POLITECNICO DI TORINO
Repository ISTITUZIONALE

Streaming Algorithms for Subspace Analysis: Comparative Review and Implementation on IoT Devices

Original
Streaming Algorithms for Subspace Analysis: Comparative Review and Implementation on IoT Devices / Marchioni, A., Prono, L., Mangia, M., Pareschi, F., Rovatti, R., Setti, G.. - In: IEEE INTERNET OF THINGS JOURNAL. - ISSN 2327-4662. - STAMPA. - 10:14(2023), pp. 12798-12810. [10.1109/JIOT.2023.3256529]

Availability:
This version is available at: 11583/2980059 since: 2023-07-07T15:45:52Z

Publisher:
Institute of Electrical and Electronics Engineers Inc.

Published
DOI:10.1109/JIOT.2023.3256529

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2023 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

19 November 2026

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529

Streaming Algorithms for Subspace Analysis: Comparative Review and Implementation on IoT Devices

Alex Marchioni, *Student Member, IEEE,* Luciano Prono, *Student Member, IEEE,* Mauro Mangia, *Member, IEEE,* Fabio Pareschi, *Senior Member, IEEE,* Riccardo Rovatti, *Fellow, IEEE,* and Gianluca Setti, *Fellow, IEEE*

***Abstract*—Subspace analysis is a widely used technique for coping with high-dimensional data and is becoming a fundamental step in the early treatment of many signal processing tasks. However, traditional subspace analysis often requires a large amount of memory and computational resources, as it is equivalent to eigenspace determination. To address this issue, specialized *streaming* algorithms have been developed, allowing subspace analysis to be run on low-power devices such as sensors or edge devices.**

Here, we present a classification and a comparison of these methods by providing a consistent description and highlighting their features and similarities. We also evaluate their performance in the task of subspace identification with a focus on computational complexity and memory footprint for different signal dimensions. Additionally, we test the implementation of these algorithms on common hardware platforms typically employed for sensors and *edge* devices.

***Index Terms*—Principal Components Analysis, Principal Subspace, Minor Component Analysis, Minor Subspace, Streaming Algorithms, Edge computing.**

I. Introduction

THE Internet of Things (IoT) covers the set of technologies enabling the interconnection between people, devices and virtual entities to offer a growing plethora of efficient and reliable services. Among these technologies, Subspace Analysis (SA) plays a key role in sensing [1]–[4], wireless communication [5]–[7], data compression [8], data analysis [9], [10], and anomaly detection [11]–[14].

A. Marchioni, M. Mangia, and R. Rovatti are with the Department of Electrical, Electronic, and Information Engineering, University of Bologna, 40136 Bologna, Italy, and also with the Advanced Research Center on Electronic Systems, University of Bologna, 40125 Bologna, Italy (e-mail: alex.marchioni@unibo.it, mauro.mangia@unibo.it, riccardo.rovatti@unibo.it).

L. Prono is with the Department of Electronics and Telecommunications (DET), Politecnico di Torino, 10129 Torino, Italy (e-mail: luciano.prono@polito.it).

F. Pareschi is with the Department of Electronics and Telecommunications (DET), Politecnico di Torino, 10129 Torino, Italy, and also with the Advanced Research Center on Electronic Systems (ARCES), University of Bologna, 40125 Bologna, Italy (e-mail: fabio.pareschi@polito.it).

G. Setti is with CEMSE, King Abdullah University of Science and Technology (KAUST), Saudi Arabia (e-mail: gianluca.setti@kaust.edu.sa). He is also on leave from the Department of Electronics and Telecommunications (DET), Politecnico di Torino, 10129 Torino,

This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

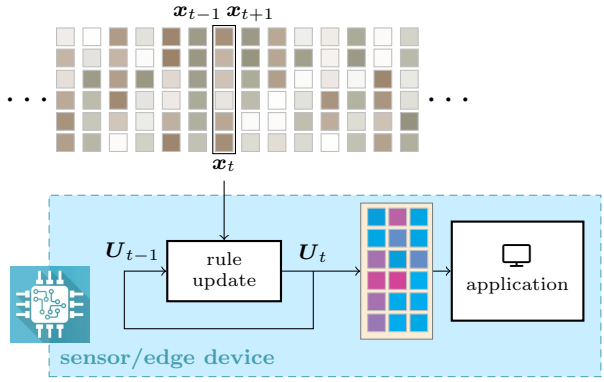


Fig. 1. General scheme of a streaming algorithm for on-board subspace analysis. Input vectors $\boldsymbol{x}_t$ are locally processed one at a time to update the current estimate of the target subspace represented by the span of the columns in the matrix $\boldsymbol{U}_t$. This matrix $\boldsymbol{U}_t$ then feeds the final application.

SA considers the space in which the signal lives and finds a subspace with some distinctive properties. In general, the information contained in the signal is assumed to have a lower dimensionality so that the signal space can be split into principal and minor/noise subspaces. Principal component/subspace analysis (PCA/PSA) deals with the subspace on which the projection of the signal preserves most of the original energy, while its orthogonal counterpart is the object of minor component/subspace analysis (MCA/MSA).

The identification of these subspaces is a building block in all mentioned application domains. For instance, representing the signal information in a lower-dimensional space hinges on the possibility of signal compression [8]. Monitoring how the signal either distributes its energy over the principal components or the unexpected presence of energy in the minor subspace may reveal the presence of anomalies [11]–[14]. In the case of Code Division Multiple Access systems, it has been proven that the identification of the principal subspace of the received signal allows the design of a simpler and better performing receiver compared to classical detector [5]. In the field of sensing, in particular, in the estimation of the direction of arrival, there are two well-known methods, ESPRIT [15] and MUSIC [16] that are the basis of a variety of innovative approaches [1]–[4], where the former requires PSA and the latter MSA.

Subspace analysis generally involves two stages: the identification of the vectors representing the target subspace, and

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529

the signal projection over those vectors. While the latter is just a linear transformation, subspace identification is traditionally solved through methods requiring the acquisition and storage of the entire dataset, which is processed all at once. These are commonly referred to as *batch* methods and consist of either the eigenvalue decomposition (EVD) or the singular-value decomposition (SVD) [17]. Both imply extensive memory usage and have a complexity in the order of at least $\mathcal{O}(n^2k)$, where n is the original dimension of a data example and k is the dimension of the sought subspace. Hence, in applications involving devices with limited storage and computational capabilities, approaches that drastically reduce the demand for resources are needed.

In this light, several algorithms (e.g. [18]–[20]) have been presented that treat data as a stream, hence data is processed as it is available with no need for extensive storage. This characteristic leads to a drastic reduction of the computational complexity up to $\mathcal{O}(nk^2)$ or even $\mathcal{O}(nk)$ at the price of some approximation. These *streaming* algorithms potentially allow SA to be run on devices with strict constraints on computational resources, such as sensors or edge devices. Fig. 1 depicts a possible scenario in which a stream of vectors $\boldsymbol{x}_t$ is locally processed to update at each iteration the matrix $\boldsymbol{U}_t$ whose columns represent the current estimate of the vectors spanning the target subspace.

In practical scenarios, the capability of a streaming method to correctly identify a subspace depends on several aspects that are generally connected to a mismatch between the signal model and acquired data. For example, an acquisition system may incur missing or corrupted data. If these are rare events, simple solutions are possible, as in the case of corrupted data for which the addition of a simple preprocessing stage discarding these occurrences may be sufficient. Conversely, specific strategies must be adopted, as in [18] for the missing data case. Another example concerns the method's convergence when the signal is more complex than a mere expansion of a k-dimensional source of information. How this deviation of the signal from the theoretical model affects convergence can be measured with the *eigengap* as analyzed in [21], [22].

In this paper, we focus on a scenario where the above issues are either not present or suitably solved by one of the above-mentioned techniques. Conversely, we focus on the aspects related to the implementation of streaming SA algorithms on devices having constraints in computational resources typical of sensor or edge devices. With respect to this, our contribution is three-fold. We compare the selected methods in a coherent framework in terms of functionality (accuracy to identify the target subspace) and resource needs. We discuss the deployment on commercial sensor/edge devices distinguishing high-end from low-end, e.g., systems equipped with multi-core ARM Cortex-A family processors and few GB of memory against microcontrollers featuring Cortex-M family processors and few MB of memory.

The paper is organized as follows. Section II proposes a classification for the different streaming methods while their description is in Section III along with a discussion about their features and relationship with each other. Section IV reports the performance of the considered methods and Section V presents a discussion about the deployment on commercial sensor/edge devices. In drawing the conclusion, we summarize the achieved result and present the *lesson learnt*.

II. Methods classification

We model the data stream as a discrete-time stochastic process that generates occurrences $\boldsymbol{x}_t \in \mathbb{R}^n$ with $t = 1, 2, \ldots$, where each $\boldsymbol{x}_t$ may contain readings from either a single physical quantity or a collection of samples coming from different sensors. The algorithms we describe process vectors $\boldsymbol{x}_t$ sequentially to extract a characterization of the whole data stream. We assume the statistical characterization of $\boldsymbol{x}_t$ to be constant or slowly variant, such that, $\boldsymbol{x}$ can represent any possible $\boldsymbol{x}_t$. We also limit the analysis to the case $\mathbf{E}[\boldsymbol{x}] = \mathbf{0}$, where $\mathbf{0}$ is the n-dimensional null vector and $\mathbf{E}[\cdot]$ indicates the expectation operator. As a result, the covariance/correlation matrix is $\boldsymbol{\Sigma} = \mathbf{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right]$, where $\cdot^\top$ indicates transposition.

As for any correlation matrix, $\boldsymbol{\Sigma}$ has an EVD $\boldsymbol{\Sigma} = \boldsymbol{\Psi}\boldsymbol{\Lambda}\boldsymbol{\Psi}^\top$, $\boldsymbol{\Lambda}$ is a diagonal matrix with diagonal entries $\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{n-1} \geq 0$, and the columns of $\boldsymbol{\Psi} \in \mathbb{R}^{n\times n}$ are the corresponding eigenvectors $\{\boldsymbol{\psi}_0, \ldots, \boldsymbol{\psi}_{n-1}\}$ in orthonormal form.

Streaming algorithms aim to identify a matrix $\boldsymbol{U} \in \mathbb{R}^{n\times k}$ that is tied to either the so-called *principal subspace*, i.e., the one spanned by the $m < n$ eigenvectors associated with the largest eigenvalues, or the so-called *minor subspace* or *noise subspace*, i.e., the one spanned by the remaining $n - m$ eigenvectors associated with the smallest eigenvalues. Note that $k = m$ or $k = n - m$ for the principal or the minor case. To identify the target matrix $\boldsymbol{U}$, streaming algorithms follow an iterative procedure that updates an estimation $\boldsymbol{U}_t$ every time a new vector $\boldsymbol{x}_t$ is acquired such that $\boldsymbol{U}_t \to \boldsymbol{U}$ when t grows.

The considered methods estimate $\boldsymbol{U}$ in different ways. Here we list some key features that can make distinctions or draw connections.

1) principal and minor subspaces: Most algorithms target the principal subspace. Some of them can also target the minor subspace, and few methods are designed for the minor subspace only.

2) eigenvectors or subspaces: When the target matrix $\boldsymbol{U}$ is $\boldsymbol{\Psi}_{|m} = [\boldsymbol{\psi}_0, \ldots, \boldsymbol{\psi}_{m-1}]$ (principal components) or $\boldsymbol{\Psi}_{m|} = [\boldsymbol{\psi}_{n-m}, \ldots, \psi_{n-1}]$ (minor components), we refer to eigenvectors estimation. Alternatively, subspace estimation relates to $\boldsymbol{U}$ spanning the same subspace of the eigenvectors in either $\boldsymbol{\Psi}_{|m}$ or $\boldsymbol{\Psi}_{m|}$.

3) objective function: As for many iterative methods, some streaming algorithms for subspace estimation derive from iterative procedures solving a minimization or maximization problem. We provide a classification according to the objective function characterizing the optimization problem.

Identifying the k-dimensional principal (minor) subspace is equivalent to finding the column-orthonormal matrix $\boldsymbol{U} \in \mathbb{R}^{n\times k}$ that maximizes (minimizes) the variance of the projection $\boldsymbol{y} = \boldsymbol{U}^\top\boldsymbol{x}$ [23], [24]. Hence, some approaches consider the objective function

$$J_{\mathrm{Var}}(\boldsymbol{U}) = \mathbf{E}\left[\|\boldsymbol{U}^\top\boldsymbol{x}\|^2\right] = \mathrm{tr}\left(\boldsymbol{U}^\top\boldsymbol{\Sigma}\boldsymbol{U}\right) \tag{1}$$

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529

where $\|\cdot\|$ denotes the l_2 norm, $\mathrm{tr}(\cdot)$ computes the trace of its argument, and $\boldsymbol{U}$ is constrained to be column-orthonormal, i.e., $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}_k$ with $\boldsymbol{I}_k$ standing for the $k\times k$ identity matrix.

Maximizing (minimizing) projection variance is equivalent to minimizing (maximizing) the average norm of residual vector $\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{x}$. Hence, some methods [25], [26] consider the objective function

$$J_{\mathrm{MSE}}(\boldsymbol{U}) = \mathbf{E}\left[\|\boldsymbol{x} - \boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{x}\|^2\right] \tag{2}$$

As a further alternative, it is possible to define an objective function based on the so-called *spiked model* [27] whereby the observable $\boldsymbol{x}$ is assumed as an expansion of an m-dimensional signal $\boldsymbol{s}$ such that $\boldsymbol{s} = \boldsymbol{\Psi}_{|m}^\top\boldsymbol{x}$ [28], [29]. Despite their obvious link, some methods consider $\boldsymbol{x}$ and $\boldsymbol{s}$ separately and minimize

$$J_{\mathrm{SM}}(U) = \mathbf{E}\left[\|\boldsymbol{x} - \boldsymbol{U}\boldsymbol{s}\|^2\right] \tag{3}$$

to enforce the relationship among them. As a remark, the signal model behind (3) makes this objective function effective only in the case of principal subspace identification, i.e., $J_{\mathrm{SM}}(U) = 0$ implies $\mathrm{span}(U) = \mathrm{span}(\boldsymbol{\Psi}_{|m})$.

Worth noting that only one of the algorithms we consider (namely ISVD as described in Section III) does not solve an optimization problem to identify the subspace.

4) column-orthonormality: Given that the target $\boldsymbol{U}$ is column-orthonormal, it is possible to distinguish between two classes of algorithms depending on the properties of the current estimate $\boldsymbol{U}_t$: *(i)* every $\boldsymbol{U}_t$ is constrained to be column-orthonormal, *(ii)* $\boldsymbol{U}_t$ is only approximately column-orthonormal.

In general, algorithms ensuring column-orthonormality have complexity $\mathcal{O}(nk^2)$, while algorithms limiting to approximate column-orthonormality can achieve $\mathcal{O}(nk)$.

Those ensuring column-orthonormality can be further distinguished depending on the used technique. Some apply a specific orthonormalization procedure (typically a QR-decomposition) at each step. In this case, if the update produces a matrix $\boldsymbol{U}'_t$ that is not column-orthonormal, then the columns of the final $\boldsymbol{U}_t$ span the same subspace as the columns of $\boldsymbol{U}'_t$, while being orthonormal.

Some others projects onto the Grassmannian manifold [24], [30] that contains all possible $n \times k$ column-orthonormal matrices. Using projection implies that the span of the result of the non-orthonormal update $\boldsymbol{U}'_t$ is not necessarily the same as that of the final orthonormal $\boldsymbol{U}_t$, and some of the improvements in selecting $\boldsymbol{U}'_t$ might be lost.

As a third option, orthonormality can be guaranteed by constraining the updates of $\boldsymbol{U}_t$ along Grassmannian geodesics. In this case, the updated estimation improves over the previous one without leaving the acceptability region.

Table I classifies the methods, whose description is in the next Section, following the criteria described above. The table also reports the computational complexity and the field of interest in which the methods were initially conceived.

III. Methods description

This section describes the methods we then test and implement. For each method, we give the *update step*, i.e., the sequence of operations that at each time step t produces the matrix $\boldsymbol{U}_t$, representing the current estimate of the target subspace, starting from the current signal occurrence $\boldsymbol{x}_t$ and the previous estimate $\boldsymbol{U}_{t-1}$.

Most update steps compute intermediate quantities that have a proper role, namely $\boldsymbol{y}_t = \boldsymbol{U}_{t-1}^\top\boldsymbol{x}_t$ that is the vector of coefficients expressing the projection of $\boldsymbol{x}_t$ onto the subspaces as it is estimated at time $t-1$, and $\boldsymbol{r}_t = \boldsymbol{x}_t - \boldsymbol{U}_{t-1}\boldsymbol{y}_t$ that is the residual of such a projection. Note that if $\boldsymbol{U}_{t-1}$ is orthonormal, the two vectors $\boldsymbol{y}_t$ and $\boldsymbol{r}_t$ are orthogonal. For the sake of brevity, the computation of $\boldsymbol{y}_t$ and $\boldsymbol{r}_t$ are not explicitly mentioned in the descriptions of the methods.

A. Oja

Originally proposed in [31] for the principal subspace estimation with $k = m = 1$, and then extended to the rank-k cases in [32], it starts from a random column-orthonormal $\boldsymbol{U}_0 \in \mathbb{R}^{n\times k}$. At the t-th sample, the estimate $\boldsymbol{U} = \boldsymbol{\Psi}_{|k}$ is updated according to the input data $\boldsymbol{x}_t$ as

$$\boldsymbol{U}_t = \Omega\left(\boldsymbol{U}_{t-1} + \gamma_t\boldsymbol{x}_t\boldsymbol{x}_t^\top\boldsymbol{U}_{t-1}\right) \tag{4}$$

where $\Omega(\cdot)$ is an operator that orthonormalizes the columns of its argument, e.g., gives the $\boldsymbol{Q}$ matrix in the QR decomposition of its argument [33, Chapter 2]. The parameter γ_t is the step size or learning rate that may change with time.

Notably, authors of [25], [34], [35] show that Oja is an extension of the well-known power method [36] that, in turn, is equivalent to solving a maximization problem where the objective function is (1) and $\boldsymbol{U}_t$ is constrained to be column-orthonormal. In particular, the gradient of (1) with respect to $\boldsymbol{U}$ is $\boldsymbol{\Sigma}\boldsymbol{U}$ such that (4) is equivalent to the update of a stochastic gradient descent algorithm where $\boldsymbol{\Sigma}$ is approximated by $\boldsymbol{x}_t\boldsymbol{x}_t^\top$, γ_t is the learning rate, and $\Omega(\cdot)$ forces the update to yield a column-orthonormal matrix.

To save some computation, one may think of applying Ω only after a certain number of updates [25]. Nevertheless, proper sizing of the number of steps without orthonormalization depends on the application.

Furthermore, when minimization of (1) is considered instead of maximization, one may aim to identify the minor subspace $\boldsymbol{U} = \boldsymbol{\Psi}_{k|}$ with $k = n-m$. With respect to (4), here stochastic gradient descent algorithm performing minimization follows the opposite of the gradient of (1), i.e.,

$$\boldsymbol{U}_t = \Omega\left(\boldsymbol{U}_{t-1} - \gamma_t\boldsymbol{x}_t\boldsymbol{x}_t^\top\boldsymbol{U}_{t-1}\right) \tag{5}$$

Lastly, since the convergence of the method depends on the choice of the initial matrix $\boldsymbol{U}_0$, [22] proposes a procedure of warm start that avoids random initialization.

B. Krasulina

Originally proposed in [37], [38] and recently revised in [23] to include the $k > 1$ case, it also starts from a random column-orthonormal $\boldsymbol{U}_0 \in \mathbb{R}^{n\times k}$. The update step is

$$\boldsymbol{U}_t = \Omega\left(\boldsymbol{U}_{t-1} + \gamma_t\boldsymbol{r}_t\boldsymbol{y}_t^\top\right) \tag{6}$$

According to [23], [26], the update in (6) converges to a matrix that approximates the minimizer of (2).

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529

TABLE I
CLASSIFICATION OF METHODS FOR SUBSPACE ANALYSIS CONSIDERED IN THIS PAPER.

Method		Principal/minor subspace	Eigenvectors or subspace	Objective function	Column orthonormalization	Complexity	Original field
Oja	[31] [32] [25] [34] [35] [22]	both	eigenvectors	(2)	QR	nk^2	neural networks
Krasulina	[37] [38] [23]	both	eigenvectors	(1)	QR	nk^2	
HFRANS	[39] [40] [41] [42]	minor	eigenvectors	(1)	approximate	nk	
PAST	[26] [43] [44] [45] [46]	principal	subspace	(3)	approximate	nk	adaptive filtering
ISVD	[47] [48]	principal	eigenvectors	N/A	QR	nk^2	
GROUSE	[28] [49] [50]	principal	subspace	(3)	Grassmannian	nk	matrix completion

As for Oja, a warm start approach has been proposed in [23] and the minor subspace can be targeted by simply changing the sign of the last equation in (6) to yield

$$\boldsymbol{U}_t = \Omega\left(\boldsymbol{U}_{t-1} - \gamma_t \boldsymbol{r}_t \boldsymbol{y}_t^\top\right) \tag{7}$$

In this case, the method converges to $\boldsymbol{U} = \boldsymbol{\Psi}_{k|}$ with $k = n - m$.

Oja and Krasulina are deeply linked. In fact, [51] proves that the

$$\Omega\left(\boldsymbol{U}_{t-1} + \gamma_t \boldsymbol{x}_t \boldsymbol{x}_t^\top \boldsymbol{U}_{t-1}\right) = \boldsymbol{U}_{t-1} + \gamma_t \boldsymbol{r}_t \boldsymbol{y}_t^\top + o(\gamma_t^2) \tag{8}$$

thus ultimately establishing equivalent convergence properties for (4) and (6) as $\gamma_t \to 0$ for growing t.

C. HFRANS

The analogy of the two previous methods highlighted by (8) has led to a class of methods that try to provide column-orthonormality avoiding the Ω operator.

Since $\boldsymbol{U}_t = \boldsymbol{U}_{t-1} + \gamma_t \boldsymbol{r}_t \boldsymbol{y}_t^\top$ tends to be column-orthonormal for $t \to \infty$, an extreme option is to consider $o(\gamma_t^2)$ negligible and simply avoid orhonormalization. Such a choice can be acceptable when targeting $\boldsymbol{U} = \boldsymbol{\Psi}_{|k}$. Nevertheless, when aiming at $\boldsymbol{U} = \boldsymbol{\Psi}_{k|}$, the minimal amplitude of the projection of the signal on the minor subspace makes the whole procedure extraordinarily error-prone and may spoil convergence.

To overcome this impasse, [41] first proposed a term to approximate the $o(\gamma_t^2)$ residue in (8) giving rise to the so-called OOja (Orthogonal Oja) method that is then extended [42] with a policy that adapts γ_t to both $\boldsymbol{U}_t$ and $\boldsymbol{x}_t$, leading to the so-called NOOja (Normalized Orthogonal Oja) method.

OOja and NOOja further evolved [39] into HFRANS (Householder Fast Rayleigh's quotient-based Adaptive Noise Subspace). This method is also an adjustment of FRANS, a previous Rayleigh quotient-based adaptive noise subspace method [40]. In HFRANS, Househölder transformations are introduced to grant the numerical stability needed to cope with the minor subspace case.

The method starts from a random $\boldsymbol{U}_0 \in \mathbb{R}^{n\times k}$ matrix and uses the following update rule that depends on a given $0 < \gamma < 2$,

$$\begin{aligned}
\tau_t &= \frac{1}{\|\boldsymbol{y}_t\|^2}\left[\left(1-(2-\gamma)\gamma\frac{\|\boldsymbol{y}_t\|^2}{\|\boldsymbol{x}_t\|^2}\right)^{-\frac{1}{2}} - 1\right] \\
\hat{\boldsymbol{u}}_t &= \left(1+\tau_t\|\boldsymbol{y}_t\|^2\right)\boldsymbol{x}_t - \frac{\tau_t\|\boldsymbol{x}_t\|^2}{\gamma}\boldsymbol{U}_{t-1}\boldsymbol{y}_t \\
\boldsymbol{u}_t &= \frac{\hat{\boldsymbol{u}}_t}{\|\hat{\boldsymbol{u}}_t\|} \\
\boldsymbol{U}_t &= \boldsymbol{U}_{t-1} - 2\boldsymbol{u}_t\boldsymbol{u}_t^\top\boldsymbol{U}_{t-1}
\end{aligned} \tag{9}$$

D. PAST

PAST (Project Approximation Subspace Tracking) [26], [43] is an algorithm obtained by minimizing (3) in which the expectation is unrolled in time as an exponentially weighted sum, i.e., by setting

$$\boldsymbol{U}_t = \underset{\boldsymbol{U}\in\mathbb{R}^{n\times k}}{\arg\min} \sum_{l=1}^{t} \beta^{t-l}\|\boldsymbol{x}_l - \boldsymbol{U}\boldsymbol{s}_l\|^2 \tag{10}$$

without the constraint of $\boldsymbol{U}_t$ being column-orthonormal and where $\beta \in [0,1]$ is the forgetting factor that weights the prior samples. Equation (10) is based on the fact that signal observances $\boldsymbol{x}_l$ are generated accordingly to a spiked model, i.e., $\boldsymbol{s}_l = \Psi_{|k}^\top \boldsymbol{x}$.

Since, at each signal occurrence, only $\boldsymbol{x}_l$ is known, $\boldsymbol{s}_l$ is approximated by the projection vector $\boldsymbol{y}_l = \boldsymbol{U}_{l-1}^\top \boldsymbol{x}_l$, i.e., by adopting the last estimated $\boldsymbol{U}$. Thanks to this approximation, the problem has a closed solution, and $\boldsymbol{U}_t$ can be retrieved by mean of recursive least squares (RLS) methods, which allow for a computational cost as low as $\mathcal{O}(nk)$.

Iterations start from a random $\boldsymbol{U}_0 \in \mathbb{R}^{n\times k}$ and $\boldsymbol{P}_0 = \delta \boldsymbol{I}_k$ for some $\delta > 0$ and the update step is

$$\begin{aligned}
\boldsymbol{h}_t &= \boldsymbol{P}_{t-1}\boldsymbol{y}_t \\
\boldsymbol{g}_t &= \boldsymbol{h}_t / \left(\beta + \boldsymbol{y}_t^\top \boldsymbol{h}_t\right) \\
\boldsymbol{P}_t &= \beta^{-1}\left(\boldsymbol{P}_{t-1} - \boldsymbol{g}_t\boldsymbol{h}_t^\top\right) \\
\boldsymbol{U}_t &= \boldsymbol{U}_{t-1} - \boldsymbol{r}_t\boldsymbol{g}_t^\top
\end{aligned} \tag{11}$$

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529

Though $\boldsymbol{U}_t$ is not guaranteed to be column-orthonormal for any finite t, [26], [46] show that column-orthonormality is achieved asymptotically as t increases. In applications where columns-orthonormality is important at each step, PAST variants can be adopted. For instance, PASTd in [26] is a version based on the deflation technique that comes at the cost of an increase in complexity. Otherwise, in [44] nearly-orthonormality is provided by correction terms applied to each update of $\boldsymbol{U}$ that keeps complexity at $\mathcal{O}(nk)$.

PAST is the base for other approaches such as [45] that deals with the case in which data is perturbed by colored noise, [29] that is designed to cope with the missing components in the vectors $\boldsymbol{x}_t$, and [52], [53] where the Approximated Power Iteration (API) extends the standard power method by exploiting the same approximation used in PAST.

E. ISVD

Given any $n \times t$ matrix $\boldsymbol{A}$, Singular Value Decomposition (SVD) [36, Chapter 2] finds three factors $\boldsymbol{P}$, $\boldsymbol{D}$ and $\boldsymbol{Q}$ such that $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{Q}^\top$ with $\boldsymbol{P}$ and $\boldsymbol{Q}$ being square orthonormal matrices of size $n \times n$ and $t \times t$ respectively, and $\boldsymbol{D}$ is a diagonal $n \times t$ matrix whose diagonal entries are called *singular values*. SVD is symbolized as $\boldsymbol{A} \overset{\text{SVD}}{\rightarrow} \boldsymbol{P}, \boldsymbol{D}, \boldsymbol{Q}$.

Let also $\boldsymbol{X}_t = \begin{bmatrix} \boldsymbol{x}_1 & \dots & \boldsymbol{x}_t \end{bmatrix}$ be the matrix containing samples up to the t-th. Subspace analysis has to do with the SVD of $\boldsymbol{X}_t$ with a very large t. In fact, if $\boldsymbol{X}_t \overset{\text{SVD}}{\rightarrow} \bar{\boldsymbol{U}}_t, \bar{\boldsymbol{S}}_t, \bar{\boldsymbol{V}}_t$, then the singular values $\sigma_j(t)$ in $\bar{\boldsymbol{S}}_t$ are such that $\sigma_j^2(t) \rightarrow \lambda_j$ and $\bar{\boldsymbol{U}}_t \rightarrow \Psi$ as $t \rightarrow \infty$.

Authors in [54] show that the SVD of $\boldsymbol{X}_t$ can be effectively computed from the SVD of $\boldsymbol{X}_{t-1}$. Moreover, it is possible to only focus on the principal subspace with the so-called *thin* SVD (tSVD) [47], which is a decomposition that only computes the first k singular values and the corrspending columns of $\bar{\boldsymbol{U}}_t$ and $\bar{\boldsymbol{V}}_t$.

The ISVD (Incremental SVD) relies on the fact that, given the decomposition $\boldsymbol{X}_{t-1} \overset{\text{tSVD}}{\rightarrow} \boldsymbol{U}_{t-1}, \boldsymbol{S}_{t-1}, \boldsymbol{V}_{t-1}$, one may express the data matrix $\boldsymbol{X}_t$ as

$$\boldsymbol{X}_t = \begin{bmatrix} \boldsymbol{U}_{t-1} & \frac{\boldsymbol{r}_t}{\|\boldsymbol{r}_t\|} \end{bmatrix} \begin{bmatrix} \boldsymbol{S}_{t-1} & \boldsymbol{y}_t \\ \boldsymbol{0}_k^\top & \|\boldsymbol{r}_t\| \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_{t-1}^\top & \boldsymbol{0}_k \\ \boldsymbol{0}_k^\top & 1 \end{bmatrix} \tag{12}$$

The equality in (12) holds exactly only if the rank of $\boldsymbol{X}_{t-1}$ is k, otherwise it is an approximation. The update rule of ISVD starts with the computation of the SVD of an adjusted version of the inner matrix in (12)

$$\begin{bmatrix} \beta\boldsymbol{S}_{t-1} & \boldsymbol{y}_t \\ \boldsymbol{0}_k^\top & \|\boldsymbol{r}_t\| \end{bmatrix} \tag{13}$$

where the parameter $0 < \beta < 1$ is added to control the memory of the algorithm. The SVD yields two orthonormal factors $\boldsymbol{P}$ and $\boldsymbol{Q}$ and a diagonal factor $\boldsymbol{D}$ whose product can be plugged into (12) to obtain an internal diagonal factor and thus yield the updated tSVD of $\boldsymbol{X}_t$. Such a representation is then shrunk to the minimum by keeping only the first k columns of the left and right factor and only the first k columns and rows of the central factor.

Overall, starting from a random orthonormal $\boldsymbol{U}_0 \in \mathbb{R}^{n \times k}$, the update step computes

$$\begin{aligned} &\begin{bmatrix} \beta\boldsymbol{S}_{t-1} & \boldsymbol{y}_t \\ \boldsymbol{0}_k^\top & \|\boldsymbol{r}_t\| \end{bmatrix} \overset{\text{SVD}}{\rightarrow} \boldsymbol{P}, \boldsymbol{D}, \boldsymbol{Q} \\ &\boldsymbol{S}_t = (\boldsymbol{D})_{\ulcorner k} \\ &\boldsymbol{U}_t = \left(\begin{bmatrix} \boldsymbol{U}_{t-1} & \frac{\boldsymbol{r}_t}{\|\boldsymbol{r}_t\|} \end{bmatrix} \boldsymbol{P} \right)_{|k} \end{aligned} \tag{14}$$

where $(\cdot)_{|k}$ is the same operator used before which selects the first k columns of its argument, while $(\cdot)_{\ulcorner k}$ selects the first k columns and the first k rows of its argument.

Besides, the speed of convergence is highly affected by the condition number of $\boldsymbol{X}_t$. To partially overcome this problem authors in [48] propose the Polar Incremental Matrix Completion (PIMC) algorithm which adapts the memory factor to the norm of the observed samples $\beta = \frac{a_t}{\|\boldsymbol{S}_t\|_{\text{F}}}$ where $a_t^2 = a_{t-1}^2 + \|\boldsymbol{x}_t\|_2^2$, $a_0 = 1$ and $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm of a matrix.

F. GROUSE

GROUSE (Grassmaniann Rank-One Update Subspace Estimation) is a streaming algorithm for subspace tracking proposed in [28]. Although it is designed to deal with the case in which some components of $\boldsymbol{x}_t$ are unknown, we here consider the version for complete data. The idea consists in applying the stochastic gradient descent to minimize (3) while making moves within the set of all possible column-orthonormal matrices, i.e., the Grassmaniann manifold of the k-dimensional subspaces of $\mathbb{R}^n$.

Starting from one of such matrices, the update rule is

$$\begin{aligned} &\boldsymbol{p}_t = \boldsymbol{U}_{t-1}\boldsymbol{y}_t \\ &\theta_t = \arctan\left[(1-\alpha_t) \frac{\|\boldsymbol{r}_t\|}{\|\boldsymbol{p}_t\|} \right] \\ &\boldsymbol{z}_t = \cos(\theta_t) \frac{\boldsymbol{p}_t}{\|\boldsymbol{p}_t\|} + \sin(\theta_t) \frac{\boldsymbol{r}_t}{\|\boldsymbol{r}_t\|} \\ &\boldsymbol{U}_t = \boldsymbol{U}_{t-1} + \left(\frac{\boldsymbol{z}_t}{\|\boldsymbol{z}_t\|} - \frac{\boldsymbol{p}_t}{\|\boldsymbol{p}_t\|} \right) \frac{\boldsymbol{y}_t^\top}{\|\boldsymbol{y}_t\|} \end{aligned} \tag{15}$$

where the expression for θ_t is derived from [49, eq. (3)–(4)] and where α_t is meant to mitigate the effect of noise.

Convergence of GROUSE is analyzed in [49], [50], and [55] shows that GROUSE and ISVD are strictly linked. In particular, the application of the ISVD to the missing data case is equivalent to GROUSE for a specific choice of its parameters.

G. Discussion

Before proceeding with the performance analysis of these methods, we can evidence their main characteristics and similarities as reported in Table I and Fig. 2. The table shows that half methods have computational complexity proportional to nk^2, the ones requiring column-orthonormalization, and half to nk. Among this latter class of methods, HFRANS is the only one able to produce as output an estimation of the eigenvectors. In addition, it is possible to identify two families,

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529

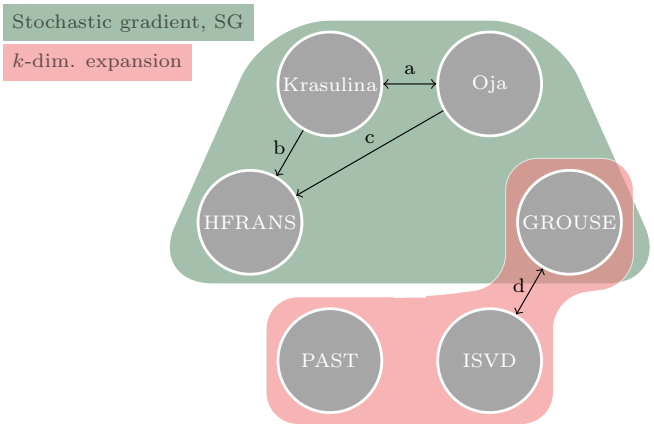


Fig. 2. Classification of the methods and their relationships. It is possible to define two main families, methods based on the stochastic gradient and methods properly designed for signals with intrinsic dimensionality equal to k (spiked model). The connections between methods are also reported according to their description: *a* - [51] discusses the relationships between both methods showing that (8) it is possible to approximate the Oja's update as the one proposed by Krasulina without the orthonormalization; *b & c* - [40]–[42] present methods derived from (8) that has been further extended in [39] to HFRANS; *d* - [55] shows that the application of the ISVD to the missing data case is equivalent to a suitable configuration of GROUSE.

one including methods based on the stochastic gradient and another for methods properly designed for signals compliant with the k-dimensional spiked model. The only method belonging to both groups is GROUSE. A visual representation of these relationships is depicted in Fig. 2, which also shows the connections between the methods.

As for any iterative algorithm, the problem of the streaming estimation of a subspace must cope with all the aspects related to the necessary guarantees for convergence [18], [21] and this is an aspect that is well investigated in all presented methods. Nevertheless, convergence guarantees strictly depends on the signal model and addressed application. Typical assumptions are: *(i)* signal stationarity or variation of the statistical properties slower than the convergence of the method; *(ii)* signal as expansion of a k-dimensional information source (spiked model); *(iii)* the *eigengap*, i.e., the difference between the k-th and $(k+1)$-th eigenvalues of $\boldsymbol{\Sigma}$, strictly greater than zero. In addition, methods convergence could also depend on the occurrence of corrupted or missing data for which a preprocessing stage may be included in the acquisition system.

IV. Methods performance

In this section, we test the functional performance of the methods described in Section III by employing each method to identify the subspace characterizing a set of signal observations.

Data samples are generated according to the spiked model [27] as follows

$$\boldsymbol{x}_t = \boldsymbol{\Phi}\boldsymbol{s}_t + \boldsymbol{\nu}_t \tag{16}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{n\times m}$ is a given expansion matrix with $m < n$, i.e., it is a column-orthonormal matrix that expands instances $\boldsymbol{s}_t \in \mathbb{R}^m$ of a zero-mean random Gaussian source with covariance $\boldsymbol{I}_m$, while $\boldsymbol{\nu}_t \in \mathbb{R}^n$ represents realizations of a zero-mean Gaussian noise term with covariance $\rho\boldsymbol{I}_n$ such that $0 < \rho < 1$ controls the noise level.

With this model we have $\boldsymbol{\Sigma} = \mathbf{E}\left[\boldsymbol{x}_t\boldsymbol{x}_t^\top\right] = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \rho\boldsymbol{I}_n$, and since $\boldsymbol{\Phi}$ is column-orthonormal and $\rho < 1$, then $\boldsymbol{\Phi}$ itself spans the m-dimensional principal subspace, while its orthogonal complement $\boldsymbol{\Phi}^\perp$ spans the $(n-m)$-dimensional minor subspace. As a result, the target $n \times k$ matrix $\boldsymbol{U}$ is $\boldsymbol{\Phi}$ in the case of principal subspace estimation, and it is $\boldsymbol{\Phi}^\perp$ for the minor subspace.

To quantify the effectiveness of subspace analysis, for each streaming method we monitor the sequence of reconstruction errors

$$e_t = \|\boldsymbol{U} - \boldsymbol{U}_t\boldsymbol{U}_t^\top\boldsymbol{U}\|_F \tag{17}$$

where $\|\cdot\|_F$ indicates the Frobenius norm of its argument. In the case of correct estimation, we have $\boldsymbol{U}_t = \boldsymbol{U}$ and thus $e_t = 0$ while the error is maximum when $\boldsymbol{U}_t$ is orthogonal to $\boldsymbol{U}$, yielding $e_t = k$.

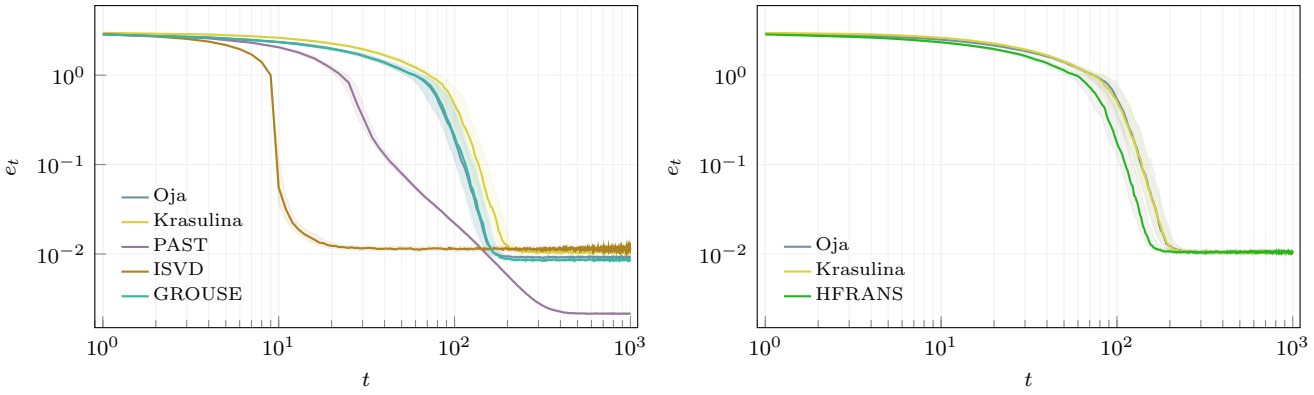
Depending on the applications, the dimension of the signal, i.e., n, may range from a few units to hundreds or even thousands. According to the model described in Section II, x may contain one reading from several sensors, a window of a single sampled signal, or a mix of these two cases in which windows from several sensors are considered. Considering this large variety of possible scenarios, simulations focus on $n = 100$, $m = 10$ and noise amplitude $\rho = 10^{-3}$. For each task, we run 100 Montecarlo trials in which we independently draw two random column-orthonormal matrices $\boldsymbol{\Phi}$ and $\boldsymbol{U}_0$, and independently generate $1\,000$ sample windows $\boldsymbol{x}_t$ following (16).

The parameters controlling each method are set as reported in Table II: Oja and Krasulina need a learning rate γ_t, PAST and ISVD need to set a forgetting factor β, and finally, GROUSE and HFRANS depend on α (controlling the effect of noise) and γ (that can be seen as a re-scaled learning rate). Parameters are tuned to optimize the capability to identify the target subspace. In detail, we consider learning rates modelled as $\gamma_t = c/t^d$ with $d \in [0,1]$ and $c > 0$ selected to increase convergence speed. In the tested framework, $d \in \{0, 1/2, 1\}$ provides similar performances, thus we consider a constant learning rate $\gamma_t = \gamma = c$, i.e., $d = 0$. For HFRANS, the tuning of γ is independent of t and, for GROUSE, α is set to cope with the noise level. Finally, the forgetting factors β of ISVD and PAST are selected as the largest possible values that guarantee convergence.

Results in terms of e_t are shown in Fig. 3 with Fig. 3a for PSA and Fig. 3b for MSA. Solid lines represent median values of the Montecarlo trials while shaded areas indicate the spread containing 50% of the values. The figure shows that all methods can deal with the subspace identification task in both PSA and MSA. In general, the variability of the performances represented by the shaded areas is almost negligible with the only exception of a limited increase in correspondence of the transaction phase, i.e., when a curve passes from high to very low error values. For PSA, ISVD and PAST are the fastest to converge while in MSA the speed of convergence is similar among the methods with HFRANS slightly faster.

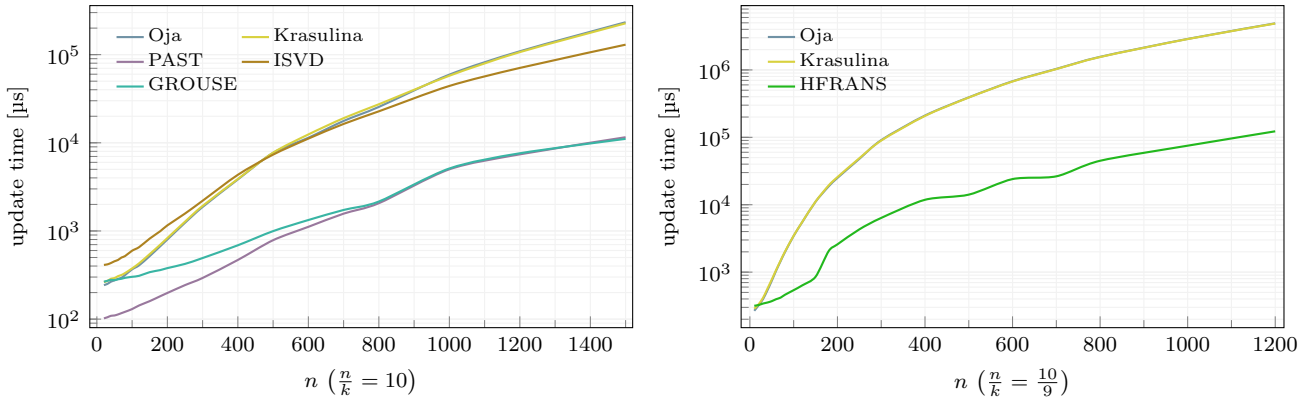
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529



(a) (b)

Fig. 3. Performance comparison for subspace identification: (a) principal subspace (PSA); (b) minor subspace (MSA).



(a) (b)

Fig. 4. Average time for each considered method to run the update step on the high-end device for different input dimension n and fixed ratio n/k. (a) methods PSA with $n/k = 10$; (b) methods for MSA with $n/k = 10/9$;

TABLE II
ALGORITHM PARAMETERS, TUNED VALUE WITH CORRESPONDING TUNING RANGES, AND UPDATE TIMES FOR THE RPI IMPLEMENTATION.

method	parameter	value	range	update time [µs] ave. ± std
Principal Subspace identification				
Oja	γ	1	[0.001, 100]	354 ± 10
Krasulina	γ	0.1	[0.001, 100]	378 ± 11
PAST	β	0.99	[0.7, 0.999]	116 ± 5
ISVD	β	0.9	[0.7, 0.999]	585 ± 13
GROUSE	α	0.2	[0, 0.5]	304 ± 10
Minor Subspace identification				
Oja	γ	0.1	[0.001, 100]	3446 ± 60
Krasulina	γ	0.1	[0.001, 100]	3472 ± 27
HFRANS	γ	1	[0.001, 100]	540 ± 29

V. IMPLEMENTATION ON COMMERCIAL SENSOR/EDGE DEVICES

Since sensor and edge devices are employed for various applications and tasks, different solutions are available, ranging from a single-board system equipped with a microcontroller to more complex systems composed of different modules. To cope with both scenarios, two subsections report details about two reference platforms.

A. High-end devices

Here, we focus on the Raspberry Pi 4 model B, which could be considered a reference for the high-end devices family. In details, we refer to the board equipped with 1GB of RAM and Raspberry Pi OS (32-bit) Lite 5.4.51-v7l+ as an operative system. Clock frequency ranges from 600 MHz to 800 MHz[1]. Methods are implemented in Python (version 3.7.3), employing the packages Scipy (version 1.1.0) and Numpy (version 1.16.2) for the required algebraic manipulations.

We set $n = 100$ and $m = 10$ (as in Section IV) reminding that the target matrix $\boldsymbol{U}$ has dimensions $n \times m$ for PSA and $n \times (n-m)$ for MSA. For each method, update time is reported in Table II as mean and standard deviation over 1 000 updates. As expected, updates for MSA methods take longer than PSA.

[1]The firmware dynamically manages clock frequency depending on the CPU temperature.

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529

For PSA, the fastest method is PAST, while the most time-consuming is ISVD which computes an SVD of an $(k+1) \times (k+1)$ matrix at each update. In the case of MSA, HFRANS is the preferred method because it does not require complex operations like orthonormalization or SVD, similar to PAST.

As anticipated, x may contain samples from different sensors and we cope with this scenario by enlarging the values for n and k, keeping constant their ratio. Fig. 4 shows how the average update time increases with n in case of $n/k = 100/10$ for PSA and $n/k = 100/90$ for MSA. In both cases, the gap between the methods imposing the estimate $\boldsymbol{U}_t$ to be column-orthonormal (Oja, Krasulina, and ISVD) and the other ones (HFRANS, GROUSE, and PAST) increases with n. This trend confirms the difference in the complexity of the methods.

B. Low-end devices

This subsection discusses the implementation of these methods on devices that are more suitable for use as sensor nodes. We developed a C library[2] targeting the ARM Cortex microcontroller (MCU) family of devices. The library backend is mainly based on the well-known ARM CMSIS-DSP library[3].

We implement the methods by adopting specific techniques, described in Appendix A, to minimize memory footprint and processing time. Both GROUSE and PAST implementations include these techniques, while Oja, Krasulina, and ISVD require additional consideration due to the use of QR decomposition or SVD, as discussed in the following section. Then, performances in terms of update time, energy consumption and memory requirements on the MCU are analyzed, along with an assessment of the error in estimating the target matrix $\boldsymbol{U}$ for both PSA and MSA.

1) QR decomposition: Oja and Krasulina methods employ QR decomposition to implement the $\Omega(\cdot)$ operator in (4) and (6) that makes the current estimate $\boldsymbol{U}_t$ column-orthonormal. Here, we consider three common algorithms to implement the QR decomposition: Householder (Hous-QR) [56], Modified Gram-Shmidt (MGS-QR) [57], and Cholesky (Chol-QR) [58]. They all are iterative algorithms that differ from addressing the trade-off between the orthogonality of the output and computational complexity. Hous-QR offers excellent numerical stability, ensuring that the output is column-orthonormal even when the matrix is poorly conditioned. However, it is the most computationally expensive option. Chol-QR, on the other hand, prioritizes computational efficiency over stability. MGS-QR is a compromise between the two, offering a middle ground in terms of both stability and computational complexity. Since Oja and Krasulina methods results to be robust to estimates $\boldsymbol{U}_t$ being only approximately orthonormal, we select Chol-QR for implementing the $\Omega(\cdot)$ operator on MCU as reported in Algorithm 1. The implementation of Cholesky decomposition is based on the ARM CMSIS-DSP library. In addition, we implement the inversion of the upper triangular matrix $\boldsymbol{R}$ with the backward substitution technique [59]. Since $\boldsymbol{R}^{-1}$ is a lower triangular matrix, we store $\boldsymbol{R}$ and $\boldsymbol{R}^{-1}$ as the two triangular parts of a single square matrix.

Algorithm 1 Cholesky-based QR decomposition

1: $\text{cholesky}(\boldsymbol{A}^\top \boldsymbol{A}) \rightarrow \boldsymbol{L}\boldsymbol{L}^\top$ where $\boldsymbol{L}$ is lower triangular
2: $\boldsymbol{R} \leftarrow \boldsymbol{L}^\top$
3: $\boldsymbol{R}^{-1} \leftarrow \text{BS}(\boldsymbol{R})$ (BS: backward substitution)
4: $\boldsymbol{Q} \leftarrow \boldsymbol{A}\boldsymbol{R}^{-1}$

TABLE III
PERFORMANCE ON STM32H743ZIT (REV. V) @1.8 V, 480 MHz, CACHE ON ($n = 100, m = 10$)

Method	Clock cycles per update	Time per update	Energy per update (no peripherals)	Energy per update (all peripherals)
Principal subspace				
Oja	109.97 k	229.10 µs	45.36 µJ	90.72 µJ
Krasulina	114.86 k	239.28 µs	47.38 µJ	94.76 µJ
PAST	20.18 k	42.04 µs	8.32 µJ	16.65 µJ
ISVD	136.02 k	283.37 µs	56.11 µJ	112.21 µJ
GROUSE	22.96 k	47.83 µs	9.47 µJ	18.94 µJ
Minor subspace				
Oja	8711.27 k	18 148.47 µs	3593.40 µJ	7186.80 µJ
Krasulina	8748.48 k	18 226.01 µs	3608.75 µJ	7217.50 µJ
HFRANS	235.30 k	490.20 µs	97.06 µJ	194.12 µJ

2) SVD operation: SVD is used in the ISVD algorithm in (14). The Golub-Reinsch technique (GR-SVD) [60] is adopted for this purpose, which has computational complexity similar to the Chol-QR method. GR-SVD is implemented on the MCU using the CControl library[4]. To optimize the use of memory on the MCU, the CControl library is modified to eliminate the unnecessary $\boldsymbol{Q}$ matrix in (14) and to overwrite the input matrix with the $\boldsymbol{P}$ matrix.

All methods are tested on a STM32H743ZIT (rev. V), an MCU based on ARM Cortex M7 family with a 32-bit floating-point unit, $f_{\text{CLK}} = 480\,\text{MHz}$, and both instructions cache and data cache enabled[5]. With this setup, the energy consumption of a single update has been estimated as $E_{\text{update}} = V_{\text{DD}} \times I_{\text{DD}} \times t_{\text{update}}$, where V_{DD} is the supply voltage, I_{DD} is the absorbed current and t_{update} is number of clock cycles necessary for a single update divided by the clock frequency. Values are obtained from datasheet. In particular, we refer to current values corresponding to $V_{\text{DD}} = 1.8\,\text{V}$ and with either no peripherals or all the peripherals enabled. Table III reports time and energy for a single update of each method. Fig. 5 instead shows how the update time scales with n with a fixed n/k ratio.

For what concerns memory footprint, the contribution of each method is split into three parts: *(i)* stack memory, which is a fixed cost negligible compared to the other contributions and independent from either the adopted method or the values of n and m; *(ii)* input vector $\boldsymbol{x}$ and matrix $\boldsymbol{U}_t$, which are the same for any method; *(iii)* buffers of various sizes necessary for computation. This last contribution is what characterizes

[2]online repository https://github.com/SSIGPRO/streaming_pca
[3]online repository https://github.com/ARM-software/CMSIS_5
[4]online repository https://github.com/DanielMartensson/CControl
[5]The code is compiled with fast target gcc option (-Ofast) in order to maximize the speed performance.

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529

TABLE IV
MEMORY REQUIREMENTS FOR EACH METHOD (MEMORY EXAMPLE WITH $n = 100$, $m = 10$, 32 bit SCALARS)

Method	Extra buffers	Overall memory	Memory example
	Principal subspace		
Oja	$k \times k$	$n(k+1)+k^2$	4.80 kB
Krasulina	$k \times k$	$n(k+1)+k^2$	4.80 kB
PAST	$k \times k$, n, k	$n(k+2)+k(k+1)$	5.24 kB
ISVD	$k{+}1 \times k{+}1$, $k{+}1$, $k{+}1$	$n(k{+}1){+}k(k{+}4){+}3$	4.97 kB
GROUSE	n, k	$n(k+2)+k$	4.84 kB
	Minor subspace		
Oja	$k \times k$	$n(k+1)+k^2$	17.20 kB
Krasulina	$k \times k$	$n(k+1)+k^2$	17.20 kB
HFRANS	n, k	$n(k+2)+k$	9.29 kB

the memory footprint of each method. Table IV[6] reports the size of extra buffers, the size of the overall memory, and the actual memory requirement for $n = 100$ and $m = 10$.

Finally, Fig. 6 shows the performance in terms of the estimation error of each method for subspace identification implemented on MCU with the same setup reported in Section IV. These plots confirm the trends observed in Fig. 3.

VI. LESSON LEARNT AND CONCLUSION

In this work, we select and compare six methods for streaming subspace identification tackling either principal or minor subspace analysis. These methods are described and compared within a consistent framework, using an input stream generated from a stationary spiked model. The implementation of these methods is also demonstrated on two different hardware platforms, designed for use with sensor and edge devices, to evaluate their resource needs and compatibility with current data collection systems.

Here are some conclusions about the usability of the different methods that can be derived from our analysis:

1) Oja's and Krasulina's are the only methods addressing the identification of both principal and minor subspace. These methods were the first proposed and have received the most study. Since these methods do not assume that the signal is an expansion of a lower dimensional source of information, they have the advantage of being more generally applicable. However, they perform poorly in terms of both speed of convergence and computational costs;
2) ISVD results to be the fastest to converge for PSA. However, since each iteration requires an SVD, its execution could be even slower than Oja's and Krasulina's;
3) GROUSE takes longer to converge compared to other methods, but it has a very low computational cost, especially for high dimensional signals, due to the lack of a column-orthonormalization step;
4) PAST features the lowest estimation error and it is the most lightweight method. As a drawback, the tuning of its main parameter may be critical;
5) HFRANS is the most effective method for MSA due to its rapid convergence and low computational cost. However, it should be noted that HFRANS cannot be utilized for PSA.

In terms of memory footprint, all methods have a similar cost for storage, with a complexity of $nk + o(k^2) \simeq nk$. The primary difference between the methods lies in the additional buffers required, which have sizes ranging from n to k^2. When the target subspace dimension is significantly larger than the signal dimension, GROUSE and HFRANS may be preferred due to their smaller buffer sizes. It is also worth noting that on low-end devices, memory constraints may affect the execution time of these methods. If the estimate $\boldsymbol{U}_t$ does not fit within the memory cache, the time required for each iteration may be significantly impacted by memory access. This is particularly relevant for methods utilizing column-orthonormalization.

As a final remark, we prove that all selected streaming algorithms are suitable for deployment on a sensor/edge device and that the selection of the method for a particular application must consider the characteristics and features of each method.

APPENDIX A
FAST AND MEMORY EFFICIENT ARITHMETIC ON MCU

The implementation of the streaming methods on the selected MCU is based on the ARM CMSIS-DSP library, which stores any $n \times k$ matrix as a linear vector of size nk to maximize memory contiguity and minimize the number of memory accesses. Hereafter we overview the techniques employed to improve the efficacy of vector-matrix multiplications.

1) Loop unroll: Since loops are massively employed, loop-unrolling significantly increases the performance by reducing the amount of data transferred and the number of loop index updates. This technique is automatically adopted at compile time by using -Ofast gcc option.

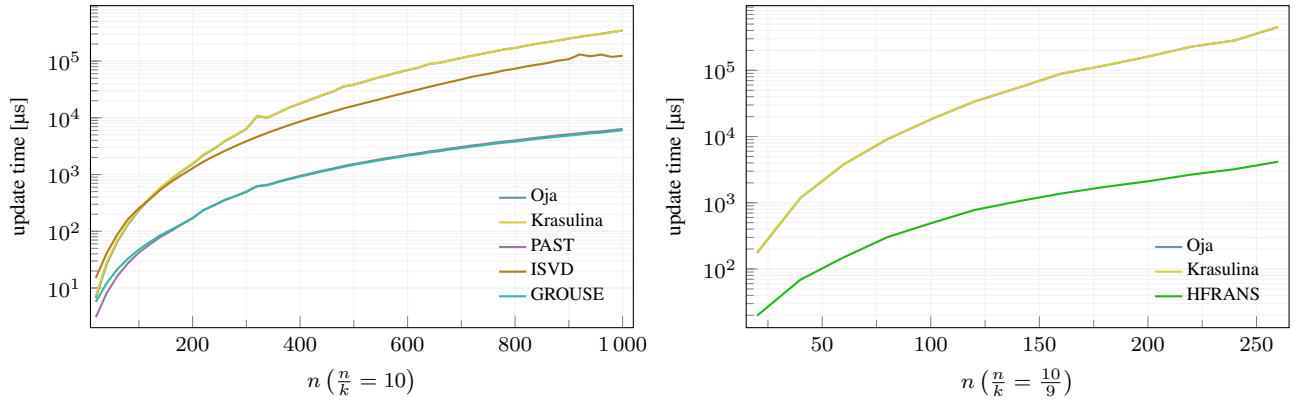
2) Register blocking: Matrix-matrix multiplication scales down to multiple subsequent vector-vector dot products performed through a sequence of multiply-and-accumulate operations whose result is an entry of the output matrix. By employing multiple accumulators simultaneously, i.e., by interleaving two or more dot products, local registers are utilised more efficiently, leading to a shorter execution time. Fig. 7(a) illustrates a graphical representation of this technique.

3) Buffered multiplication: Multiplying a $n \times k$ matrix by a $k \times k$ matrix and its transposition ($k \times k$ times $k \times n$ matrix-matrix) is common, e.g., in Oja, Krasulina, and ISVD methods. Classical implementation requires a memory space of $(2n + k)k$ values, which, however, can be almost halved if $n \gg k$ by storing the output matrix in the same memory location of the first input matrix. This overwriting procedure is possible if, for each output row, the input row is temporarily copied in a k-sized buffer, and therefore it comes at the cost of a slight increase in the computation time. This technique is illustrated in Fig. 7(b).

[6]The values in Table IV refer to memory footprint for methods implemented without the buffered multiplication technique described in Appendix A. When this technique is applied, the memory requirements for the Oja and Krasulina methods increase by about 45%, but the computation is about 19% faster.

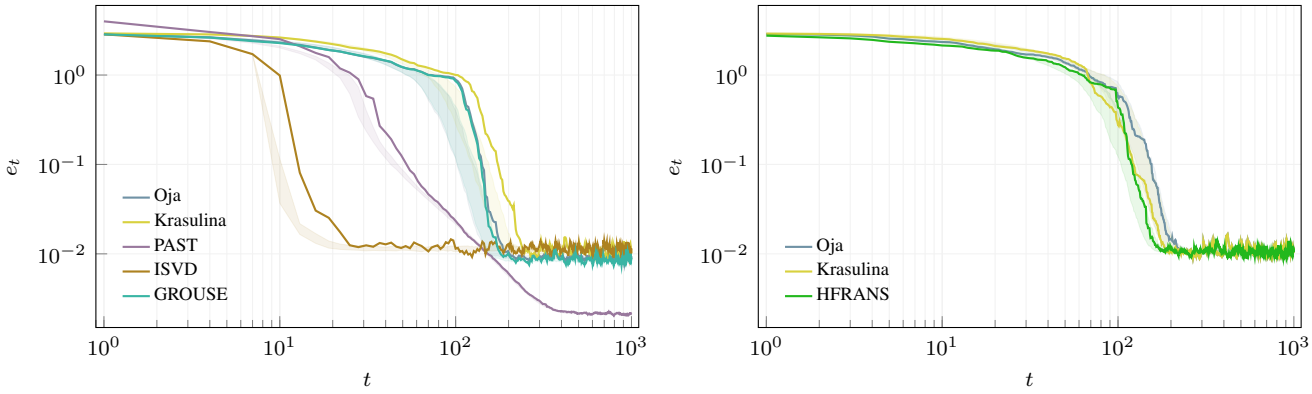
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529



(a) (b)

Fig. 5. Maximum time (over 10 trials) for each considered method to run the update step on STM32H743ZIT (rev. V) @ 480 MHz for different input dimension n and fixed ratio n/k. (a) methods PSA with $n/k = 10$; (b) methods for MSA with $n/k = 10/9$;



(a) (b)

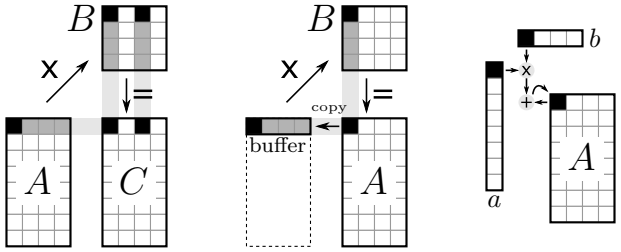
Fig. 6. Performance comparison for subspace identification on STM32H743ZIT (rev. V) with $n = 100$, $m = 10$, a single random instance of Φ, and noise $\rho = 10^{-3}$: (a) PSA ($k = 10$); (b) MSA ($k = 90$). Shaded areas indicate the 50% spread of Fig. 3 for comparison purposes.

4) Vector outer product and matrix addition merging: Some methods (Krasulina, PAST, GROUSE, HFRANS) require operations of the type $A = A + ab^\top$, where $A \in \mathbb{R}^{n\times k}$, $a \in \mathbb{R}^n$ and $b \in \mathbb{R}^k$. Instead of firstly evaluating the outer product $B = ab^\top$ and then performing the sum $A = A + B$, one can directly sum each entry of $ab^\top$ to each entry of A while performing the outer product, thus reducing memory requirements and the number of operations. This is shown in Fig. 7(c).

5) Transposition of square matrices: In general, the transposition of a rectangular matrix requires the copy of the entire matrix in another memory space. In the case of a square matrix, transposition can overwrite the original matrix by swapping each value in the two triangular parts.

6) Transposed multiplications: The operations $AB^\top$ and $A^\top B$ can avoid the transpose operation by modifying the multiplication operation and scanning the transposed matrix row-first instead of column-first (or vice-versa).

7) Column concatenation: ISVD method requires the concatenation of a column vector. Columns are not memory-contiguous while rows are. Therefore, transposing the whole method to turn column concatenation into row concatenation reduces memory space and computation time.



(a) (b) (c)

Fig. 7. Matrix arithmetic optimization techniques. (a) register blocking: interleaving multiple dot-products for a more efficient usage of registers, (b) buffered multiplication: buffering the row of the input matrix so that it can be overwritten with the output, (c) vector outer product and matrix addition merging: overwriting each element of the input matrix with the result of the operation.

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529

REFERENCES

[1] Y. Cui, J. Wang, H. Sun, H. Jiang, K. Yang, and J. Zhang, "Gridless underdetermined doa estimation of wideband lfm signals with unknown amplitude distortion based on fractional fourier transform," *IEEE Internet of Things Journal*, vol. 7, no. 12, pp. 11 612–11 625, 2020.

[2] Y. Tian, S. Liu, W. Liu, H. Chen, and Z. Dong, "Vehicle positioning with deep learning-based direction-of-arrival estimation of incoherently distributed sources," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[3] X. Wang, L. T. Yang, D. Meng, M. Dong, K. Ota, and H. Wang, "Multi-uav cooperative localization for marine targets based on weighted subspace fitting in sagin environment," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5708–5718, 2022.

[4] P. Gerstoft, Y. Hu, M. J. Bianco, C. Patil, A. Alegre, Y. Freund, and F. Grondin, "Audio scene monitoring using redundant ad hoc microphone array networks," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4259–4268, 2022.

[5] X. Wang and H. Poor, "Blind multiuser detection: a subspace approach," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 677–690, 1998.

[6] L. Sun, G. Bi, and L. Zhang, "Orthonormal subspace tracking algorithm for space–time multiuser detection in multipath cdma channels," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 6, pp. 3838–3845, 2007.

[7] E. E. Petrosky, A. J. Michaels, and D. B. Ridge, "Network scalability comparison of ieee 802.15.4 and receiver-assigned cdma," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6060–6069, 2019.

[8] A. Burrello, A. Marchioni, D. Brunelli, S. Benatti, M. Mangia, and L. Benini, "Embedded streaming principal components analysis for network load reduction in structural health monitoring," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4433–4447, 2021.

[9] V. Senger and R. Tetzlaff, "New signal processing methods for the development of seizure warning devices in epilepsy," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 5, pp. 609–616, 2016.

[10] B. Yu, T. Mak, X. Li, F. Xia, A. Yakovlev, Y. Sun, and C. S. Poon, "Real-time FPGA-based multichannel spike sorting using hebbian eigenfilters," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 4, pp. 502–515, 12 2011.

[11] A. Marchioni, M. Mangia, F. Pareschi, R. Rovatti, and G. Setti, "Subspace Energy Monitoring for Anomaly Detection @Sensor or @Edge," *IEEE Internet of Things Journal*, pp. 1–1, 4 2020.

[12] T. Yu, X. Wang, and A. Shami, "Recursive principal component analysis-based data outlier detection and sensor data aggregation in iot systems," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2207–2216, 2017.

[13] E. Zugasti, M. Iturbe, I. Garitano, and U. Zurutuza, "Null is not always empty: Monitoring the null space for field-level anomaly detection in industrial iot environments," in *2018 Global Internet of Things Summit (GIoTS)*, 2018, pp. 1–6.

[14] K. Cha, C. Sadek, and Z. Asgharzadeh, "Anomaly and degradation detection using subspace tracking in streaming data," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 3476–3481.

[15] R. Roy and T. Kailath, "Esprit-estimation of signal parameters via rotational invariance techniques," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 7, pp. 984–995, 1989.

[16] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, 1986.

[17] I. T. Jolliffe, *Principal components analysis*. Elsevier, 1986.

[18] L. Balzano, Y. Chi, and Y. M. Lu, "Streaming PCA and Subspace Tracking: The Missing Data Case," *Proceedings of the IEEE*, vol. 106, no. 8, pp. 1293–1310, 2018.

[19] T. V. Marinov, P. Mianjy, and R. Arora, "Streaming Principal Component Analysis in Noisy Settings," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds. Stockholmsmässan, Stockholm Sweden: PMLR, 7 2018, pp. 3413–3422.

[20] N. Lassami, A. Aïssa-El-Bey, and K. Abed-Meraim, "Low cost sparse subspace tracking algorithms," *Signal Processing*, vol. 173, p. 107522, 8 2020.

[21] 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 *29th Annual Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, V. Feldman, A. Rakhlin, and O. Shamir, Eds., vol. 49. Columbia University, New York, New York, USA: PMLR, 23–26 Jun 2016, pp. 1147–1164.

[22] Z. Allen-Zhu and Y. Li, "First efficient convergence for streaming k-pca: A global, gap-free, and near-optimal rate," in *Annual Symposium on Foundations of Computer Science - Proceedings*, vol. 2017-Octob. IEEE Computer Society, 11 2017, pp. 487–492.

[23] C. Tang, "Exponentially convergent stochastic k-PCA without variance reduction," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, Beygelzimer A., F. d'Alché Buc, E. Fox, and R. Garnett., Eds. Curran Associates, Inc., 2019, pp. 12 393–12 404.

[24] N. Tripuraneni, N. Flammarion, F. Bach, M. I. Jordan, S. Bubeck, V. Perchet, and P. Rigollet, "Averaging Stochastic Gradient Descent on Riemannian Manifolds," in *Proceedings of Machine Learning Research*, S. Bubeck, V. Perchet, and P. Rigollet, Eds., vol. 75. Proceedings of Machine Learning Research, 2018, pp. 1–38.

[25] R. Arora, A. Cotter, K. Livescu, and N. Srebro, "Stochastic optimization for PCA and PLS," in *2012 50th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2012*, 2012, pp. 861–868.

[26] B. Yang, "Projection Approximation Subspace Tracking," *IEEE Transactions on Signal Processing*, vol. 43, no. 1, pp. 95–107, 1995.

[27] I. M. Johnstone, "On the distribution of the largest eigenvalue in principal components analysis," *Ann. Statist.*, vol. 29, no. 2, pp. 295–327, 04 2001.

[28] 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 2010*, 2010, pp. 704–711.

[29] Y. Chi, Y. C. Eldar, and R. Calderbank, "PETRELS: Subspace estimation and tracking from partial observations," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2012, pp. 3301–3304.

[30] A. Edelman, T. A. Arias, and S. T. Smith, "The Geometry of Algorithms with Orthogonality Constraints," *SIAM Journal on Matrix Analysis and Applications*, vol. 20, no. 2, pp. 303–353, 1 1998.

[31] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of Mathematical Biology*, vol. 15, no. 3, pp. 267–273, 1982.

[32] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *Journal of Mathematical Analysis and Applications*, vol. 106, no. 1, pp. 69–84, 2 1985.

[33] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[34] I. Mitliagkas, C. Caramanis, and P. Jain, "Memory limited, streaming pca," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013.

[35] M. Hardt and E. Price, "The Noisy Power Method: A Meta Algorithm with Applications," in *Advances in Neural Information Processing Systems 27*, G. Z., W. M., C. C., L. N. D., and W. K. Q., Eds. Curran Associates, Inc., 2014, pp. 2861–2869.

[36] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012, vol. 3.

[37] T. P. Krasulina, "The method of stochastic approximation for the determination of the least eigenvalue of a symmetrical matrix," *USSR Computational Mathematics and Mathematical Physics*, vol. 9, no. 6, pp. 189–195, 1 1969.

[38] ——, "Method of stochastic approximation in the determination of the largest eigenvalue of the mathematical expectation of random matrices," *Automation and Remote Control*, no. 2, pp. 215–221, 1970.

[39] S. Attallah, "The generalized Rayleigh's quotient adaptive noise subspace algorithm: a householder transformation-based implementation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 1, pp. 3–7, 1 2006.

[40] S. Attallah and K. Abed-Meraim, "Low-cost adaptive algorithm for noise subspace estimation," *Electronics Letters*, vol. 38, no. 12, pp. 609–611, 2002.

[41] K. Abed-Meraim, A. Chkeif, and Y. Hua, "Orthogonal Oja Algorithm," *IEEE Signal Processing Letters*, vol. 7, no. 5, pp. 116–119, 5 2000.

[42] S. Attallah and K. Abed-Meraim, "Fast algorithms for subspace tracking," *IEEE Signal Processing Letters*, vol. 8, no. 7, pp. 203–206, 2001.

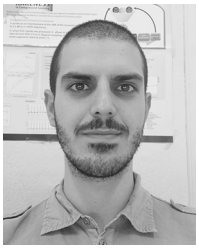
[43] B. Yang, "Subspace tracking based on the projection approach and the recursive least squares method," in *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Minneapolis, MN, USA: Institute of Electrical and Electronics Engineers (IEEE), 4 1993, pp. 145–148.

[44] K. Abed-Meraim, A. Chkeif, and Y. Hua, "Fast orthonormal PAST algorithm," *IEEE Signal Processing Letters*, vol. 7, no. 3, pp. 60–62, 3 2000.

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529

[45] T. Gustafsson, "Instrumental variable subspace tracking using projection approximation," *IEEE Transactions on Signal Processing*, vol. 46, no. 3, pp. 669–681, 1998.
[46] Y. Miao and Y. Hua, "Fast subspace tracking and neural network learning by a novel information criterion," *IEEE Transactions on Signal Processing*, vol. 46, no. 7, pp. 1967–1979, 1998.
[47] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra and Its Applications*, vol. 415, no. 1, pp. 20–30, 5 2006.
[48] R. Kennedy, C. J. Taylor, and L. Balzano, "Online completion of Ill-conditioned low-rank matrices," in *2014 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2014*. Institute of Electrical and Electronics Engineers Inc., 2 2014, pp. 507–511.
[49] D. Zhang and L. Balzano, "Global convergence of a Grassmannian gradient descent algorithm for subspace estimation," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, A. Gretton and C. C. Robert, Eds. Cadiz, Spain: PMLR, 2016, pp. 1460–1468.
[50] L. Balzano and S. J. Wright, "Local Convergence of an Algorithm for Subspace Identification from Partial Data," *Foundations of Computational Mathematics*, vol. 15, no. 5, pp. 1279–1314, 10 2015.
[51] E. Oja, "Principal components, minor components, and linear neural networks," *Neural Networks*, vol. 5, no. 6, pp. 927–935, 11 1992.
[52] R. Badeau, G. Richard, B. David, and K. Abed-Meraim, "Approximated power iterations for fast subspace tracking," in *Seventh International Symposium on Signal Processing and Its Applications, 2003. Proceedings.* IEEE, 2003, pp. 583–586.
[53] R. Badeau, B. David, and G. Richard, "Fast approximated power iteration subspace tracking," *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2931–2941, 8 2005.
[54] J. R. Bunch and C. P. Nielsen, "Updating the singular value decomposition," *Numerische Mathematik*, vol. 31, no. 2, pp. 111–129, 6 1978.
[55] L. Balzano and S. J. Wright, "On GROUSE and incremental SVD," in *2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, CAMSAP 2013*, 2013, pp. 1–4.
[56] R. Schreiber and C. Van Loan, "A Storage-Efficient WY Representation for Products of Householder Transformations," *SIAM Journal on Scientific and Statistical Computing*, vol. 10, no. 1, pp. 53–57, Jan. 1989.
[57] Fuyun Ling, D. Manolakis, and J. Proakis, "A recursive modified Gram-Schmidt algorithm for least-squares estimation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 4, pp. 829–836, Aug. 1986.
[58] T. Terao, K. Ozaki, and T. Ogita, "LU-Cholesky QR algorithms for thin QR decomposition," *Parallel Computing*, vol. 92, p. 102571, Apr. 2020.
[59] S. Yin, S. X. Ding, X. Xie, and H. Luo, "A Review on Basic Data-Driven Approaches for Industrial Process Monitoring," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 11, pp. 6418–6428, Nov. 2014.
[60] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, vol. 14, no. 5, pp. 403–420, Apr. 1970.



Alex Marchioni (S'17-M'22) received the B.Sc. and M.Sc. degrees (with honors) in Electronic engineering and the Ph.D. degree in Electronic, Telecommunication, and Information Technology from the University of Bologna, Bologna, Italy, in 20011, 2015, and 2022, respectively. He is currently a Research Fellow with the Department of Electrical, Electronic, and Information Engineering (DEI) of the University of Bologna. He is also a member of the Statistical Signal Processing Group at the University of Bologna. His research interests are in Signal Processing, Machine Learning, Anomaly Detection, Compressed Sensing, Internet of Things, and Big Data analytics.



Luciano Prono received the M.Sc. degree in Electronics Engineering in 2019 and the Ph.D. degree "cum laude" in Electrical, Electronics and Communication Engineering in 2023 from Politecnico di Torino, Turin, Italy, where he is currently working as a postodoctoral researcher with the Department of Electronics and Telecommunications. He was a visiting Ph.D. student at the Institute of Neuroinformatics (University of Zurich and ETH) in 2022. His research interests are in low-power systems and applications of artificial intelligence, mainly in the fields of IoT, tinyML, edge computing, compressed sensing and neuromorphic computing.



Mauro Mangia (S'09-M'13) received the B.Sc. and M.Sc. degrees in electronic engineering and the Ph.D. degree in information technology from the University of Bologna, Bologna, Italy, in 2005, 2009, and 2013, respectively. He was a Visiting Ph.D. Student with the Ecole Polytechnique Federale de Lausanne in 2009 and 2012. He is currently an Assistant Professor with the Department of Electrical, Electronic and Information Engineering of the University of Bologna. He is also a member of ARCES, Statistical Signal Processing Group, University of Bologna. His research interests are in nonlinear systems, machine learning, compressed sensing, anomaly detection, Internet of Things, Big Data analytics and optimization.

He was a recipient of the 2013 IEEE CAS Society Guillemin-Cauer Award and of the 2019 IEEE BioCAS Transactions Best Paper Award. He received the Best Student Paper Award at ISCAS2011. He was the Web and Social Media Chair for ISCAS2018.



Fabio Pareschi (Senior Member, IEEE) received the Dr. Eng. degree (Hons.) in electronic engineering from the University of Ferrara, Ferrara, Italy, in 2001, and the Ph.D. degree in information technology from the University of Bologna, Bologna, Italy, in 2007, under the European Doctorate Project (EDITH). He is currently an Associate Professor with the Department of Electronic and Telecommunication, Politecnico di Torino, Turin, Italy. He is also a Faculty Member with ARCES, University of Bologna. His research activity focuses on analog and mixed-mode electronic circuit design, statistical signal processing, compressed sensing, dc-dc converters, random number generation and testing, and electromagnetic compatibility. Dr. Pareschi was a recipient of the 2019 IEEE BioCAS Transactions Best Paper Award. He also received the Best Paper Award at ECCTD 2005 and the Best Student Paper Award at EMC Zurich 2005 and IEEE EMCCompo 2019. He served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS from 2010 to 2013. He is currently Associate Editor for the IEEE OPEN JOURNAL OF CIRCUITS AND SYSTEMS.

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3256529



Riccardo Rovatti (M'99-SM'02-F'12) received the M.S. degree in electronic engineering and the Ph.D. degree in electronics, computer science, and telecommunications from the University of Bologna, Italy, in 1992 and 1996, respectively. He is currently a Full Professor of electronics with the University of Bologna. He has authored more than 300 technical contributions to international conferences and journals and two volumes. His research focuses on mathematical and applicative aspects of statistical signal processing, on machine learning for signal processing, and on the application of statistics to nonlinear dynamical systems.

He was Distinguished Lecturer of the IEEE CAS Society for the years 2017-2018. He was a recipient of the 2004 IEEE CAS Society Darlington Award, the 2013 IEEE CAS Society Guillemin-Cauer Award and the 2019 IEEE BioCAS Transactions Best Paper Award. He received the Best Paper Award at ECCTD 2005 and the Best Student Paper Award at the EMC Zurich 2005 and ISCAS 2011. He is an IEEE fellow for his contribution to nonlinear and statistical signal processing applied to electronic systems.



Gianluca Setti received a Dr. Eng. degree (honors) and a Ph.D. degree in Electronic Engineering from the University of Bologna, in 1992 and in 1997. From 1997 to 2017 he was with the Department of Engineering, University of Ferrara, Italy, as an Assistant- (1998-2000), Associate- (2001-2008) and as a Professor (2009-2017) of Circuit Theory and Analog Electronics. From 2017 to 2022 he was a Professor of Electronics, Signal and Data Processing at the Department of Electronics and Telecommunications (DET) of Politecnico di Torino, Italy. Since November 2022, Dr. Setti is the Dean of the Computer, Electrical, Mathematical Sciences and Engineering (CEMSE) at the King Abdullah University of Science and Technology (KAUST), Saudi Arabia.

He held several positions as Visiting Professor/Scientist at EPFL (2002, 2005), UCSD (2004), IBM (2004, 2007) and at the University of Washington (2008, 2010) and he is also a permanent (in-kind) faculty member of ARCES, University of Bologna. His research interests include nonlinear circuits, recurrent neural networks, electromagnetic compatibility, compressive sensing and statistical signal processing, biomedical circuits and systems, power electronics, design and implementation of IoT nodes, circuits and systems for machine learning, and applications of AI techniques for anomaly detection and predictive maintenance.

Dr. Setti received the 2013 IEEE CAS Society Meritorious Service Award, he is co-recipient of the 2004 IEEE CAS Society Darlington Award, and the 2013 IEEE CAS Society Guillemin-Cauer Award, the 2019 IEEE Transactions on Circuits and Systems Best Paper Award, as well as of the best paper award at ECCTD2005, and the best student paper award at EMCZurich2005 and at ISCAS2011. He also received the 1998 Caianiello prize for the best Italian Ph.D. thesis on Neural Networks. Since 2016 he is also a Fellow of the IEEE.

He served as an Associate Editor for the IEEE Transactions on Circuits and Systems - Part I (1999-2002 and 2002-2004) and for the IEEE Transactions on Circuits and Systems - Part II (2004-2007), as the Deputy-Editor-in-Chief, for the IEEE Circuits and Systems Magazine(2004-2007), as well as the Editor-in-Chief for the IEEE Transactions on Circuits and Systems - Part II (2006-2007) and of the IEEE Transactions on Circuits and Systems - Part I (2008-2009). He also served in the editorial Board of IEEE Access (2013-2015) and of the Proceedings of the IEEE (2015-2018). Since 2019 he is serving as the first non US Editor-in-Chief of the Proceedings of the IEEE, the flagship journal of the Institute. He was a Distinguished Lecturer of the IEEE CAS Society (2004-2005 and 2013-2014) of the same Society, a member of the CASS Board of Governors (2005-2008), and served as the 2010 CAS Society President.

In 2012, he was the Chair of the IEEE Strategic Planning Committee of the Publication Services and Products Board (PSPB-SPC) and in 2013-2014 he was the first non North-American Vice President of the IEEE for Publication Services and Products. Dr. Setti served in the program committee of many conferences and was, in particular, the Special Sessions Co-Chair of ISCAS2005 (Kobe) and ISCAS2006 (Kos), the Technical Program Co-Chair of NDES2000 (Catania), ISCAS2007 (New Orleans), ISCAS2008 (Seattle), ICECS2012 (Seville), BioCAS2013 (Rotterdam), MWSCAS2023 (Phoenix) as well as the General Co-Chair of NOLTA2006 (Bologna) and ISCAS2018 (Florence).

He is co-editor of the book Chaotic Electronics in Telecommunications (CRC Press, Boca Raton, 2000), Circuits and Systems for Future Generation of Wireless Communications (Springer, 2009) and Design and Analysis of Biomolecular Circuits (Springer, 2011), co-author of the book Adapted Compressed Sensing for Effective Hardware Implementations (2018) as well as one of the guest editors of the May 2002 special issue of the IEEE Proceedings on "Applications of Non-linear Dynamics to Electronic and Information Engineering".

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.