A Modern Perspective on Streaming PCA and Subspace Tracking: The Missing Data Case
Laura Balzano, Member, IEEE, Yuejie Chi, Senior Member, IEEE, and Yue M. Lu, Senior Member, IEEE

Abstract—For many modern applications in science and engineering, data are collected in a streaming fashion carrying time-varying information, and practitioners need to process them with a limited amount of memory and computational resources in a timely manner for decision making. This often is coupled with the missing data problem, such that only a small fraction of data attributes are observed. These complications impose significant, and unconventional, constraints on the problem of streaming Principal Component Analysis (PCA) and subspace tracking, which is an essential building block for many inference tasks in signal processing and machine learning. This survey article reviews a variety of classical and recent algorithms for solving this problem with low computational and memory complexities, particularly those applicable in the big data regime with missing data. We illustrate that streaming PCA and subspace tracking algorithms can be understood through algebraic and geometric perspectives and they need to be adjusted carefully to handle missing data. Both asymptotic and non-asymptotic convergence guarantees are reviewed. Finally, we benchmark the performance of several competitive algorithms in the presence of missing data for both well-conditioned and ill-conditioned systems.

Index Terms—subspace tracking, subspace and low-rank models, missing data, ODE analysis, streaming PCA

I. INTRODUCTION

The explosion of data collection across a variety of domains, for purposes that range from scientific to commercial to policy-oriented, has created a data deluge that requires new tools for extracting useful insights from data. Principal Component Analysis (PCA) [1] and subspace tracking are arguably some of the most commonly used tools for exploring and understanding data. The fundamental mathematics and algorithms for identifying signal subspaces from data have been studied for nearly a century. However, in the modern context, many novel challenges arise, due to the severe mismatch between the limited resources available at computational platforms and the increasing demand of processing high-dimensional data. In particular, this survey article is motivated by the following aspects of modern data processing.

• Large-scale and high-rate. Data are collected at an extremely large scale with many variables, such as in video surveillance or internet monitoring, and they can accumulate at such high rates that necessitate real-time processing for timely decision making. Therefore, classical batch algorithms for data processing are replaced by online, streaming algorithms that have much smaller memory and computational footprints.

• Missing data. At each time instance, only a very small subset of the data attributes may be measured, due to hardware limitations, power constraints or simple lack of observations. Therefore, classical algorithms that do not account for missing data may yield highly sub-optimal performance and need to be redesigned.

To elaborate on these modern challenges, we describe two concrete examples in more detail. First, consider recommendation systems [2], where users’ past product use and opinions are collected. Based on such data, the system attempts to predict other products of interest to those (and potentially other) users. This is of course a scenario involving extremely sparse observations in high dimensions—a user has only purchased or rated a vanishingly small number of products from a company. Moreover, as the users rate more products and new products become available, it is desirable to update the system’s predictions on user preference in an online manner.

As another example, consider the rigid structure from motion problem in computer vision [3], [4]. One seeks to build a 3D model of a scene based on a sequence of 2D images that, for an orthographic camera, are projections of that scene onto a plane. Features in the scene can be tracked through the images, and a matrix of their locations in the images has a low-rank (3-dimensional) factorization in terms of the true 3D locations of feature points and the locations of the cameras at each image frame. The problem is obviously high dimensional and it is also natural to consider the streaming setting, as large numbers of features can be tracked across image frames that arrive sequentially at a high rate. Moreover, many points in the scene are not visible in all image frames due to occlusion. Therefore, while the low-rank subspace of the data recovers the 3D structure of the entire scene, one must estimate this subspace in the presence of missing data.

The list of modern applications continues. The question is: can we have scalable and accurate algorithms for subspace learning that work well even in the presence of missing data?

A. Subspace Models and Streaming PCA

Subspace models have long been an excellent model for capturing intrinsic, low-dimensional structures in large datasets. A celebrated example, PCA [1], has been successfully applied to many signal processing applications including medical imaging, communications, source localization and clutter tracking.
in radar and sonar, computer vision for object tracking, system identification, traffic data analysis, and speech recognition, to name just a few. The calculated principal components and best-fit subspaces to a dataset not only allow powerful dimensionality reduction but also provide intermediate means for signal estimation, noise removal, and anomaly detection [5]. As we highlight in this paper, the principal components can be updated using incoming data in a streaming manner, thus offering tracking capabilities that are necessary for real-time decision making.

While there are a plethora of traditional algorithms for performing PCA on a batch dataset, e.g., the Singular Value Decomposition (SVD), and for tracking the principal components in a streaming scenario (see, e.g., [6], for a survey of earlier literature), most of these algorithms were developed during a time when datasets of interest had a moderate number of variables and were collected in a controlled environment without missing entries. As argued earlier, modern datasets are being collected on vast scales, in a much less controlled way, often with overwhelmingly many missing entries. In light of this prevalent and modern challenge in signal processing and machine learning, classical algorithms must be adjusted in order to gracefully handle missing data.

When do we have hope to recover missing data? If the complete full-dimensional data are well-approximated by their projection onto a lower-dimensional subspace, and hence in some sense redundant, then it is conceivable that incomplete or subsampled data may provide sufficient information for the recovery of that subspace. A related problem in the batch setting is the celebrated problem of low-rank matrix completion [7], which suggests that it is possible to recover a highly incomplete matrix if its rank is much smaller than the dimension. This is the central intuition that motivates work on subspace tracking with missing data. A burst of research activity has been devoted to developing algorithms and theoretical underpinnings for this problem over the last several years in signal processing, machine learning, and statistics. Moreover, powerful results from random matrix theory and stochastic processes are leveraged to develop performance guarantees for both traditional and newly proposed methods. At the same time, these methods are also finding new applications to emerging data science applications such as monitoring of smart infrastructures [8], neurological, and physiological signal processing and understanding [9].

B. Overview of Subspace Tracking Algorithms

There is a long history of streaming PCA and subspace tracking algorithms in the literature of signal processing. An extensive survey of methods prior to 1990 was provided in a popular Proceedings of the IEEE article by Comon and Golub [6]. As the common problem dimension was relatively small back then, the focus was mostly on performance and computational complexity for fully observed data of moderate dimensions. Since then, new algorithms have been, and continue to be developed with a focus on minimizing computation and memory complexity for very high-dimensional problems and missing data, without suffering too much on performance [10]. Consider the problem of estimating or tracking a k-dimensional subspace in \( \mathbb{R}^d \), where \( k \ll d \). For modern applications, it is desirable that both the computational complexity (per update) and the memory complexity scale at most linearly with respect to \( d \). Moreover, modern applications may require the algorithm to handle a range of missing data, from just a small fraction of missing entries to the information-theoretic limit of \( O(k \log d) \) entries observed in each data vector\(^1\).

Broadly speaking, there are two perspectives from which researchers have developed and studied streaming PCA and subspace tracking algorithms, as categorized by Smith [13]. The first class of algorithms can be interpreted through an algebraic lens; these can be regarded as variants of incremental methods for calculating top-k eigenvectors or singular vectors of a time-varying matrix, such as the sample covariance matrix. Since this time-varying matrix is typically updated by a rank-one modification, various matrix manipulation techniques can be exploited to reduce computational and memory complexities. This viewpoint is particularly useful for understanding algorithms such as incremental SVD [14], Karasalo’s method [15], Oja’s method [16], Kruskal’s method [17], [18], and other algorithms based on power iterations [19], [20], to name a few.

The other class of algorithms can be interpreted through a geometric lens. These algorithms are constructed as solutions to the optimization of certain loss functions, e.g., via gradient descent, designed in either Euclidean space or on a matrix manifold such as the Grassmannian. We focus mainly on methods where the loss function is updated by one additional term per streaming column vector, and the previous estimate can be used as a warm start or initialization. This viewpoint is particularly useful in the presence of missing data, and has been leveraged more often in the design of subspace tracking algorithms that are tolerant to missing data. Examples include GROUSE [21], PETRELs [22], [23], ReProCS [24], PAST [25], online nuclear norm minimization [26], and other algorithms based on stochastic approximation [27], to name a few.

The two classes of algorithms, while having distinct features, can often be unified, as an algorithm can often be interpreted through both perspectives. Trade-offs between convergence speed in static environments and tracking speed in dynamic environments are also important considerations in practice, achieved by balancing the influence from historical data and from current data. This can be done by discounting historical data in the construction of the time-varying matrix in algebraic methods, and in the construction of the loss function or selection of step sizes in geometric methods.

There is also a vast literature on establishing theoretical performance guarantees for various streaming PCA and subspace tracking algorithms. Classical analysis is primarily done in the asymptotic regime (see, e.g., [28], [29]), where the discrete-time stochastic processes associated with the algorithms are shown to converge, in the scaling limit [30], [31], to the

\(^1\)This is the information-theoretic lower bound of measurements for an arbitrary incoherent rank-k matrix when about d total column vectors are observed [11]. For a generic matrix, we need only \( O(\max(k, \log d)) \) entries per column with \( O(kd) \) total columns [12].
solution of some deterministic differential equations. Recent developments in performance analysis include new and more tractable asymptotic analysis for high-dimensional cases [32–34], as well as finite-sample probabilistic performance guarantees [35–39].

C. Organization of the Paper

We first describe in Section II the problem formulation of PCA and streaming PCA in the presence of missing data. We then survey algorithms that perform streaming subspace estimation and tracking with full or incompletely observed columns: Section III focuses on those using algebraic approaches and Section IV on those using geometric approaches. Many of these algorithms have associated theoretical analysis with regards to the estimation accuracy and algorithmic convergence rates, which we discuss in Section V. We then provide numerical comparisons of a number of competitive algorithms in Section VI and conclude in Section VII.

D. Notations

Throughout this paper, we use boldface letters to denote vectors and matrices, e.g., \( \mathbf{a} \) and \( \mathbf{A} \). For a positive semidefinite (PSD) matrix \( \mathbf{A} \), we write \( \mathbf{A} \succeq 0 \). The transpose of \( \mathbf{A} \) is denoted by \( \mathbf{A}^T \), and \( \| \mathbf{A} \| \), \( \| \mathbf{A} \|_F \), and \( \text{Tr}(\mathbf{A}) \) denote the spectral norm, the Frobenius norm and the trace, respectively. The expectation of a random variable \( \mathbf{a} \) is written as \( \mathbb{E}[\mathbf{a}] \). The identity matrix of dimension \( k \) is written as \( \mathbf{I}_k \). We shall use \( d \) to denote the dimension of the fully observed data vector and \( k \) to denote the dimension of the subspace to be estimated. A subscript \( n \) on the data vector \( \mathbf{x}_n \in \mathbb{R}^d \) refers to its order in a sequence of vectors, and the notation \( \mathbf{x}_n(i) \) refers to the \( i \)-th component of the vector \( \mathbf{x}_n \).

II. PROBLEM FORMULATION

In this section, we will start by formulating the problem of subspace estimation in the batch setting, which serves as a good starting point to motivate subspace tracking in the streaming setting with missing data.

A. PCA in the Batch Setting

The PCA or subspace estimation problem can be formulated either \textit{probabilistically}, where data are assumed to be random vectors drawn from some distributions with mean zero and some covariance matrix whose principal subspace we wish to estimate, or \textit{deterministically}, where we seek the best rank-\( k \) subspace that fits the given data. Both models are used extensively throughout the literature. The former is used more prevalently in the signal processing and statistics literature, while the latter is more prevalent in applied mathematics, optimization, and computer science literature. The problem formulations result in equivalent optimization problems, and so we put them here together for a unified view.

(a) Probabilistic view: Consider a stationary, \( d \)-dimensional random process \( \mathbf{x} \in \mathbb{R}^d \), which has a zero mean and a covariance matrix \( \Sigma = \mathbb{E}[\mathbf{x}\mathbf{x}^T] \). Denote the eigenvalue decomposition (EVD) of the covariance matrix as \( \Sigma = \mathbf{U} \Lambda \mathbf{U}^T \), where \( \mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_d] \) has orthonormal columns, and \( \Lambda = \text{diag}\{\lambda_1, \ldots, \lambda_d\} \), where \( \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0 \) are the eigenvalues arranged in a non-increasing order. Our goal is to estimate the top-\( k \) eigenvectors, also called the principal components \( \mathbf{U}^* = [\mathbf{u}_1, \ldots, \mathbf{u}_k] \), of \( \Sigma \), given a finite number of i.i.d. data samples, \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \sim \mathbf{x} \). Note that we do not require \( \Sigma \) be a rank-\( k \) matrix.

(b) Deterministic view: In a deterministic formulation, the data samples \( \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d \) are considered arbitrary. We wish to find the rank-\( k \) subspace that best fits these data in the sense of minimizing the \( \ell_2 \) projection error, that is

\[
\hat{\mathbf{U}}_n = \arg\min_{\mathbf{U} \in \mathbb{R}^{d \times k}, \mathbf{U}^T \mathbf{U} = \mathbf{I}_k} \sum_{i=1}^n \| \mathbf{x}_i - \mathbf{P}_\mathbf{U}(\mathbf{x}_i) \|_2^2
\]

\[
= \arg\min_{\mathbf{U} \in \mathbb{R}^{d \times k}, \mathbf{U}^T \mathbf{U} = \mathbf{I}_k} \| \mathbf{X}_n - \mathbf{P}_\mathbf{U}(\mathbf{X}_n) \|_F^2
\]

\[
= \arg\max_{\mathbf{U} \in \mathbb{R}^{d \times k}, \mathbf{U}^T \mathbf{U} = \mathbf{I}_k} \text{Tr}(\mathbf{U} \mathbf{U}^T \Sigma_n)
\]

where \( \mathbf{P}_\mathbf{U} \) denotes the projection operator onto the column span of the matrix \( \mathbf{U} \), \( \mathbf{X}_n = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \) concatenates the data vectors as columns into a matrix, and \( \Sigma_n = \sum_{t=1}^n \mathbf{x}_t \mathbf{x}_t^T = \mathbf{X}_n \mathbf{X}_n^T \) is the (unscaled) Sample Covariance Matrix (SCM).

The equivalence of Eq. (1) and (2) suggests that finding the subspace that maximizes the explainable variance of \( \Sigma_n \) is equivalent to minimizing the approximation error of the data matrix \( \mathbf{X}_n \). While the formulations (1) or (2) are non-convex, both due to the cost function’s non-convexity in \( \mathbf{U} \) and the non-convex constraint \( \mathbf{U}^T \mathbf{U} = \mathbf{I}_k \), they admit a closed-form solution, solved by the SVD of \( \mathbf{X}_n \), equivalently the EVD of \( \Sigma_n \), as was discovered independently by [41] (see [42], [43] for details) and [44]. Specifically, the solution \( \hat{\mathbf{U}}_n \) is given as the top-\( k \) eigenvectors of the SCM \( \Sigma_n \).

(c) Unified perspective: Consider the following expected loss function

\[
J(\mathbf{U}) = \mathbb{E} \| \mathbf{x} - \mathbf{U} \mathbf{U}^T \mathbf{x} \|_2^2
\]

where \( \mathbf{U} \in \mathbb{R}^{d \times k} \) and the expectation is taken with respect to \( \mathbf{x} \). The following important result was proven in [25]: \( \mathbf{U} \) is a stationary point of \( J(\mathbf{U}) \) if and only if it is an orthonormal matrix with a column space spanned by \( k \) eigenvectors of \( \Sigma \). Moreover, if \( \lambda_k > \lambda_{k+1} \), i.e., if there is a strict eigengap, then all stationary points of \( J(\mathbf{U}) \) are saddle points with at least one descent direction, except those corresponding to global optima where \( \mathbf{U} \) contains the top-\( k \) eigenvectors of \( \Sigma_n \) up to an orthonormal transformation.

Interestingly, the solution to the deterministic formulation (2) can be thought of as an empirical version of (3), if the data samples are indeed drawn according to the probabilistic model. Moreover, in this case, \( \hat{\mathbf{U}}_n \) produces an order-wise near-optimal estimate to \( \mathbf{U}^* \) for a large family of distributions [45]. In this regard, the two formulations are equivalent in

\footnote{It is straightforward to consider the complex-valued case \( \mathbf{x} \in \mathbb{C}^d \), but we only consider the real case in this survey for simplicity. For more information, see [40].}
some sense, though in the deterministic setting, there need not be any generative model or “ground truth” for the underlying subspace.

B. Streaming PCA and Subspace Tracking

In a streaming setting, the data samples arrive sequentially over time, and each sample is only seen once, and one wishes to update the subspace estimate sequentially without accessing historical data. In a dynamic environment, either the covariance matrix or the best rank-$k$ subspace can be time-varying — therefore, we wish to track such changes as quickly as possible, and this problem is known as subspace tracking in the signal processing literature.

Streaming PCA can be considered as a nonconvex stochastic approximation problem [given by Eq. (3)]. The solution to the batch problem that we outlined in Section II-A is no longer appropriate for the streaming setting — it requires one to formulate and store the SCM $\Sigma_n$, which has a memory complexity of $O(d^2)$, and to estimate the top-$k$ eigenvectors, which has a time complexity of $O(nd^2)$. Both these memory and time complexities are too expensive for large-scale problems. It is greatly desirable to have algorithms with computation and memory complexity that grows at most linearly in $d$.

C. Missing Data

An important perspective that we will consider in this survey is missing data, when each sample may not be completely observed. In this case, only a subset of the coordinates of $x_n$ are observed. We denote this measurement as

$$y_n = \mathcal{P}_{\Omega_n}(x_n),$$

where $\mathcal{P}_{\Omega_n}$ is an observation mask, $\Omega_n \in \{0, 1\}^d$, where the $i$th entry of $x_n$, $x_n(i)$, is observed if and only if $\Omega_n(i) = 1$. This issue poses severe challenges for most PCA algorithms, particularly when the number of observed entries is much smaller than $d$. To begin, one may be concerned with identifiability: can we find a unique subspace of rank-$k$ that are consistent with the partial observations? Luckily, the answer to this question is yes, at least in the batch setting where the problem is equivalent to that of low-rank matrix completion: under mild assumptions, the low-rank subspace can be reconstructed from subsampled column vectors as long as there are enough measurements. It may also be tempting to execute subspace tracking algorithms by ignoring the missing data and padding with zeros at the missing entries, however the sample covariance matrix constructed in this way leads to a biased estimator [46], [47]. Therefore, one must think more carefully about how to handle missing data in this context.

III. ALGEBRAIC METHODS

In this section and the section below, we will discuss two classes of algorithms based on algebraic approaches and geometric approaches respectively, as outlined in Section I-B. The algebraic approaches are based on finding the top eigenvectors of a recursively updated SCM, or a surrogate of it, given as

$$\Sigma_n = \alpha_n \Sigma_{n-1} + \beta_n x_n x_n^T,$$

where $\alpha_n$ and $\beta_n$ balance the contributions from the previous SCM and the current data sample. Two popular choices are equal weights on all time slots, which is

$$\alpha_n = 1, \quad \beta_n = 1;$$

and discounting on history data, which is

$$\alpha_n = \lambda, \quad \beta_n = 1, \quad 0 < \lambda < 1.$$

Equivalently, the above can be reworded for finding the top singular vectors of a recursively updated data matrix $X_n$. As we are interested in calculating or approximating the top-$k$ eigenvectors of $\Sigma_n$, algebraic methods use matrix manipulations and exploit the simplicity of the rank-one update to reduce the complexity.

A. Incremental Singular Value Decomposition (ISVD)

We begin by discussing the ISVD approach of Bunch and Neilsen [48], which is an exact method to solve the streaming PCA problem with sequentially arrived, full data vectors. This algorithm is given in Algorithm 1. Suppose we are given the SVD of the data matrix at time $n-1$,

$$X_{n-1} = [x_1 \ldots x_{n-1}] = \tilde{U}_{n-1} \tilde{S}_{n-1} \tilde{V}_{n-1}^T,$$

where $\tilde{U}_{n-1} \in \mathbb{R}^{d \times d}$ and $\tilde{V}_{n-1} \in \mathbb{R}^{(n-1) \times (n-1)}$ are orthonormal, and $\tilde{S}_{n-1} \in \mathbb{R}^{d \times (n-1)}$ is the concatenation of two matrices: a diagonal matrix (of size $\max\{d, n-1\}$) with non-negative non-increasing diagonal entries, and an all-zero matrix. For simplicity of exposition, let’s assume $d \geq n$ (but both cases $d < n$ and $d \geq n$ are described in Algorithm 1). We wish to compute the SVD of

$$X_n = [x_1 \ldots x_n] = \tilde{U}_{n-1} \tilde{S}_{n-1} \tilde{V}_{n-1}^T,$$

where $\tilde{U}_{n}$, $\tilde{S}_{n}$, and $\tilde{V}_{n}$ are defined similarly as $\tilde{U}_{n-1}$, $\tilde{S}_{n-1}$, and $\tilde{V}_{n-1}$. Recognizing that

$$X_n X_n^T = X_{n-1} X_{n-1}^T + x_n x_n^T,$$

and

$$\tilde{U}_{n-1}^T X_n X_n^T \tilde{U}_{n-1} = \tilde{S}_{n-1} \tilde{S}_{n-1}^T + z_n z_n^T,$$

where $z_n = \tilde{U}_{n-1}^T x_n$, we can compute the new singular values by finding the eigenvalues of $\tilde{S}_{n-1} \tilde{S}_{n-1}^T + z_n z_n^T$ using the zeros of the characteristic equation [49], which in this case has a special structure; in particular, if $\tilde{\sigma}_i$ are the diagonal values of $\tilde{S}_{n-1}$, then the zeros of

$$1 + \sum_{i=1}^d \frac{z_n(i)^2}{\tilde{\sigma}_i^2 - \lambda}$$

with respect to the variable $\lambda$ identify the eigenvalues of $\tilde{S}_{n-1} \tilde{S}_{n-1}^T + z_n z_n^T$. Denote the resulting eigenvalues as $\lambda_i$ for $i = 1, \ldots, d$. To update the left singular vectors $\tilde{U}_n = [u_1 \ldots u_d]$, we need to solve
Algorithm 1 ISVD

1: Given $x_1$, set $U_1 = x_1/\|x_1\|$, $S_1 = \|x_1\|$, $V_1 = 1$;
2: Set $n = 2$;
3: repeat
4: Define $w_n := U_n^T x_n$;
5: Define $p_n := U_n^{-1} w_n$; $r_n := x_n - p_n$;
6: if $\|r_n\| \neq 0$ then
7: The decomposition of Eq. (11) is applicable.
8: Compute the SVD of the update matrix:
9: \[
\begin{bmatrix}
    S_{n-1} & w_n \\
    0 & \|r_n\|
\end{bmatrix} = \hat{U} \hat{S} \hat{V}^T,
\]
10: by solving Eq. (6) for $\hat{S}$ with
11: $z_n = [w_n^T \|r_n\| 0]^T$,
12: solving Eq. (7) for $\hat{U}$.
13: Set
14: $U_n := [U_{n-1} \quad \frac{r_n}{\|r_n\|}] \hat{U}$, $S_n := \hat{S}$.
15: else if $\|r_n\| = 0$ then
16: (this happens when $n > d$ or $x_n \in \text{span}(U_{n-1})$)
17: Compute the SVD of the update matrix:
18: \[
\begin{bmatrix}
    S_{n-1} & w_n \\
    0 & \|r_n\|
\end{bmatrix} = \hat{U} \hat{S} \hat{V}^T,
\]
19: by solving Eq. (6) for $\hat{S}$ with $z_n = [w_n^T 0]^T$,
20: solving Eq. (7) for $\hat{U}$.
21: Set
22: $U_n := U_{n-1} \hat{U}$, $S_n := \hat{S}$.
23: end if
24: $n := n + 1$;
25: until termination

\[
\left(\frac{S_{n-1} S_{n-1}^T + z_n z_n^T}{S_{n-1} S_{n-1}^T - \lambda I}\right) u_i = \lambda_i u_i \text{ and normalize the solution. Therefore } [6], [49],
\]
\[
u_i = \frac{(S_{n-1} S_{n-1}^T - \lambda I)^{-1} z_n}{\|S_{n-1} S_{n-1}^T - \lambda I\|^{-1} z_n} , \quad i = 1, \ldots, d.\tag{7}
\]

So far, the above derivations assume $U_{n-1}$ is a square orthonormal matrix, and the resulting computations are suitable for incremental updates of the full SVD. However, this still requires an $O(\max\{d, n\}^3)$ complexity to compute the full SVD incrementally, which is very expensive. On the other end, estimating a thin SVD or the top $k$-dimensional singular subspace can improve computation. In fact, if $X_{n-1}$ is rank-$k$, this incremental approach requires fewer computations as pointed out in [50]. In this case, we can take just the first $k$ columns in $U_{n-1}$; call these $U_{n-1}$. Let $S_{n-1}$ and $V_{n-1}$ be the corresponding matrices for this thin SVD so that $X_{n-1} = U_{n-1} S_{n-1} V_{n-1}^T$. We then notice as in [50], [51] that

\[
X_n = \begin{bmatrix}
    U_{n-1} & \frac{r_n}{\|r_n\|} \\
    S_{n-1} & w_n \\
    0 & \|r_n\|
\end{bmatrix} \begin{bmatrix}
    S_{n-1} & w_n \\
    0 & \|r_n\|
\end{bmatrix} \begin{bmatrix}
    V_{n-1}^T & 0 \\
    0 & 1
\end{bmatrix} = \hat{U} \hat{S} \hat{V}^T
\]

Algorithm 2 MD-ISVD, Brand’s algorithm, PIMC

1: Given an orthonormal matrix $U_0 \in \mathbb{R}^{d \times k}$, $S_0 = 0$;
2: For PIMC, $\gamma_0 = 1$.
3: Set $n = 1$;
4: repeat
5: Define $w_n := \arg \min_{w} \|P_{\Omega_n} (U_{n-1} w - x_n) \|^2$;
6: Define $p_n := U_{n-1} w_n$;
7: Compute the SVD of the update matrix:
8: \[
\begin{bmatrix}
    \Gamma_{n-1} & w_n \\
    0 & \|r_n\|
\end{bmatrix} = \hat{U} \hat{S} \hat{V}^T,
\]
9: where
10: $\Gamma_{n-1} = \begin{cases}
    S_{n-1} & \text{for MD-ISVD,} \\
    \lambda S_{n-1} & \text{for Brand’s algorithm,}
\end{cases}$
11: $\begin{cases}
    S_{n-1} & \text{for MD-ISVD,} \\
    \frac{\gamma_n^2}{\|S_{n-1}\|} S_{n-1} & \text{for PIMC,}
\end{cases}$
12: with $\gamma_n^2 = \gamma_{n-1}^2 + \|P_{\Omega_n} (x_n)\|^2$ for PIMC.
13: Set $U_n := [U_{n-1} \quad \frac{r_n}{\|r_n\|}] \hat{U}$;
14: Set $U_n$ as the first $k$ columns of $U_n$ and $S_n$ as the top $k$-by-$k$ block of $\hat{S}$.
15: $n := n + 1$;
16: until termination

where $w_n = U_{n-1}^T x_n$ are the projection weights onto the span of this now tall matrix $U_{n-1}$ and $r_n = x_n - U_{n-1} U_{n-1}^T x_n$ is the residual from the projection. We only must diagonalize $\hat{S}$ to find the SVD of $X_n = [X_{n-1} \quad x_n]$. Using this particular $\hat{U}$ as the first $k + 1$ columns of the above orthogonal matrix $U_{n-1}$; note that $x_n = \hat{U} \hat{V}^T x_n$ and that

\[
\{z_n(i)\}_{i=1}^{k+1} = \hat{U}^T x_n = [w_n^T \|r_n\|]^T
\]

is the rightmost column of $\hat{S}$, and the rest of the entries $z_n(i) = 0$, $i > k + 1$. In other words, when $X_{n-1}$ is rank-$k$, solving by diagonalizing Eq. (11) is equivalent to using only the first $k + 1$ terms of Eq. (6) for computing eigenvalues. That $X_{n-1}$ is rank-$k$ is a strong assumption; however this technique is used successfully as a heuristic, by truncating smaller singular values, even when this assumption does not hold. Finally, we point out that Karasalo’s subspace averaging algorithm [15] is similar to ISVD, but it uses specific information about the noise covariance.

B. MD-ISVD, Brand’s Algorithm and PIMC

A major drawback of many linear algebraic techniques is their inapplicability to datasets with missing data. While it is not straightforward on how to adapt the ISVD for missing data, there are several different approaches in the literature [3], [14], [32]. All approaches begin in the same way, by considering how to compute two key quantities, the projection weights $w_n$ and the residual $r_n$, given missing data. Whereas for complete
data, we have $w_n = \arg\min_w \|x_n - U_{n-1}w\|^2 = U_{n-1}^Tx_n,$ for missing data one may solve
\[ w_n = \arg\min_w \|P_{\Omega_n}(x_n - U_{n-1}w)\|^2 \tag{14} \]
where $P_{\Omega_n}$ is the measurement operator with missing data as in (4). Then letting $p_n = U_{n-1}w_n$, define the residual to be
\[ r_n(i) = \begin{cases} x_n(i) - p_n(i) & \text{if } \Omega_n(i) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{15} \]
All the methods use these quantities in place of $w_n$ and $r_n$ in Algorithm 1. They also both only use (13) for the update, and once they have observed enough vectors they both truncate the $k + 1$ singular value and corresponding singular vectors.

The methods diverge only in the way they treat the singular value matrix $S_n$ in (8). Brand [14] replaces $S_n$ with $\lambda S_n$ where $0 < \lambda < 1$ is a scalar weight that diminishes the influence of previous singular values. If one takes $\lambda = 1$, this is arguably the most direct extension of ISVD to missing data, and following [3] we call this algorithm Missing Data-ISVD (MD-ISVD). Kennedy et al. [52] present a Polar Incremental Matrix Completion (PIMC) method, which weights $S_n$ with a scalar based on the norm of the data observed thus far. These are different approaches to modeling the uncertainty in the singular values arising from incomplete observations. These algorithms are together summarized in Algorithm 2. These different rules represent different trade-offs in (5).

C. Oja’s method

Oja’s method is originally proposed in 1982 [16]. It is a very popular method for streaming PCA, and recent attention has yielded great insights into its practical performance. Given an orthonormal initialization $U_0 \in \mathbb{R}^{d \times k}$, at the $n$th time, Oja’s method updates it according to the input data $x_n$ as
\[ U_n = \Pi(U_{n-1} + \eta_nx_nx_n^TU_{n-1}), \tag{16} \]
where $\Pi(W) = Q$ is an orthogonalization operator, i.e., $W = QR$ is the QR decomposition. The parameter $\eta_n$ is the step size or learning rate that may change with time.

While Oja’s method has not been derived for the missing data case in the literature, following our discussion on ISVD, one realizes that if we let $w_n = U_{n-1}^Tx_n$, the coefficient of $x_n$ in the previous estimate $U_{n-1}$, then Oja’s method is equivalent to
\[ U_n = \Pi(U_{n-1} + \eta_nx_nx_n^TU_{n-1}). \tag{17} \]
A straightforward extension in the missing data case is then to estimate the coefficient $w_n$ as (14), and to fill in the missing entries in $x_n$ as follows. Let $p_n = U_{n-1}w_n$, and the data vector can be interpolated as
\[ \tilde{x}_n = \begin{cases} x_n(i) & \text{if } \Omega_n(i) = 1 \\ p_n(i) & \text{otherwise} \end{cases}. \]
Then Oja’s update rule in the missing data case becomes
\[ U_n = \Pi(U_{n-1} + \eta_n\tilde{x}_n\tilde{x}_n^TU_{n-1}). \tag{18} \]
This algorithm is summarized in (16). Note that the original Oja’s method with full data becomes a special case of this update. We study this extension in the numerical experiments reported in Section VI.

Finally, we note that closely related to Oja’s is another method called Krasulina’s algorithm [18], which is developed for updating a rank-1 subspace with full data:
\[ U_n = U_{n-1} + \eta_n(x_nx_n^T - \frac{U_{n-1}^Tx_nx_n^TU_{n-1}}{\|U_{n-1}\|^2}I_d)U_{n-1}. \tag{19} \]
It can be viewed as a stochastic gradient descent method with the Rayleigh quotient as its objective. Oja’s method is equivalent to Krasulina’s method up to the second order terms [17], [53].

Remark 1 (Block Power Method). A block variant of Oja’s method has been developed in the literature [35], [36], [54], where it partitions the input into blocks and each time the algorithm processes one block in a way similar to Oja’s method. These methods are referred to as the block power method, or block Oja’s method, or the noisy power method. They are easier to analyze but yield suboptimal performance [39].

IV. Geometric Methods

In this section, we review subspace estimation and tracking algorithms via geometric approaches. They are developed by optimization of certain loss functions over the Riemannian or Grassmanian manifolds. Subspace tracking is enabled by optimizing a recursively updated loss function, such as the projection loss onto the subspace, as
\[ F_n(U) = \alpha_nF_{n-1}(U) + \beta_n\|x_n - P_{\mathcal{U}}(x_n)\|^2_{1/2}, \tag{20} \]
where $U \in \mathbb{R}^{d \times k}$, and $n$ is the time index, which is typically updated by using the previous estimate as a warm start. Similarly, the choice of $\alpha_n$ and $\beta_n$ balances the convergence rate (how fast it converges in static) and the tracking capability (how fast it can adapt to changes). Additionally, the step size of the gradient algorithm can also be used as a tuning knob for tracking: a more aggressive step size will adapt more quickly to new data. Given the necessity of scalable and memory-efficient algorithms, first-order and second-order stochastic gradient descent [55] are gaining a lot of popularity recently in signal processing and machine learning.
A. GROUSE

Grassmannian Rank-One Update Subspace Estimation (GROUSE) was first introduced in [21] as a gradient algorithm to build high quality subspace estimates from very sparsely sampled vectors, and has since been analyzed with fully sampled data [56], [57], noisy data [57], and missing or compressed data [56], [58]. The objective function for this gradient algorithm is given by

$$F_n(U) = \sum_{\ell=1}^{n} \| P_{\Omega_{\ell}}(x_\ell - UU^T x_\ell) \|_2^2,$$  \hspace{1cm} (21)

which is a special case of (20) with $\alpha_n = \beta_n = 1$. GROUSE implements a first-order incremental gradient procedure [59] to minimize this objective with respect to the subspace variable $U$ constrained to the Grassmannian [60], the manifold of all $U$ whose solutions can be expressed in a closed-form and efficiently rewritten as

$$\min_{U \in \mathbb{R}^{d \times k}: U^TU = I_k} F_n(U).$$

GROUSE has iteration complexity $O(dk + |\Omega_n|^2)$ at the $n$th update and so is scalable to very high-dimensional applications. The algorithm steps are described in Algorithm 4. We note that, if the step size is not given, one can use the step size prescribed in (22). This step size maximizes the per-update and so is scalable to very high-dimensional applications. GROUSE can therefore be susceptible to noise.

Algorithm 4 GROUSE [21]

1: Given $U_0$, an $d \times k$ orthonormal matrix, $0 < k < d$;
2: Optional input: Step size scheme $\eta_n > 0$
3: Set $n := 1$;
4: repeat
5: Define $w_n := \arg\min_w \| P_{\Omega_n} (x_n - U_{n-1} w_n) \|_2^2$;
6: Define $p_n := U_{n-1} w_n$;
7: if $\eta_n$ given then
8: Set $\theta_n = \eta_n \| r_n \| / \| p_n \|$
9: else
10: Set $\theta_n = \arctan \left( \| r_n \| / \| p_n \| \right)$ \hspace{1cm} (22)
11: end if
12: $U_n := U_{n-1} + (\cos(\theta_n) - 1) \frac{p_n}{\| p_n \|} \frac{w_n^T}{\| w_n \|}$
13: $+ \sin(\theta_n) \frac{r_n}{\| r_n \|} \frac{w_n^T}{\| w_n \|}$. \hspace{1cm} (23)
14: until termination

In a follow-up work, Balzano et al. [3], [51] describe SAGE GROUSE, which was derived in the context of Algorithm 2 and replaces $S_{n-1}$ with an identity matrix the same size as $S_{n-1}$, which makes the algorithm completely agnostic to singular values or the relative weight of singular vectors that have been learned. This can be considered as yet another way of modeling uncertainty in the singular values learned thus far. SAGE GROUSE has been proven to be equivalent to the GROUSE gradient algorithm for a given step-size [51], showing that indeed the distinction of “algebraic” and “geometric” algorithms is not fundamental.

Remark 2 (SNIPE). A block variant of GROUSE was presented in [61], called Subspace Navigation via Interpolation from Partial Entries (SNIPE). This algorithm partitions the input into blocks and for each block optimizes a subspace to fit the observed entries on that block but remain close to the previous subspace estimate.

B. PAST

The Projection Approximation Subspace Tracking (PAST) is proposed by Yang [25], [28] for subspace tracking with full data, which is described in Algorithm 5. PAST optimizes the following function at time $n$ without constraining $U$ to have orthogonal columns:

$$U_n = \arg\min_{U \in \mathbb{R}^{d \times k}} \sum_{\ell=1}^{n} \lambda^{n-\ell} \| x_\ell - UU^T x_\ell \|_2^2,$$ \hspace{1cm} (24)

where prior observations are discounted by a geometric factor $0 < \lambda \leq 1$. The name “projection approximation” comes from the fact that the projection onto the subspace $U$ is approximated by $UU^T$, without the constraint $U^TU = I_k$.

This sum is further approximated by replacing the second $U$ in (24) by $U_{\ell-1}$, yielding

$$U_n = \arg\min_{U \in \mathbb{R}^{d \times k}} \sum_{\ell=1}^{n} \lambda^{n-\ell} \| x_\ell - UU_{\ell-1}^T x_\ell \|_2^2.$$ \hspace{1cm} (25)

Let the coefficient vector be $w_\ell = U_{\ell-1}^T x_\ell$, then (25) can be rewritten as

$$U_n = \arg\min_{U \in \mathbb{R}^{d \times k}} \sum_{\ell=1}^{n} \lambda^{n-\ell} \| x_\ell - Uw_\ell \|_2^2,$$ \hspace{1cm} (26)

whose solution can be written in a closed-form and efficiently found via recursive least-squares. The PAST algorithm has a computational complexity of $O(dk)$. PAST has been very popular due to its efficiency, and it has been extended and modified in various ways [19], [62], [63].

Algorithm 5 PAST [25]

1: Given $U_0 \in \mathbb{R}^{d \times k}$, $R_0 = \delta I_k$;
2: Set $n := 1$;
3: repeat
4: Define $w_n := U_{n-1}^T x_n$;
5: $\beta_n = 1 + \lambda^{-2} w_n^T R_{n-1} w_n$;
6: $v_n = \lambda^{-1} R_{n-1} w_n$;
7: $R_n = \lambda^{-1} R_{n-1} - (\beta_n)^{-1} v_n v_n^T$;
8: $U_n = U_{n-1} + (x_n - U_{n-1} w_n^T) R_n w_n$;
9: $n := n + 1$;
10: until termination
Algorithm 6 PETRELS [23]

1: Given $U_0 = [\mathbf{u}_0^1, \mathbf{u}_0^2, \ldots, \mathbf{u}_0^d]^T$, and $R_0^i = \delta I_k$, $\delta > 0$ for all $i = 1, \ldots, d$.
2: Set $n := 1$;
3: repeat
4:  Define $\mathbf{w}_n := \operatorname{argmin}_\mathbf{w} \left\| P_{\Omega_n} (\mathbf{x}_n - U_{n-1} \mathbf{w}) \right\|_2^2$;
5:  for $i = 1, \ldots, d$ do
6:      $\beta_n^i = 1 + \lambda^{-1} \mathbf{w}_n^T R_{n-1}^i \mathbf{w}_n$,
7:      $\mathbf{w}_n^i = \lambda^{-1} R_{n-1}^i \mathbf{w}_n$,
8:      $R_n^i = \lambda^{-1} R_{n-1}^i - \Omega_n(i) \mathbf{w}_n^i (\mathbf{w}_n^i)^T / \beta_n^i$,
9:      $\mathbf{u}_n^i = \mathbf{u}_{n-1}^i + \Omega_n(i) \left[ \mathbf{x}_n(i) - \mathbf{w}_n^i \mathbf{u}_{n-1}^i \right] R_n^i \mathbf{w}_n$;
10:  end for
11:  $n := n + 1$;
12: until termination

C. PETRELS

The PETRELS algorithm, proposed in [22], [23], can be viewed as a modification of the PAST algorithm to handle missing data, which is summarized by Algorithm 6. PETRELS optimizes the following function at time $n$ without constraining $U$ to have orthogonal columns:

$$U_n = \operatorname{argmin}_{U \in \mathbb{R}^{d \times k}} \sum_{\ell=1}^{n} \lambda^{n-\ell} \min_{\mathbf{w}_\ell \in \mathbb{R}^k} \left\| P_{\Omega_\ell} (\mathbf{x}_\ell - U \mathbf{w}_\ell) \right\|_2^2.$$  

(27)

At each time $n$, PETRELS alternates between coefficient estimation and subspace update. We first estimate the coefficient vector by minimizing the projection residual using the previous subspace estimate:

$$\mathbf{w}_n = \operatorname{argmin}_w \left\| P_{\Omega_n} (\mathbf{x}_n - U_{n-1} \mathbf{w}_n) \right\|_2^2,$$  

(28)

where $U_0 \in \mathbb{R}^{d \times k}$ is a random subspace initialization. The subspace $U$ is then updated by minimizing

$$U_n = \operatorname{argmin}_U \sum_{\ell=1}^{n} \lambda^{n-\ell} \left\| P_{\Omega_\ell} (\mathbf{x}_\ell - U \mathbf{w}_\ell) \right\|_2^2,$$  

(29)

where $\mathbf{w}_\ell, \ell = 1, \ldots, n$ are estimates from (28). The objective function in (29) decomposes into a parallel set of smaller problems, one for each row of $U_n = [\mathbf{u}_1^1, \mathbf{u}_1^2, \ldots, \mathbf{u}_1^k]^T$, where $\mathbf{u}_1^i \in \mathbb{R}^k$. Thus the $i$th row can be estimated by solving

$$\mathbf{u}_n^i = \operatorname{argmin}_{\mathbf{u}_i \in \mathbb{R}^k} \sum_{\ell=1}^{n} \lambda^{n-\ell} \Omega_\ell(i) (\mathbf{x}_\ell(i) - \mathbf{w}_\ell^T \mathbf{u}_i)^2$$

$$= \mathbf{u}_{n-1}^i + \Omega_\ell(i) \cdot \left[ \mathbf{x}_\ell(i) - \mathbf{w}_\ell^T \mathbf{u}_{n-1}^i \right]$$

$$\cdot \left( \sum_{\ell=1}^{n} \lambda^{n-\ell} \Omega_\ell(i) \mathbf{w}_\ell \mathbf{w}_\ell^T \right)^{-1} \mathbf{w}_n,$$  

(30)

for $i = 1, \ldots, d$. Again, the problem can be solved efficiently via recursive least-squares. Moreover, PETRELS can be made very efficient by parallelizing the implementation of (30).

Both PAST and PETRELS can be regarded as applying second-order stochastic gradient descent [55] to the loss function, and each step of the update is approximately a Newton step. Therefore, it is expected that the algorithm will converge quadratically when it is close to the optimal solution. Several algorithms can be developed along similar lines of PETRELS, where the loss function is revised to include regularization terms on the Frobenius norms of the subspace $U$ and the weight vector $\mathbf{w}_n$, which we refer the readers to [26], [27].

V. Performance Analysis

In this section we will describe general analysis methodologies as well as specific theoretical results for characterizing the performance of the aforementioned streaming PCA and subspace tracking algorithms.

To carry out the analysis, we need to make assumptions on how the data are generated. A popular approach that has been taken in the literature is to assume that each data vector is generated according to the following “spiked model” [64]:

$$\mathbf{x}_n = U^* \mathbf{a}_n + \sigma \mathbf{e}_n,$$  

(31)

where $U^*$ is a deterministic $d \times k$ matrix, $\mathbf{a}_n$ is a random signal vector with covariance matrix $\Sigma_\mathbf{a}$, and $\mathbf{e}_n$ is the noise vector. For simplicity, we assume that the covariance matrix of $\mathbf{e}_n$ is the identity matrix $\mathbf{I}_k$, and we use $\sigma$ to denote the noise level. This model arises in applications such as array signal processing, where $U^*$ is the “steering matrix” and $k$ denotes the number of targets to be tracked by the array. The generative model (31) can also be seen as a special case of the probabilistic model described in Section II-A, since $\mathbb{E}[\mathbf{x}_n, \mathbf{x}_n^T] = U^* \Sigma_\mathbf{a} U^* + \sigma^2 \mathbf{I}_k$ is a sum of an exact low-rank matrix and a full-rank identity matrix.

A. Classical Asymptotic Analysis

Historically, the first analysis of subspace tracking algorithms was done in the asymptotic regime (see, e.g., [28], [29]), where the algorithms are shown to converge, in the small step size limit, to the solution of some deterministic Ordinary Differential Equations (ODEs).

To understand the basic ideas underlying such analysis, we note that the essence of almost all the online algorithms described in Section III and Section IV is a stochastic recursion of the form

$$U_n = U_{n-1} + \eta_n Q(U_{n-1}, \mathbf{y}_n).$$  

(32)

Here, $U_n$ is the estimate at time $n$, $Q(\cdot, \cdot)$ is some nonlinear function of the previous estimate $U_{n-1}$ and new measurement $\mathbf{y}_n$, and $\eta_n$ is the step size (i.e., the learning rate). For example, Krasulina’s method given in (19) is just a special case of (32). When the step size $\eta_n$ is small, we can perform Taylor’s expansion on the recursion formulas of Oja’s method (16) and GROUSE (23), and show that these two algorithms can also be written in the form of (32) after omitting higher-order terms in $\eta_n$.

Under the statistical model (31), the general algorithm (32) is simply a Markov chain with state vectors $U_n \in \mathbb{R}^{d \times k}$. The challenge in analyzing the convergence of (32) comes from the nonlinearity in the function $Q(\cdot, \cdot)$. In the literature, a very powerful analytical tool is the so-called ODE method. It was introduced to the control and signal processing communities by Ljung [30] and Kushner [31] in the 1970s, and similar approaches have an even longer history in the literature of
statistical physics and stochastic processes (see, e.g., [65], [66] for some historical remarks.)

The basic idea of the ODE method is to associate the discrete-time stochastic process (32) with a continuous-time deterministic ODE. Asymptotically, as the step size $\eta_n \to 0$ and the number of steps $n \to \infty$, the process (32) can be shown to converge to the solution of the ODE. Specifically, we let the step sizes be such that

$$\sum_{n=1}^{\infty} \eta_n = \infty \text{ and } \sum_{n=1}^{\infty} \eta_n^2 < \infty.$$ 

For example, a popular choice is to choose $\eta_n = c/n$ for some $c > 0$. By defining $t_n = \sum_{k \leq n} \eta_k$ as the “fictitious” time, we can convert the discrete-time process $U_n$ to a continuous-time process $U_t$ via linear interpolation:

$$U_t = U_{n-1} + \frac{t-t_{n-1}}{t_n-t_{n-1}}(U_n-U_{n-1}), \quad t_{n-1} \leq t \leq t_n. \quad (33)$$

Under mild regularity conditions on the function $Q(\cdot, \cdot)$, one can then show that, as $t \to \infty$, the randomness in the trajectory of $U_t$ will diminish and $U_t$ will converge to the deterministic solution of an ODE [30], [31].

Although a rigorous proof of the above convergence is technical, the limiting ODE, if the convergence indeed holds, can be easily derived, at least in a nonrigorous way. To start, we can rewrite (32) as

$$\frac{U_n - U_{n-1}}{\eta_n} = E_{y_n|U_{n-1}}[Q(U_{n-1}, y_n)] + m_n, \quad (34)$$

where $E_{y_n|U_{n-1}}[\cdot]$ denotes the conditional expectation of $y_n$ given $U_{n-1}$, and $m_n$ captures the remainder terms. From the construction of $U_t$ in (33), the left-hand side of (34) is equal to $(U_{n-1}+\eta_n-U_{n-2})/\eta_n$, which converges to $d/dt U_t$ since the step size $\eta_n \to 0$. Moreover, one can show that the remainder $m_n$, often referred to as the martingale difference term, is of order $o(1)$. It follows that we can write the limit form of (34) as an ODE

$$\frac{d}{dt} U_t = h(U_t), \quad (35)$$

where $h(U_t) = E_{y|U}[Q(U, y)]$.

The ODE approach is a very powerful analysis tool. By studying the fixed points of the limiting dynamical system in (35), we can then draw conclusions about the convergence behavior of the original stochastic process (32). This approach was taken in [67], where the author used an ODE analysis to show that the PAST algorithm [25] globally converges to the target signal subspace $U^*$ with probability one. This result was later adapted in [23] to analyze PETRELS for the fully observed case.

### B. Asymptotic Analysis in the High-Dimensional Regime

Despite its versatility and strong theoretical value, the above classical asymptotic approach has several limitations: First, the analysis requires the step size $\eta_n$ to tend to zero as $n \to \infty$. While using a decreasing sequence of step sizes $\eta_n$ helps the stochastic algorithm to converge to the globally optimal solution, it is not a good strategy for applications where the target low-dimensional subspace can be time-varying. In that scenario, a small but fixed step size is often more preferable, as it would make the algorithms more nimble in tracking the changing subspace. Second, the classical asymptotic analysis leads to an ODE with $O(d)$ variables. In modern applications, the number of variables, i.e., $d$ can be very large, making it less practical to numerically solve the ODE.

In what follows, we briefly review a different asymptotic analysis approach [32]–[34] that addresses the above problems. For simplicity, we present the underlying idea using the example of Oja’s method (16) for learning a one-dimensional subspace using full data, although the same approach applies to the general rank-$k$ case with missing data.

When $k = 1$, the orthogonalization operator $\Pi$ in (16) is just a norm normalization step, and thus the update rule can be simplified as

$$U_n = \frac{U_{n-1} + \eta_n x_n \sigma_n^2 U_{n-1}}{\|U_{n-1} + \eta_n x_n \sigma_n^2 U_{n-1}\|} \quad (36)$$

This stochastic process is a Markov chain in $\mathbb{R}^d$, where the dimension $d$ can be large. To reduce the underlying dimension of the system we need to analyze, we note that the quality of the estimate $U_n$ can be fully captured by a scalar quantity

$$s_n \defn \frac{U_n^T U^*}{\|U_n\| \|U^*\|}. \quad (37)$$

Clearly, $s_n \in [-1, 1]$, with $s_n = \pm 1$ indicating perfect alignment of $U^*$ and $U_n$. In what follows, we refer to $s_n$ as the cosine similarity.

Substituting (36) and (31) into (37), we get a recursion formula for the cosine similarity:

$$s_{n+1} = \frac{s_n + \eta_n a_n (a_n^T Q n + \sigma q_n)}{(1 + \eta_n (a_n s_{n-1} + \sigma q_n)^2 \|a_n + \sigma \|\|e_n\|^2 + 2 \sigma a_n p_n)^{1/2}} \quad (38)$$

where $a_n$, $\epsilon_n$ are the signal and noise vector in the generating model (31), respectively, and $p_n \defn e_n^T U^*$ and $q_n \defn e_n^T U_{n-1}$. The expression (38) might appear a bit complicated, but the key observation is the following: If the noise vector $\epsilon_n$ is drawn from the normal distribution $\mathcal{N}(0, I_d)$, then it follows from the rotational symmetry of the multivariate normal distribution that

$$P(p_n, q_n | s_{n+1}) \sim \mathcal{N} \left(0, \begin{bmatrix} 1 & s_{n-1} \\ s_{n-1} & 1 \end{bmatrix} \right).$$

In other words, given $s_{n-1}$, the two random variables $p_n$ and $q_n$ are joint normal random variables whose distribution is a function of $s_{n-1}$. Consequently, the recursion (38) from $s_{n-1}$ to $s_n$ forms a one-dimensional Markov chain. Note that this exact Markovian property relies on the assumption that the noise vector $\epsilon_n$ be normally distributed. However, due to central limit theorem, we can show that this property still holds asymptotically, when the underlying dimension $d$ is large and when the elements of $\epsilon_n$ are independently drawn from more general distributions with bounded moments.

Further analysis shows that, by choosing the step size $\eta_n = \tau/d$ for some fixed $\tau > 0$, we can apply the similar
ODE idea used in the classical asymptotic analysis to obtain a deterministic, limit process for the cosine similarity. More specifically, we can show that a properly time-rescaled version of \( s^{(d)}(t) \) will converge weakly, as \( d \to \infty \), to a deterministic function that is characterized as the unique solution of an ODE (see [32]–[34] for details).

In [33], the exact dynamic performance of GROUSE and PETRELS was analyzed in this asymptotic setting for \( k = 1 \). For the latter algorithm, it turns out that we just need to study two scalar processes: the cosine similarity \( s_n \) as defined in (37) and an auxiliary parameter \( g_n = (\| \mathbf{U}_n \|^2 R_n)^{-1} \), where \( R_n \) is the average of the correlations \( R_n \) in Algorithm 6. By introducing the rescaled “fictitious” time \( t = n/d \), we can embed the discrete-time sequences \( s_n, g_n \) into continuous-time as \( s^{(d)}(t) = s_{[td]} \) and \( g^{(d)}(t) = g_{[td]} \). As the underlying dimension \( d \to \infty \), we can show that the stochastic processes \( \{ s^{(d)}(t), g^{(d)}(t) \} \) converges weakly to the unique solution of the following systems of coupled ODEs:

\[
\begin{align*}
\frac{ds(t)}{dt} &= 2\alpha s(1-s)g - \sigma^2 f(\alpha s + \sigma^2)g^2 \\
\frac{dg(t)}{dt} &= -g^2(\sigma^2 g + 1)(\alpha s + \sigma^2) + \mu g,
\end{align*}
\]

(39)

where \( \alpha \) is the probability with which each coordinate of the data vectors can be observed, and \( \mu > 0 \) is a constant such that the discount parameter \( \lambda \) in (27) is set to \( \lambda = 1 - \frac{\mu}{n} \).

Compared to the classical ODE analysis [23], [67] which keeps the ambient dimension \( d \) fixed and studies the asymptotic limit as the step size tends to 0, the ODEs in (39) only involve 2 variables \( s(t) \) and \( g(t) \). This low-dimensional characterization makes the new limiting results more practical to use, especially when the dimension is large.

Similar asymptotic analysis can also be carried out for GROUSE to show that the time-varying cosine similarity \( s_n \) associated with the algorithm converges, as \( d \to \infty \), to the solution of a limiting ODE:

\[
\frac{ds}{dt} = \tau(2\alpha - \tau \sigma^4)s - \alpha \tau(2 + \tau \sigma^2)s^2,
\]

(40)

where \( \tau > 0 \) is a constant such that the step size parameter \( \eta_n \) used in Algorithm 4 is \( \eta_n = \tau/d \), and \( \alpha \) is again the subsampling probability. Numerical verifications of the asymptotic results are shown in Figure 1. We can see that the theoretical prediction given by the ODEs (39) and (40) can precisely characterize the actual dynamic performance of the PETRELS and GROUSE algorithms.

The convergence behavior of the algorithms can also be established by analyzing the fixed points of the dynamical system associated with the limiting ODEs. For example, by studying the stability of the fixed points of (39) for PETRELS, one can show that \( \lim_{n \to \infty} s(t) > 0 \) if only if

\[
\mu < (2\alpha/\sigma^2 + 1/2)^2 - 1/4,
\]

(41)

where \( \alpha \in (0, 1) \) is the probability with which each element of the data vector \( x_n \) is observed. A “non-informative” solution corresponds to \( s(t) = 0 \), in which case the estimate \( \mathbf{U}_n \) and the underlying subspace \( U^* \) are orthogonal (i.e., uncorrelated.) The expression in (41) predicts a sharp phase transition phenomenon for PETRELS, where a critical choice of \( \mu \) (as a function of \( \sigma \) and \( \alpha \)) separates informative solutions from non-informative ones. This prediction is confirmed numerically in Figure 2.

C. Finite Sample Analysis

In addition to the asymptotic analysis described in the previous subsections, there have also been many recent efforts in establishing finite-sample performance guarantees for various streaming PCA algorithms. We begin with analysis in the case of fully observed data vectors.

One of the earlier works is [35], where the authors analyze a block variant of Oja’s method: within each iteration, multiple sample vectors are drawn, whose empirical covariance matrix is then used in place of \( \mathbf{X}_n \mathbf{X}_n^T \) in (36). Under the generative model (31), the authors show that this algorithm can reach accuracy \( \| \mathbf{U}_n - \mathbf{U}^* \| \leq \varepsilon \), in the rank-one case, if the total number of samples is of order

\[
n = O \left( \frac{(1 + 3(\sigma + \sigma^2)\sqrt{d})^2 \log(d/\varepsilon)}{\varepsilon^2 \log((\sigma^2 + 3/4)/\sigma^2 + 1/2)} \right),
\]
where $\sigma$ is the noise level in (31). Similar analysis is also available for the general rank-$k$ case. Block Oja’s methods have also been studied in [36], [54], but the analysis is done under a model much more general than (31): the data vectors $x_n$ are assumed to be drawn i.i.d. from a general distribution on $\mathbb{R}^d$ with zero-mean and covariance $\Sigma$.

The performance of Oja’s original method has also been studied under the above general model. For the rank-1 case, Li et al. [37] established that the error is on the order of $O\left(\frac{\lambda_1+\lambda_2}{\lambda_1-\lambda_2} d \log n \right)$, which is near optimal up to a logarithmic factor, with the step size $\eta_n = \frac{2\log n}{\lambda_1-\lambda_2}$. Similarly, Jain et al. [68] also provides a near-optimal result for Oja’s method, with a time-varying learning rate $\eta_n = \frac{\lambda_1-\lambda_2}{\lambda_1-\lambda_2} + n$, where $n_2$ is some starting time. Other results include Balsubramani et al. [17], Shamir [38], Li et al. [69]. Allen-Zhu and Li [39] provides a near-optimal guarantee for rank-$k$ Oja’s method very recently. This paper also contains a comprehensive table that summarizes many recent results.

Most of the existing analysis and performance guarantees in the literature assume that there is a positive gap between $\lambda_k$ and $\lambda_{k+1}$. This eigenengap assumption was removed in Shamir [38] and Allen-Zhu and Li [39], where the authors provide sample complexity bounds that are gap-free, i.e. they do not require a strictly positive eigengap.

Also for fully observed vectors, the global convergence of the GROUSE algorithm is established in [57], [58] under the generative model (31) for the special noise-free case, i.e., $\sigma = 0$. Let $\epsilon^* > 0$ be the desired accuracy of our estimated subspace as measured by the sum of the squared principal angles [i.e., a rank-$k$ generalization of the cosine similarity (37)]. Initialize the starting point $(U_0)$ of GROUSE as the orthonormalization of an $d \times k$ matrix with entries being standard normal variables. Then for any $\rho > 0$, after

\[ n \geq n_1 + n_2 = \left(\frac{2k^2}{\rho} + 1\right) \mu_0 \log(d) + 2k \log \left(\frac{1}{2\rho(1-\epsilon^*)}\right) \]

iterations of GROUSE Algorithm 4, $\epsilon_n \leq \epsilon^*$ with probability at least $1 - 2\rho$, where $\mu_0 = 1 + \frac{k \log d}{\log k \log d}$ with $C > 0$ a constant approximately equal to 1. This result is divided into two parts: $n_1$ is the bound on the number of observed vectors (also iterations) required to get to a basin of attraction, and $n_2$ is the number required for linear convergence in a local region of the minimizer. In practice we see that the bound on $n_1$ is loose but $n_2$ is accurate when discarding dependence on $\rho$ [57], [58].

In contrast to the fully observed case, the literature is much sparser for finite-sample analysis in the missing data case. Expected improvement at each iteration for the GROUSE algorithm has been bounded given noisy data [57] and missing or compressed data [56], [58]. These results for missing or compressed data can be generalized to a local convergence result [56], [58]. It remains an important open problem to establish finite sample global performance guarantees for GROUSE and other algorithms such as Oja’s and PETRELS in the missing data case.

VI. Numerical Experiments

We benchmark the performance of several competitive algorithms reviewed in this paper that are able to handle missing data, including GROUSE [21], PETRELS [23], PIMC [52], MD-ISVD [3], Brand’s algorithm [14] and Oja’s algorithm [16] adapted to the missing data case as proposed in the current paper in (18).

A. Simulation Setup

Let $d = 200$ and $k = 10$. We select the ground truth subspace as $U = \text{orth}(U) \in \mathbb{R}^{d \times k}$ where $U$ is composed of standard i.i.d. Gaussian entries. The coefficient vectors are generated as $a_n \sim N(0, \text{diag}(\epsilon))$, where the loading vector $\epsilon$ is given as

\[ \epsilon = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]^T, \]

for a well-conditioned system, and

\[ \epsilon = [1, 1, 1, 1, 0.3, 0.3, 0.3, 0.1, 0.1]^T \]

for an ill-conditioned system. The data vector is generated using Eq. (31) as $x_n = U a_n + \epsilon_n$, where $\epsilon_n$ is composed of i.i.d. $N(0, \sigma^2)$ entries, where the noise level is given as $\sigma = [10^{-2}, 10^{-5}, 0]$. Each vector is observed with a fixed percent $\alpha = [0.1, 0.5, 1]$ of entries selected uniformly at random. We note that the lowest sampling fraction is near the information theoretic lower bound of number of uniform random samples that will guarantee matrix reconstruction. The reconstruction error is calculated as the projection error $\|U - \hat{U}\|^2$, where $\hat{U}$ is the estimated (orthogonalized) subspace. All the algorithms are initialized with the same orthogonalized random subspace. We assume all algorithms are given the true rank. Throughout the simulations, we set the discount parameter $\eta = 0.98$ in PETRELS and $\beta = 0.98$ in Brand’s algorithm, and set $\eta = 0.5$ in Oja’s algorithm. Note that these parameters are not perfectly tuned, and indeed picking them differently will likely yield different trade-offs in performance. Our goal is to illustrate their typical behaviors without claiming relative superiority.

B. Performance Evaluations

We first examine the performance of subspace estimation with streaming observations. Figure 3 shows the reconstruction error of the algorithms with respect to the number of snapshots for both the well-conditioned case and the ill-conditioned case, where every data vector is only observed and processed once. It can be seen that as the fraction of missing data increases, more snapshots are needed for the algorithms to converge.

We next examine the performance of subspace estimation in a multi-pass streaming PCA setting. We assume there is a fixed number of $6000$ partially observed data vectors, and in each iteration one vector is accessed at random to update the subspace estimation. Indeed, this setup is equivalent to solving a low-rank matrix completion problem. Therefore, faster algorithms may access the data set in more passes in a given amount of time. Figure 4 shows the reconstruction error of the algorithms with respect to the wall-clock time for both the well-conditioned case and the ill-conditioned case.
Note that no attempt is made to speed up algorithms such as PETRELS using parallel computing. Again, while many algorithms converge very fast with complete data, they may take significantly longer in the presence of missing data.

Next, we examine the performance of the algorithms on subspace tracking, where we assume there is abrupt change in the underlying subspace and the goal is to examine how fast the algorithms are able to track the change. Moreover, we generate the loading vector $c$ before and after the same with entries drawn from a uniform distribution in $[0,1]$. The noise level is set as $\sigma = 10^{-5}$ and the fraction of observation is set as $\alpha = 0.3$. Figure 5 shows the performance of the algorithms when the subspace changes abruptly at the 4000th snapshot.

VII. CONCLUDING REMARKS

In this paper, we have reviewed a variety of streaming PCA and subspace tracking algorithms, focusing on the ones that have a near-linear memory complexity and computational time, appealing convergence rates, and can tolerate missing data. Convergence guarantees based on classical and
new asymptotic analysis as well as recent finite-sample non-asymptotic analysis are discussed. While we divide our discussions into algebraic and geometric approaches, it is important to point out that these approaches are not actually distinct. For example, Oja’s is equivalent to Krasulina’s, a stochastic gradient descent method, by ignoring second-order terms; GROUSE is equivalent to a version of Incremental SVD that is agnostic to past singular values [51]; and the PAST algorithm can also be interpreted from both perspectives [20]. It is an exciting open question to understand the connections between algebraic and geometric methods more generally.

Due to space limitations, we have focused on the problem of estimating a single low-dimensional subspace under the squared loss for streaming data with possibly missing entries, which is most suitable when the noise is modeled as Gaussian. There are many important extensions, motivated by real-world applications, including subspace tracking for non-Gaussian data [70], [71], tracking a union-of-subspace model [72], [73], tracking a low-dimensional subspace with multi-scale representations [74], and subspace tracking in the presence of missing data [75].

Fig. 4: Reconstruction error versus the wall-clock time for various algorithms in both well-conditioned and ill-conditioned cases for multi-pass streaming PCA on a finite data set with 6000 snapshots.
Fig. 5: Reconstruction error versus the number of observed snapshots with an abrupt subspace change.


Laura Balzano
Laura Balzano is an assistant professor in Electrical Engineering and Computer Science at the University of Michigan. She is an Intel Early Career Faculty Fellow. She received her Ph.D. in Electrical and Computer Engineering from the University of Wisconsin, Madison supported by a 3M Fellowship. At Wisconsin she also won the ECE Best Dissertation Award. Her main research focus is online learning and optimization for low-rank models, with high-dimensional data that are highly incomplete or corrupted. She works on applications in networks, environmental monitoring, and computer vision.
Yuejie Chi (S’09-M’12-SM’17) received the Ph.D. degree in Electrical Engineering from Princeton University in 2012, and the B.E. (Hon.) degree in Electrical Engineering from Tsinghua University, Beijing, China, in 2007. Since January 2018, she is an Associate Professor with the department of Electrical and Computer Engineering at Carnegie Mellon University, after spending five years at The Ohio State University. She is the recipient of the IEEE Signal Processing Society Young Author Best Paper Award in 2013 and the Best Paper Award at the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) in 2012. She received the NSF CAREER Award in 2017, Young Investigator Program Awards from AFOSR and ONR in 2015, Ralph E. Powe Junior Faculty Enhancement Award from Oak Ridge Associated Universities in 2014, Google Faculty Research Award in 2013, and Roberto Padovani scholarship from Qualcomm Inc. in 2010. She is an Elected Member of the MLSP and SPTM Technical Committees of the IEEE Signal Processing Society since January 2016. Her research interests include signal processing, machine learning, large-scale optimization, and their applications in data science, inverse problems, imaging, and sensing systems.

Yue M. Lu did his undergraduate studies at Shanghai Jiao Tong University, China. He then attended the University of Illinois at Urbana-Champaign, where he received the M.Sc. degree in mathematics and the Ph.D. degree in electrical engineering, both in 2007. Following his work as a postdoctoral researcher at the Audiovisual Communications Laboratory at Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, he joined Harvard University in 2010, where he is currently an Associate Professor of Electrical Engineering at the John A. Paulson School of Engineering and Applied Sciences.

He received the Most Innovative Paper Award of IEEE International Conference on Image Processing (ICIP) in 2006, the Best Student Paper Award of IEEE ICIP in 2007, and the Best Student Presentation Award at the 31st SIAM SEAS Conference in 2007. Student papers supervised and coauthored by him won the Best Student Paper Award of IEEE International Conference on Acoustics, Speech and Signal Processing in 2011, the Best Student Paper Award of IEEE Global Conference on Signal and Information Processing (GlobalSIP) in 2014, and the Best Student Paper (First Prize) of the IEEE CAMSAP Workshop in 2017. He is a recipient of the 2015 ECE Illinois Young Alumni Achievement Award.

He has been an Associate Editor of the IEEE Transactions on Image Processing, and he has served as an Elected Member of the IEEE Image, Video, and Multidimensional Signal Processing Technical Committee, and an Elected Member of the IEEE Signal Processing Theory and Methods Technical Committee. His research interests include theoretical and algorithmic aspects of high-dimensional statistical signal and information processing, imaging, and image processing.