situ fashion. In this way, we only need to store the current snapshot chunk together with the truncated SVD. In our experiment, the truncation dimension is q = 200, which reduces the necessary storage from 1.07PB to roughly 54GB. This fits into the main memory of the compute nodes that we use. The simulation and data-processing is executed on an NVIDIA H-100 GPU with 80GB of memory; for implementation details we refer to our solver at https://github.com/Algopaul/sqm_demo_gpu. In Figure 8, we show the reconstruction of the test-trajectory (where $\alpha = 0.5$) for a reduced dimension n=20. The corresponding point-wise absolute errors are shown in Figure 9 and the test error is shown in Figure 10. As the Kelvin-Helmholtz instability is emerging, the approximation obtained with linear dimensionality reduction exhibits substantial artifacts while the online greedy method provides an approximation that can even resolve the finer features of the underlying flow field. We stress that in this example our online greedy method processes more than one Petabyte of training data and that it is unnecessary to store the data as the greedy method is executed in an in-situ fashion alongside the numerical solver.

5 Conclusion

This work underscores the necessity of in-situ data processing for handling the ever-increasing volumes of data generated by complex numerical simulations. Traditional approaches, which rely on storing large datasets on disk for post-processing, are becoming unsustainable due to the limited main memory as well as slow disk input/output operations. The online greedy method introduced in this work constructs accurate quadratic manifold embeddings directly from streaming data. The data points are processed as they are generated, which allows using the online greedy method as an in-situ data analysis tool. The method's scalability is demonstrated by handling Petabyte-scale data. Furthermore, the method builds on standard linear algebra routines, ensuring compatibility and efficiency across various computing architectures.

Acknowledgement

This material is based upon work supported by the U.S. Department of Energy, Office of Science Energy Earthshot Initiative as part of the project "Learning reduced models under extreme data conditions for design and rapid decision-making in complex systems" under Award #DE-SC0024721, and the U.S. Department of Energy, Office of Scientific Computing Research, Award #DE-SC0019334. Furthermore, this work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by Department of Energy Office of Science Advanced Scientific Computing Research Program under Work Authorization No. KJ/AL16/24.

References

- [1] A. C. Antoulas, C. Beattie, and S. Gugercin. *Interpolatory Methods for Model Reduction*. SIAM, Philadelphia, 2020.
- [2] C. Baker, K. Gallivan, and P. Van Dooren. Low-rank incremental methods for computing dominant singular subspaces. *Linear Algebra Appl.*, 436(8):2866–2888, 2012. Special Issue dedicated to Danny Sorensen's 65th birthday.
- [3] C. G. Baker. A Block Incremental Algorithm for Computing Dominant Singular Subspaces. PhD thesis, 2004. Dissertation: retrieved from http://purl.flvc.org/fsu/fd/FSU_migr_etd-0961.
- [4] L. Balzano, Y. Chi, and Y. M. Lu. Streaming pca and subspace tracking: The missing data case. *Proceedings of the IEEE*, 106(8):1293–1310, 2018.
- [5] L. Balzano, R. Nowak, and B. Recht. Online identification and tracking of subspaces from highly incomplete information. In 2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pages 704–711, 2010.
- [6] J. Barnett and C. Farhat. Quadratic approximation manifold for mitigating the Kolmogorov barrier in nonlinear projection-based model order reduction. J. Comput. Phys., 464:111348, 2022.
- [7] P. Benner and T. Breiten. Two-sided projection methods for nonlinear model order reduction. SIAM Journal on Scientific Computing, 37(2):B239–B260, 2015.
- [8] P. Benner, P. Goyal, and S. Gugercin. H₂-quasi-optimal model order reduction for quadratic-bilinear control systems. SIAM Journal on Matrix Analysis and Applications, 39(2):983–1032, 2018.
- [9] P. Benner, P. Goyal, J. Heiland, and I. Pontes Duff. A quadratic decoder approach to nonintrusive reduced-order modeling of nonlinear dynamical systems. *PAMM*, 23(1):e202200049, 2023.
- [10] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. SIAM Rev., 57(4):483–531, 2015.
- [11] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018.
- [12] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Computer Vision — ECCV 2002*, pages 707–720, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

- [13] M. Brand. Fast low-rank modifications of the thin singular value decomposition. Linear Algebra and its Applications, 415(1):20–30, 2006. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.
- [14] M. Brazell, S. Ananthan, G. Vijayakumar, L. Cheung, M. Sprague, ExaWind Exascale Computing Project Team, and High Fidelity Modeling Project Team. AMR-Wind: Adaptive mesh-refinement for atmospheric-boundary-layer wind energy simulations. In APS Division of Fluid Dynamics Meeting Abstracts, APS Meeting Abstracts, page T29.007, 2021.
- [15] Y. Chahlaoui, K. Gallivan, and P. Van Dooren. Recursive calculation of dominant singular subspaces. SIAM Journal on Matrix Analysis and Applications, 25(2):445– 463, 2003.
- [16] K. L. Clarkson and D. P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 205–214, New York, NY, USA, 2009. Association for Computing Machinery.
- [17] A. Cohen, C. Farhat, A. Somacal, and Y. Maday. Nonlinear compressive reduced basis approximation for PDE's. working paper or preprint, 2023.
- [18] P. Comon and G. Golub. Tracking a few extreme singular values and vectors in signal processing. *Proceedings of the IEEE*, 78(8):1327–1343, 1990.
- [19] M. Couplet, P. Sagaut, and C. Basdevant. Intermodal energy transfers in a proper orthogonal decomposition—Galerkin representation of a turbulent separated flow. *Journal of Fluid Mechanics*, 491:275–284, 2003.
- [20] K. Duraisamy, G. Iaccarino, and H. Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51(1):357–377, 2019.
- [21] I.-G. Farcas, R. P. Gundevia, R. Munipalli, and K. E. Willcox. Distributed computing for physics-based data-driven reduced modeling at scale: Application to a rotating detonation rocket engine. *arXiv*, 2407.09994, 2024.
- [22] R. Geelen, L. Balzano, and K. Willcox. Learning latent representations in high-dimensional state spaces using polynomial manifold constructions, 2023.
- [23] R. Geelen, L. Balzano, S. Wright, and K. Willcox. Learning physics-based reducedorder models from data using nonlinear manifolds, 2023.
- [24] R. Geelen, S. Wright, and K. Willcox. Operator inference for non-intrusive model reduction with quadratic manifolds. Comput. Methods Appl. Mech. Engrg., 403:115717, 2023.
- [25] A. Gouasmi, E. J. Parish, and K. Duraisamy. A priori estimation of memory effects in reduced-order models of nonlinear systems using the Mori-Zwanzig formalism.

- Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 473(2205):20170385, 2017.
- [26] P. Goyal and P. Benner. Generalized quadratic embeddings for nonlinear dynamics using deep learning. arXiv, 2211.00357, 2024.
- [27] P. Goyal, I. P. Duff, and P. Benner. Guaranteed stable quadratic models and their applications in SINDy and operator inference. arXiv, 2308.13819, 2024.
- [28] C. Gu. Qlmor: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(9):1307–1320, 2011.
- [29] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review, 53(2):217–288, 2011.
- [30] M. S. Hemati, M. O. Williams, and C. W. Rowley. Dynamic mode decomposition for large and streaming datasets. *Physics of Fluids*, 26(11):111701, 11 2014.
- [31] J. S. Hesthaven. *Numerical Methods for Conservation Laws*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2018.
- [32] D. Huang, J. Niles-Weed, and R. Ward. Streaming k-pca: Efficient guarantees for oja's algorithm, beyond rank-one updates. In M. Belkin and S. Kpotufe, editors, Proceedings of Thirty Fourth Conference on Learning Theory, volume 134 of Proceedings of Machine Learning Research, pages 2463–2498. PMLR, 15–19 Aug 2021.
- [33] P. Jain, C. Jin, S. M. Kakade, P. Netrapalli, and A. Sidford. Streaming pca: Matching matrix bernstein and near-optimal finite sample guarantees for oja's algorithm. In V. Feldman, A. Rakhlin, and O. Shamir, editors, 29th Annual Conference on Learning Theory, volume 49 of Proceedings of Machine Learning Research, pages 1147–1164, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.
- [34] S. Jain, P. Tiso, J. B. Rutzmoser, and D. J. Rixen. A quadratic manifold for model order reduction of nonlinear structural dynamics. *Computers & Structures*, 188:80–94, 2017.
- [35] A. A. Kaptanoglu, J. L. Callaham, A. Aravkin, C. J. Hansen, and S. L. Brunton. Promoting global stability in data-driven models of quadratic nonlinear dynamics. *Phys. Rev. Fluids*, 6:094401, Sep 2021.
- [36] B. Kramer, B. Peherstorfer, and K. E. Willcox. Learning nonlinear reduced models from data with operator inference. *Annual Review of Fluid Mechanics*, 56(1):521–548, 2024.
- [37] Y. Li. On incremental and robust subspace learning. *Pattern Recognition*, 37(7):1509–1518, 2004.

- [38] J. Liew, T. Göçmen, W. H. Lio, and G. C. Larsen. Streaming dynamic mode decomposition for short-term forecasting in wind farms. Wind Energy, 25(4):719– 734, 2022.
- [39] H. Lu and D. M. Tartakovsky. Lagrangian dynamic mode decomposition for construction of reduced-order models of advection-dominated phenomena. *Journal of Computational Physics*, 407:109229, 2020.
- [40] B. Nadler. Finite sample approximation results for principal component analysis: A matrix perturbation approach. *The Annals of Statistics*, 36(6):2791–2817, 2008.
- [41] S. Pan and K. Duraisamy. Data-driven discovery of closure models. SIAM Journal on Applied Dynamical Systems, 17(4):2381–2413, 2018.
- [42] B. Peherstorfer. Breaking the Kolmogorov barrier with nonlinear model reduction. Notices of the American Mathematical Society, 69(5):725–733, 2022.
- [43] B. Peherstorfer and K. Willcox. Dynamic data-driven reduced-order models. Computer Methods in Applied Mechanics and Engineering, 291:21–41, 2015.
- [44] B. Peherstorfer and K. Willcox. Online adaptive model reduction for nonlinear systems via low-rank updates. SIAM Journal on Scientific Computing, 37(4):A2123–A2150, 2015.
- [45] B. Peherstorfer and K. Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016.
- [46] B. Peherstorfer and K. Willcox. Dynamic data-driven model reduction: adapting reduced models from incomplete data. Advanced Modeling and Simulation in Engineering Sciences, 3(1):11, Mar 2016.
- [47] E. Qian, B. Kramer, B. Peherstorfer, and K. Willcox. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 406:132401, 2020.
- [48] H. Qiao, P. Zhang, D. Wang, and B. Zhang. An explicit nonlinear mapping for manifold learning. *IEEE Transactions on Cybernetics*, 43(1):51 63, 2013. Cited by: 74; All Open Access, Green Open Access.
- [49] G. Rozza, D. B. P. Huynh, and A. T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. Arch. Comput. Methods Eng., 15(3):229–275, 2008.
- [50] A. M. Rueda-Ramírez and G. J. Gassner. A subcell finite volume positivity-preserving limiter for DGSEM discretizations of the Euler equations, 2021.

- [51] J. Rutzmoser, D. Rixen, P. Tiso, and S. Jain. Generalization of quadratic manifolds for reduced order modeling of nonlinear structural dynamics. *Computers & Structures*, 192:196–209, 2017.
- [52] P. Sagaut. Large Eddy Simulation for Incompressible Flows: An Introduction. Springer-Verlag, 2006.
- [53] N. Sawant, B. Kramer, and B. Peherstorfer. Physics-informed regularization and structure preservation for learning stable reduced models from data with operator inference. Computer Methods in Applied Mechanics and Engineering, 404:115836, 2023.
- [54] M. Schlegel and B. R. Noack. On long-term boundedness of Galerkin models. *Journal of Fluid Mechanics*, 765:325–352, 2015.
- [55] P. Schwerdtner and B. Peherstorfer. Greedy construction of quadratic manifolds for nonlinear dimensionality reduction and nonlinear model reduction, 2024.
- [56] H. Sharma, H. Mu, P. Buchfink, R. Geelen, S. Glas, and B. Kramer. Symplectic model reduction of Hamiltonian systems using data-driven quadratic manifolds. Computer Methods in Applied Mechanics and Engineering, 417:116402, 2023.
- [57] Z. Wang, I. Akhtar, J. Borggaard, and T. Iliescu. Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison. Computer Methods in Applied Mechanics and Engineering, 237-240:10-26, 2012.
- [58] X. Xie, M. Mohebujjaman, L. G. Rebholz, and T. Iliescu. Data-driven filtered reduced order modeling of fluid flows. SIAM Journal on Scientific Computing, 40(3):B834–B857, 2018.
- [59] S. Yildiz, P. Goyal, T. Bendokat, and P. Benner. Data-driven identification of quadratic representations for nonlinear Hamiltonian systems using weakly symplectic liftings. arXiv, 2308.01084, 2024.
- [60] L. Zanna and T. Bolton. Data-driven equation discovery of ocean mesoscale closures. Geophysical Research Letters, 47(17):e2020GL088376, 2020.
- [61] H. Zhang, C. W. Rowley, E. A. Deem, and L. N. Cattafesta. Online dynamic mode decomposition for time-varying systems. SIAM Journal on Applied Dynamical Systems, 18(3):1586–1609, 2019.