PHAST: MODEL-FREE PHASELESS SUBSPACE TRACKING

Seyedehsara Nayer and Namrata Vaswani

Iowa State University, Ames, IA, USA.

ABSTRACT

Phaseless subspace tracking is the problem of recovering a time sequence of discrete signals from magnitude-only measurements of their linear projections, when the true signal lies in a low-dimensional subspace that can change with time. A typical assumption used in a lot of work is that the subspace changes gradually over time. We define this as (i) the maximum principal angle between the old and new subspaces is not too large (less than 90 degrees) or the number directions that changes is few or both; and (ii) the delay between subspace change times is large enough. This paper presents a novel algorithm, that we call PhaST, for model-free, mini-batch and fast Phaseless Subspace Tracking. We show via experiments that PhaST is significantly faster, and significantly more memory-efficient, than an existing algorithm for low-rank phase retrieval (which can be interpreted as a batch version of phaseless subspace tracking that does not assume anything about subspace changes). When fewer measurements are available, it also has significantly better recovery performance than both LRPR and single signal phase retrieval methods when its structural assumptions are valid.

Index Terms— Phase retrieval, PCA, low-rank

1. INTRODUCTION

The Phase Retrieval (PR) problem occurs in many applications such as ptychography, crystallography, astronomy. The original PR problem involves recovering an n length signal $\boldsymbol{\ell}$ from the magnitudes of its Discrete Fourier Transform (DFT) coefficients. Generalized PR (see [1] and [2]) replaces DFT by inner products with any set of measurement vectors, $\boldsymbol{d}_i$. Thus, the goal is to recover $\boldsymbol{\ell}$ from $|\langle \boldsymbol{d}_i, \boldsymbol{\ell}\rangle|$, $i = 1, 2, \dots, m$. It is clear that, without extra assumptions, PR will require $m \geq n$. In recent works, structural assumptions such as sparsity (see [3, 4, 5]) or low-rank (see [2]) have been incorporated into the PR problem in order to reduce the number of measurements m required for exact or accurate recovery. Low-rank has been used in two ways. One is to assume that a single signal re-arranged as a matrix (or a single image) is itself approximately or exactly low-rank. The goal is to recover this low-rank "signal" from its phaseless linear projections [6, 7]. The second is to assume that a time sequence of signals (or vectorized images) together form a matrix that is well modeled as being low-rank. Each signal/image is one column of this matrix. The measurements are phaseless linear projections of each signal or image (each column of the matrix) [2]. This problem has been referred to as "Low-Rank Phase Retrieval (LRPR)" in [2] where it was first studied.

Another way to interpret the LRPR problem is as follows [2]. We need to recover a time sequence of signals $\boldsymbol{\ell}_t$, $t = 1, 2, \dots, d$, that are assumed to be generated from an unknown low dimensional subspace. Mathematically, $\boldsymbol{\ell}_t = \boldsymbol{P}\boldsymbol{a}_t$, where $\boldsymbol{P}$ is an $n \times r$ basis matrix (tall matrix with mutually orthonormal columns) with $r \ll n$, and $\boldsymbol{a}_t$ is an $r \times 1$ coefficients' vector. For each $\boldsymbol{\ell}_t, t = 1, 2, \dots, d$, we have access to m phaseless measurements, $y_{i,t} = |\langle \boldsymbol{d}_{i,t}, \boldsymbol{\ell}_t\rangle|, i = 1, 2, ...m, t = 1, 2, ...d$. The goal is to either just recover the subspace, $\mathrm{span}(\boldsymbol{P})$, or to recover both $\mathrm{span}(\boldsymbol{P})$ and the coefficients and hence recover the signals $\boldsymbol{\ell}_t$'s (equivalently, recover the low-rank matrix $\boldsymbol{L} := [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \dots, \boldsymbol{\ell}_d]$).

This work studies the Phaseless Subspace Tracking problem. This can be simply understood as the dynamic or time-varying subspace extension of LRPR. Thus, instead of the subspace $\mathrm{span}(\boldsymbol{P})$ being fixed, we assume that it can change with time, albeit slowly. Time varying subspaces is a better model (than just fixed subspace) for long data sequences, e.g., long image sequences or videos, because if one tries to use a single low-dimensional subspace to represent the entire data sequence, the required subspace dimension may end up being quite large. This can be problematic because it means that the resulting data matrix may not be sufficiently low-rank.

The most general model for time-varying subspaces is to allow the subspace to change at each time. However such a model involves too many unknowns. An r dimensional subspace in n-dimensional ambient space is fully specified by nr parameters. But the signal $\boldsymbol{\ell}_t$ is an $n \times 1$ vector (has only n unknowns). Thus, allowing the subspace to change at each time will result in an increase in the number of unknowns per time instant from n to nr (rather than a decrease which is the purpose of incorporating structure into the PR problem). A less general model, but one that prevents an increase in the number of unknowns, is to assume that the data subspace is piecewise constant with time. This model has been extensively used in the robust subspace tracking literature [8, 9, 10]. We will use this model here as well. We explain it next.

Notation. $\|\cdot\|$ denotes the l_2 norm of a vector or the induced l_2 norm of a matrix. For other l_p norms, we use $\|\cdot\|_p$.

A matrix with mutually orthonormal columns is referred to as a *"basis matrix"*. For basis matrix $\boldsymbol{P}$ we let $\boldsymbol{P}^{\perp}$ to be

a basis for the subspace orthogonal to subspace spanned by $\boldsymbol{P}$. For basis matrices $\hat{\boldsymbol{P}}, \boldsymbol{P}$, the subspace error (SE) between their respective column spans is quantified by $\mathrm{SE}(\hat{\boldsymbol{P}}, \boldsymbol{P}) := \|\hat{\boldsymbol{P}}^{\perp}\boldsymbol{P}\|$. This measures the sine of the principal angle between the subspaces.

For any two integers i_1, i_2, the interval $[i_1 : i_2]$ denotes the set of integer values $\{i_1, i_1+1, \ldots, i_2\}$ and interval $[i_1 : i_2)$ denotes the set $\{i_1, i_1+1, \ldots, i_2-1\}$. Also, for integer values of t and α we let $[t; \alpha] = \{t-\alpha+1, \ldots, t\}$.

Problem Setting. For the identifiability reason explained above, we assume a piecewise constant (with time) model on subspace change. Denote the subspace change times by t_j for $j = 1, 2, \ldots, J$ and let $t_0 = 0$. We assume that $\boldsymbol{\ell}_t = \boldsymbol{P}_j \boldsymbol{a}_t$ for all $t = t_j, t_j+1, t_j+2, \ldots, t_{j+1}-1$. The goal is to recover the $\boldsymbol{\ell}_t$'s and their subspaces from m phaseless measurements at each time, i.e., from $y_{i,t} := |\langle \boldsymbol{d}_{i,t}, \boldsymbol{\ell}_t \rangle|$, $i = 1, 2, \ldots, m$ for each $t = 1, 2, \ldots, d$. Here, $\boldsymbol{d}_{i,t}$ is measurement vector and we let $\boldsymbol{D}_t = [\boldsymbol{d}_{1,t}, \boldsymbol{d}_{2,t}, \ldots, \boldsymbol{d}_{m,t}]$ to be measurement matrix at time t. Similarly, we define phaseless measurement vector $\boldsymbol{y}_t = [y_{1,t}, y_{2,t}, \ldots, y_{m,t}]'$ at time t. Then, $\boldsymbol{y}_t$ can be presented as $\boldsymbol{y}_t = |\boldsymbol{D}_t' \boldsymbol{\ell}_t|$.

We assume that the subspace changes are gradual or "slow". This means the following:

- $\mathrm{SE}(\boldsymbol{P}_{j-1}, \boldsymbol{P}_j) \le \Delta \ll 1$, or only a few subspace directions change at each t_j, or both (we need at least one of these two assumptions to hold);
- and the delay between subspace change times, $q_j := \min_j (t_{j+1} - t_j)$, is large enough. Of course this delay needs to be at least r to even compute the subspace even when $\boldsymbol{\ell}_t$'s are directly available.

Under this model, the question is when can one solve this problem either using a smaller m per signal than what is needed for LRPR? Moreover, can we get a faster and more memory-efficient mini-batch solution instead of the LRPR algorithm which is a batch one (and hence slow and memory-intensive)?

The LRPR work [2] has demonstrated that just exploiting the low-rank assumption enables a reduction in the required m compared to simple PR done for each signal $\boldsymbol{\ell}_t$ individually. However, the time complexity of LRPR is roughly r times higher than that of simple PR; while its memory requirement is equal to the size of the entire batch of data. This can be prohibitive for large-sized image sequences. This work aims to achieve a significant speed up while also significantly reducing the memory complexity of the resulting mini-batch approach. Moreover, we also show empirically that our proposed algorithm, which we call "fast Phaseless Subspace Tracking (PhaST)", needs a smaller m or accurate recovery when compared with LRPR (and of course with simple PR, e.g., the truncated Wirtinger Flow (TWF) algorithm of [1].

Algorithm 1 An overview of the proposed PhaST algorithm; for simplicity we present a version of the algorithm that assumes t_j known; we explain the approach to detect t_j in the text. Our actual code used for all our experiments does not assume t_j known.

Input: $\{\boldsymbol{y}_t, \boldsymbol{D}_t\}$, K, α, $T_P = 2$, T_a.

Initialize: $j \leftarrow 0$, $k \leftarrow 1$.
$\hat{\boldsymbol{P}}_0 \leftarrow$ LRPR algorithm with $\{\boldsymbol{y}_t, \boldsymbol{D}_t\}_{t=0:t_{\text{train}}}$
for $t > t_{\text{train}}$ **do**
  if $t = t_j + \alpha$ **then**
    $\hat{\boldsymbol{P}}_{j,1}^{new} \leftarrow$ top r eigenvectors of $\tilde{\boldsymbol{Y}}_P(\hat{\boldsymbol{P}}_{j-1})$ with $\tilde{\boldsymbol{Y}}_P(\hat{\boldsymbol{P}})$ as given in (3).
    $\hat{\boldsymbol{P}}_{j,1}^{init} \leftarrow [\hat{\boldsymbol{P}}_{j-1}, \hat{\boldsymbol{P}}_{j,1}^{new}]$
    for $\tau \in [t; \alpha]$ **do**
      $\hat{\boldsymbol{a}}_\tau^{init} \leftarrow$ top eigenvector of $\boldsymbol{Y}_{\boldsymbol{a}}(\hat{\boldsymbol{P}}_{j,1}^{init})$ with $\boldsymbol{Y}_{\boldsymbol{a}}(\hat{\boldsymbol{P}})$ defined in (2).
      for $v \in [1 : T_a]$ **do**
        $\hat{\boldsymbol{C}}_\tau \leftarrow \text{Phase}(\boldsymbol{D}_\tau \hat{\boldsymbol{P}}_{j,1}^{init} \hat{\boldsymbol{a}}_\tau)$
        $\hat{\boldsymbol{a}}_\tau^{init} \leftarrow \arg\min_{\tilde{\boldsymbol{a}}} \|\hat{\boldsymbol{C}}_\tau \boldsymbol{y}_\tau - \boldsymbol{D}_\tau \hat{\boldsymbol{P}}_{j,1}^{init} \tilde{\boldsymbol{a}}\|^2$
      end for
    end for
    $\hat{\boldsymbol{L}}^{init} = \hat{\boldsymbol{P}}_{j,1}^{init} \hat{\boldsymbol{A}}_{t;\alpha}^{init}$
    $\hat{\boldsymbol{P}}_{j,1}^{(0)} \leftarrow$ top r singular vectors of $\hat{\boldsymbol{L}}^{init}$
  else if $t = t_j + k\alpha$, $k = 2, \ldots$ **then**
    $\hat{\boldsymbol{P}}_{j,k}^{(0)} \leftarrow \hat{\boldsymbol{P}}_{j,k-1}$.
  end if
  if $t = t_j + k\alpha$, $k = 1, \ldots$ **then**
    if $k = 1$ set $T_p \leftarrow 3$ else if $k = 2, \ldots, K$ set $T_p \leftarrow 2$
    for $v = 1, \ldots, T_p$ **do**
      for $\tau \in [t; \alpha]$ **do**
        $\hat{\boldsymbol{a}}_\tau \leftarrow$ top eigenvector of $\boldsymbol{Y}_{\boldsymbol{a}}(\hat{\boldsymbol{P}}_{j,k}^{(v-1)})$
      end for
      for $u = 1, \ldots, T_a$ **do**
        for $\tau \in [t; \alpha]$ **do**
          $\hat{\boldsymbol{C}}_\tau \leftarrow \text{Phase}\left(\boldsymbol{D}_\tau' \hat{\boldsymbol{P}}_{j,k}^{v-1} \hat{\boldsymbol{a}}_\tau\right)$
          $\hat{\boldsymbol{a}}_\tau \leftarrow \arg\min_{\boldsymbol{a}} \|\hat{\boldsymbol{C}}_\tau \boldsymbol{y}_\tau - \boldsymbol{D}_\tau \hat{\boldsymbol{P}}_{j,k}^{v-1} \boldsymbol{a}\|^2$
        end for
      end for
      $\hat{\boldsymbol{P}}_{j,k}^{(v)} \leftarrow \arg\min_{\boldsymbol{P}} \sum_{\tau \in [t;\alpha]} \sum_{i=1}^{m} \|\hat{\boldsymbol{C}}_\tau \boldsymbol{y}_\tau - \boldsymbol{D}_\tau{}' \boldsymbol{P} \hat{\boldsymbol{a}}_\tau\|^2$
    end for
    $\hat{\boldsymbol{P}}_{j,k} \leftarrow \hat{\boldsymbol{P}}_{j,k}^{(T_p)}$
    if $t = t_{j+1}$ **then**
      $\hat{\boldsymbol{P}}_j \longleftarrow \hat{\boldsymbol{P}}_{j,k}^{(T_p)}$, $k \leftarrow 0$, $j \leftarrow j+1$
    end if
    $k \leftarrow k+1$
  end if
  Output: $\hat{\boldsymbol{L}}_{j,k} := \hat{\boldsymbol{P}}_{j,k}^{(1)} \hat{\boldsymbol{A}}_{j,k}$ with $\hat{\boldsymbol{A}}_{j,k} := [\hat{\boldsymbol{a}}_\tau]_{[t;\alpha]}$
end for

Our Contribution. This work takes the first steps towards a mini-batch and model-free phaseless subspace tracking algorithm that we call "fast Phaseless Subspace Tracking (PhaST)". We show via experiments that PhaST is significantly faster, and significantly more memory-efficient than LRPR, while still being more accurate than LRPR when m is small. Of course this holds only when the slow changing subspace assumption holds. Under this assumption, it also needs a smaller m than simple (single signal) PR methods. These methods need $m \geq n$, while both LRPR and PhaST can tolerate smaller m's because they exploit structural assumptions.

We studied a preliminary model-based version of the phaseless subspace tracking problem in [11]. That work assumed a very specific model on subspace change: it required that only one subspace direction change at each time while the rest of the subspace needed be fixed. Moreover, it needed a very accurate estimate of the previous subspace before it could detect or track changes; and it did not provide any experimental verification of a true tracking approach with multiple subspace changes. This work removes all of these requirements and demonstrates automatic detection and tracking of subspace changes.

2. FAST PHASELESS SUBSPACE TRACKING (PHAST)

Our proposed algorithm, which we term "PhaST", consists of two parts. The first part is to detect if there has been any change in the subspace. The second part is to update (obtain better and better estimates of) the changed subspace on-the-fly (in a mini-batch fashion).

To understand the main idea of the update steps, first assume that t_j is known. We split the update step into recovering α-consecutive-column-sub-matrices of $\boldsymbol{L}$ at a time. Let $\boldsymbol{L}_{t;\alpha} := [\boldsymbol{\ell}_{t-\alpha+1}, \boldsymbol{\ell}_{t-\alpha+2}, \ldots, \boldsymbol{\ell}_t]$ to be one α column interval in $[t_j : t_{j+1})$. Then $\boldsymbol{L}_{t;\alpha} := \boldsymbol{P}\boldsymbol{A}_{t;\alpha}$ in this interval with $\boldsymbol{P} = \boldsymbol{P}_j$. For each such sub-matrix, we have access to m measurements per column, i.e., we have $y_{i,\tau} := |\boldsymbol{d}_{i,\tau}{}'\boldsymbol{P}_j\boldsymbol{a}_\tau|$, $i \in [1:m]$, $\tau \in [t;\alpha]$; and we also have access to a subspace estimate $\hat{\boldsymbol{P}}$ available from the previous interval. Recall that $[t;\alpha] := [(t-\alpha+1) : t]$. Suppose $\mathrm{SE}(\hat{\boldsymbol{P}}, \boldsymbol{P}) \leq b$ with b small. We would like to solve for $\tilde{\boldsymbol{P}}$, $\tilde{\boldsymbol{a}}$ that minimize

$$\sum_{\tau\in[t;\alpha]} \sum_{i\in[1:m]} |y_{i,\tau} - |\boldsymbol{d}_{i,\tau}{}'\tilde{\boldsymbol{P}}\tilde{\boldsymbol{a}}_\tau||^2, \tag{1}$$

subject to the constraint that $\mathrm{SE}(\hat{\boldsymbol{P}}, \boldsymbol{P}) \leq b$. Alternatively when b is unknown, we could instead try to solve the Lagrangian (unconstrained) version of this problem. In either case, if we tried to develop an alternating minimization (alt-min) solution that alternates over $\boldsymbol{P}$, $\boldsymbol{a}_\tau$'s, and the measurements' phase matrices $\boldsymbol{C}_\tau := \mathrm{diag}(\mathrm{phase}(\boldsymbol{D}_\tau{}'\boldsymbol{P}\boldsymbol{a}_\tau))$, we would not get a closed-form update rule for $\boldsymbol{P}$. One trick that is often used in such situations is the following: an indirect way to impose a constraint of the form $distance(\mathbf{x}_0, \mathbf{x}) \leq b$ (small) when minimizing a cost function over $\mathbf{x}$ is to start the alt-min algorithm with initialization $\mathbf{x}_0$ and then to run only one (or a few) iterations of alt-min for the unconstrained problem over $\mathbf{x}$. In our setting, this means we initialize the alt-min algorithm with $\hat{\boldsymbol{P}}$, and run only one or two iterations of alt-min to minimize (1) over $\boldsymbol{P}$; of course for each value of $\boldsymbol{P}$, we run many iterations of alt-min over $\boldsymbol{a}_\tau$'s and $\boldsymbol{C}_\tau$'s. This also needs init for $\boldsymbol{a}_\tau$'s. Given a $\hat{\boldsymbol{P}}$, we initialize $\boldsymbol{a}_\tau$ using the approach introduced for LRPR [2]: compute $\boldsymbol{a}_\tau$ as the top eigenvector of

$$\tilde{\boldsymbol{Y}}_{\boldsymbol{a}}(\hat{\boldsymbol{P}}) := \hat{\boldsymbol{P}}' \left(\frac{1}{m} \sum_{i=1}^{m} y_{i,t}^2 \boldsymbol{d}_{i,t}\boldsymbol{d}_{i,t}' \right) \hat{\boldsymbol{P}}. \tag{2}$$

When b is not small, but the subspaces are still close in other metrics (for example, if a large part of the subspace does not change at all, c.f. the model of [11]), we can still leverage knowledge of $\hat{\boldsymbol{P}}$ as follows. We initialize the alt-min solution not with $\hat{\boldsymbol{P}}$, but with a spectral initialization step that uses $\hat{\boldsymbol{P}}$ to first find the "newly added subspace" (basis for $\hat{\boldsymbol{P}}^{\perp\prime}\boldsymbol{P}$), followed by using a simple trick to delete the removed directions. Let $\boldsymbol{P}_{\mathrm{add}}$ denote a basis for $\hat{\boldsymbol{P}}^{\perp\prime}\boldsymbol{P}$. Using a simple modification of the idea developed in our earlier work [11], our spectral initialization step computes $\hat{\boldsymbol{P}}_{\mathrm{add}}$ as the top r eigenvectors of the matrix

$$\tilde{\boldsymbol{Y}}_P(\hat{\boldsymbol{P}}) := (\boldsymbol{I} - \hat{\boldsymbol{P}}\hat{\boldsymbol{P}}') \left(\frac{1}{m\alpha} \sum_{\tau\in[t;\alpha]} \sum_{i\in[1:m]} y_{i,\tau}\boldsymbol{d}_{i,\tau}\boldsymbol{d}_{i,\tau}{}' \right) (\boldsymbol{I} - \hat{\boldsymbol{P}}\hat{\boldsymbol{P}}'). \tag{3}$$

We then let $\boldsymbol{P}^{init,temp} := [\hat{\boldsymbol{P}}, \hat{\boldsymbol{P}}_{\mathrm{add}}]$. Notice here that because of the initialization approach, $\hat{\boldsymbol{P}}^{init,temp}$ is a $n \times 2r$ matrix. Using this we initialize the coefficient vectors, $\boldsymbol{a}_\tau$, (which are now $2r$ length vectors) as explained above but with $\hat{\boldsymbol{P}}$ is replaced by $\hat{\boldsymbol{P}}^{init,temp}$ followed by obtaining improved estimates via minimizing (1) by alt-min over $\boldsymbol{a}_\tau$ and $\boldsymbol{C}_\tau$ (phase matrix) with $\hat{\boldsymbol{P}}$ held constant at $\hat{\boldsymbol{P}}^{init,temp}$. We then set $\hat{\boldsymbol{L}}_{t;\alpha}^{init,temp} := \hat{\boldsymbol{P}}^{init,temp}\hat{\boldsymbol{A}}$. By compute its top r left singular vectors, we get a good r-dimensional initialization of the subspace. We denote this by $\hat{\boldsymbol{P}}^{init}$.

The complete algorithm is summarized in Algorithm 1. It splits the matrix $\boldsymbol{L}$ into α-consecutive column intervals, followed by recursively getting better and better estimates of the subspace $\boldsymbol{P} := \boldsymbol{P}_j$. When a subspace change is detected, the subspace change is assumed to be large (b is not small enough) and hence at this time the above initialization step is needed. After at least one estimate of the new subspace is computed, it assumes b is small enough and $\boldsymbol{P}$ can be initialized just with the previous final subspace estimate.

3. EXPERIMENTS

This section provides two distinct set of experiments. To generate the data we adopt a similar approach as in [11]

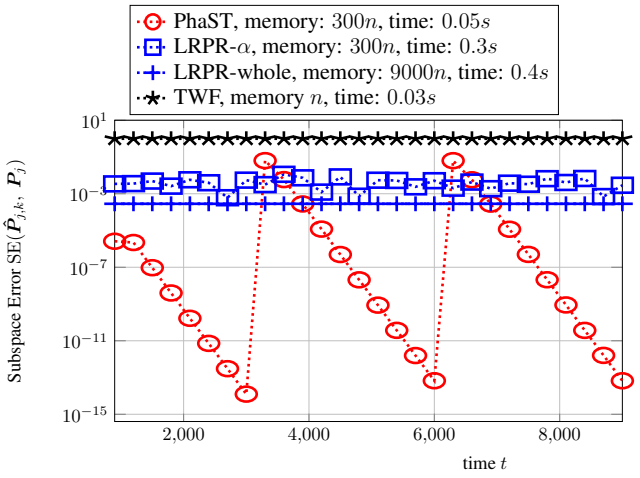

Fig. 1: Subspace error of $\boldsymbol{P}_j$ vs. time for $n = 500$, $m = 400$, $r = 10$, $r_{\text{add}} = 10$, and $\theta = 15$. Here, $J = 3$, $\alpha = 300$, and $t_0 = 0, t_1 = 3300, t_3 = 6300$. LRPR-$\alpha$ shows subspace error when it is fed with one mini-batch at each time. LRPR-whole shows error when data of all time sequences is fed to LRPR and the obtained error is repeated for the time to simplify the representation in this case. Average time taken per column and the memory complexity is reported in the legend. Memory needed by PhaST and LRPR-α is $n\alpha = 300n$. TWF needs smaller memory of size n and is faster.

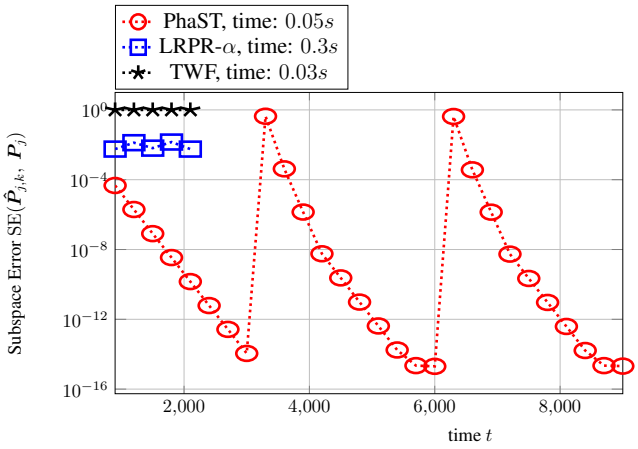

Fig. 2: Subspace error of $\boldsymbol{P}_j$ vs. time for $n = 500$, $m = 400$, $r = 10$, $r_{\text{add}} = 4$, and $\theta = 75$. Here, $J = 3$, $\alpha = 300$, and $t_0 = 0$, $t_1 = 3300$, $t_3 = 6300$. Average time taken per column is reported

for a general case with more than one direction is changing. Let $\boldsymbol{P}_{j-1,\text{chg}}$ denote directions from $\text{span}(\boldsymbol{P}_{j-1})$ that changes at t_j and let $\boldsymbol{P}_{j,\text{chd}}$ denote its changed version, then $\text{span}(\boldsymbol{P}_{j-1})$ can be written as $\text{span}([\boldsymbol{P}_{j-1,\text{fix}}, \boldsymbol{P}_{j-1,\text{chg}}])$ and $\text{span}(\boldsymbol{P}_j)$ as $\text{span}([\boldsymbol{P}_{j-1,\text{fix}}, \boldsymbol{P}_{j,\text{chd}}])$, where $\boldsymbol{P}_{j-1,\text{fix}}$ is an $n \times (r - r_{\text{add}})$ matrix corresponding to the fixed part of the subspace at t_j. This model allows only a part of the subspace to change. In our experiments here, we generate $\boldsymbol{P}_{j,\text{chd}}$ as $\boldsymbol{P}_{j,\text{chd}} = \boldsymbol{P}_{j,\text{add}} \sin\theta_j - \boldsymbol{P}_{j-1,\text{chg}} \cos\theta_j$ with a single angle for all changed columns (only for simplicity). Here $\boldsymbol{P}_{j,\text{add}}$ contains a set of $r_{\text{add}} \leq r$ directions from $\text{span}({\boldsymbol{P}_{j-1}}^{\perp})$. Beginning with $t_0 = 0$, the subspace changes at $t_1 = 3300$ and $t_2 = 6300$. All experiments are averaged over 50 Monte-Carlo repeats.

To illustrate the performance of proposed method, we consider two different scenarios. The first scenario models situations where the change is very small and aims at evaluating robustness of PhaST for cases with small values of θ. The second scenario is designed to show performance of the algorithm with larger values of θ. Both experiments are done for the setting with $n = 500$, $m = 400$, $r = 10$. PhaST uses $K = 3$, $\alpha = 300$, $T_P = 2$, $T_a = 10$. Detection is done with a similar approach as in [11]. The algorithm assumes that the subspace remains unchanged at least for three α time sequences. After three α time intervals of the last detected change, detection phase of the algorithm starts. Threshold constant for detection is set to 1.75 for both experiments. Figure 1 illustrates simulation results for the first scenario with $\theta = 15$ and $r_{\text{add}} = 10$. As figure suggests, PhaST is robust to small changes. Figure 2 belongs to the second scenario with $\theta = 75$, $r_{\text{add}} = 3$, and shows the ability of PhaST to detect larger changes.

To demonstrate the superiority of proposed algorithm, we compare it with two other existing state of the art PR methods, namely, LRPR and TWF. Comparison with LRPR can be done in two different ways. First is to feed LRPR with data of all time sequences. Hence $q = \sum_j q_j$ for LRPR. Although more measurements are available in this case, LRPR is unable to recover the subspace.

Another way of evaluating LRPR is to feed it with data of one mini-batch at each time. Since LRPR does not take advantage of the previous subspace estimates, it shows almost the same performance for different such batches. On the other hand TWF is also not taking advantage of the previous knowledge and hence its performance is almost the same for every column. Therefore since these approaches do not work for all time sequences as it can be seen in Fig. 1, we just report the average error of first 5 mini-batches for 50 independent trials in Fig. 2. Number of loop iterations for LRPR and TWF are 20. Number of power method iterations used in TWF initialization is 50 and parameter settings are the same as suggested in [1]. Simulation results confirm that PhaST needs less number of measurements to track the subspace than what is needed by TWF and LRPR. Besides it makes shorter delay than LRPR.

4. CONCLUSION

This paper proposed a novel algorithm for model-free, mini-batch and fast Phaseless Subspace Tracking and provided first simulation results for complete PST algorithm. Experiments show that PhaST is memory efficient and needs less sample complexity in comparison with existing PR methods. Providing convergence analysis and real world applications will be part of the future work.

5. REFERENCES

[1] Y. Chen and E. Candes, "Solving random quadratic systems of equations is nearly as easy as solving linear systems," in *Adv. Neural Info. Proc. Sys. (NIPS)*, 2015, pp. 739–747.

[2] N. Vaswani, S. Nayer, and Y. C. Eldar, "Low rank phase retrieval," *IEEE Trans. Sig. Proc.*, August 2017.

[3] K. Jaganathan, S. Oymak, and B. Hassibi, "Recovery of sparse 1-d signals from the magnitudes of their fourier transform," in *IEEE Intl. Symp. on Information Theory (ISIT)*. IEEE, 2012, pp. 1473–1477.

[4] Y. Shechtman, A. Beck, and Y. C. Eldar, "Gespar: Efficient phase retrieval of sparse signals," *IEEE Trans. Sig. Proc.*, vol. 62, no. 4, pp. 928–938, 2014.

[5] A. Szameit, Y. Shechtman, E. Osherovich, E. Bullkich, P. Sidorenko, H. Dana, S. Steiner, E. B. Kley, S. Gazit, T. Cohen-Hyams, S. Shoham, M. Zibulevsky, I. Yavneh, Y. C. Eldar, O. Cohen, and M. Segev, "Sparsity-based single-shot subwavelength coherent diffractive imaging," *Nature Materials*, vol. 11, pp. 455–9, Apr. 2012.

[6] S. Tu, R. Boczar, M. Soltanolkotabi, and B. Recht, "Low-rank solutions of linear matrix equations via procrustes flow," *arXiv preprint arXiv:1507.03566*, 2015.

[7] Q. Zheng and J. Lafferty, "A convergent gradient descent algorithm for rank minimization and semidefinite programming from random linear measurements," in *Adv. Neural Info. Proc. Sys. (NIPS)*, 2015.

[8] C. Qiu, N. Vaswani, B. Lois, and L. Hogben, "Recursive robust pca or recursive sparse recovery in large but structured noise," *IEEE Trans. Info. Th.*, pp. 5007–5039, August 2014.

[9] P. Narayanamurthy and N. Vaswani, "Provable dynamic robust pca or robust subspace tracking," *IEEE Trans. Info. Theory*, to appear, 2018.

[10] P. Narayanamurthy and N. Vaswani, "Nearly optimal robust subspace tracking," in *ICML 2018*, 2018, pp. 3701–3709.

[11] S. Nayer and N. Vaswani, "Phaseless subspace tracking," *Accepted for publication in IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2018.