ROBUST AND SCALABLE METHODS FOR THE DYNAMIC MODE DECOMPOSITION*

TRAVIS ASKHAM†, PENG ZHENG‡, ALEKSANDR ARAVKIN§, AND J. NATHAN KUTZ¶

Abstract. The dynamic mode decomposition (DMD) is a broadly applicable dimensionality reduction algorithm that decomposes a matrix of time-series data into a product of a matrix of exponentials, representing Fourier-like time dynamics, and a matrix of coefficients, representing spatial structures. This interpretable spatio-temporal decomposition is classically formulated as a nonlinear least squares problem, and solved within the variable projection framework. When the data contains outliers, or other features that are not well-represented by exponentials in time, the standard Frobenius norm misfit penalty creates significant biases in the recovered time dynamics. As a result, practitioners are left to clean such defects from the data manually or to use a black-box cleaning approach like robust PCA. As an alternative, we propose a robust statistical framework for the optimization used to compute the DMD itself. We also develop variable projection algorithms for these new formulations, which allow for regularizers and constraints on the decomposition parameters. Finally, we develop a scalable version of the algorithm by combining the structure of the variable projection framework with the stochastic variance reduction (SVRG) paradigm. The approach is tested on a range of synthetic examples, and the methods are implemented in an open source software package `RobustDMD`[1].

Key words. robust statistics, dynamic mode decomposition, scalable algorithms

AMS subject classifications. 37M02, 65P02, 62F35

1. Introduction.

Dimensionality reduction is a critically enabling tool in machine learning applications. Specifically, extracting the dominant low-rank features from a high-dimensional data matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ allows one to efficiently perform tasks associated with clustering, classification and prediction. As defined by [11], *linear* dimensionality reduction methods solve an optimization problem with objective $f_{\mathbf{X}}(\cdot)$ over a manifold $\mathcal{M}$ to produce a linear transformation P which maps the columns of $\mathbf{X}$ to a lower dimensional space. Many popular methods can be written in this framework by an appropriate definition of $f_{\mathbf{X}}(\cdot)$ and specification of the manifold $\mathcal{M}$. For instance, the principal component analysis (PCA) may be written as

$$\hat{M} = \arg\min_{M \in \mathcal{M}} \|\mathbf{X} - MM^{\intercal}\mathbf{X}\|_F \,, \qquad \mathcal{M} = \mathcal{O}^{m \times k} \,, \tag{1.1}$$

where $\mathcal{O}^{m \times k}$ is the manifold of $m \times k$ matrices with orthonormal columns, i.e. $\mathcal{M}$ is a Stiefel manifold. The map P is then given by $\hat{M}^{\intercal}$. One of the primary conclusions of the survey [11], is that — aside from the PCA itself — many of the common methods for linear dimensionality reduction based on eigenvalue solvers are actually

*Submitted to the editors DATE.

Funding: T. Askham and J. N. Kutz acknowledge support from the Air Force Office of Scientific Research (FA9550-15-1-0385). J. N. Kutz also acknowledges support from the Defense Advanced Research Projects Agency (DARPA contract HR0011-16-C-0016). The work of A. Aravkin and P. Zheng was supported by the Washington Research Foundation Data Science Professorship.

†Department of Applied Mathematics, New Jersey Institute of Techonology, Newark, NJ (askham@njit.edu).

‡Department of Health Metrics Sciences, University of Washington, Seattle, WA (zhengp@uw.edu).

§Department of Applied Mathematics, University of Washington, Seattle, WA (saravkin@uw.edu).

¶Department of Applied Mathematics, University of Washington, Seattle, WA (kutz@uw.edu).

[1] https://github.com/UW-AMO/RobustDMD.jl

sub-optimal heuristics and the direct solution of the optimization problem (1.1) should be preferred.

In this manuscript, we consider a particular linear dimensionality reduction technique: the dynamic mode decomposition (DMD). In the past decade, the DMD has been applied to the analysis of fluid flow experiments and simulations, machine learning enabled control systems, and Koopman spectral analysis, among other data-intensive problems described by dynamical systems. The success of the algorithm is largely due to the interpretability of the low-rank spatio-temporal modes it generates in approximating the dominant features of the data matrix $\mathbf{X}$. The DMD was originally defined to be the output of an algorithm for characterizing time-series measurements of fluid flow data [28, 27]. It was later reformulated by [30] as a least-squares regression problem whereby the DMD could be stably computed using a Moore-Penrose pseudo-inverse and an eigenvalue decomposition.

An earlier though less commonly used formulation, the *optimized* DMD [9], can be phrased as the optimization problem

$$\hat{M} = \arg\min_{M \in \mathcal{M}} \|\mathbf{X} - MM^{\dagger}\mathbf{X}\|_F \,, \quad \mathcal{M} = \mathbf{\Phi}(\mathbb{C}^k) \,, \tag{1.2}$$

where the map $\boldsymbol{\alpha} \mapsto \mathbf{\Phi}(\boldsymbol{\alpha})$ defines a matrix with columns corresponding to exponential time dynamics (see Section 2.1) and $M^{\dagger}$ denotes the Moore-Penrose pseudo-inverse of M. This can be thought of as a best-fit linear dynamical system approximation of the data. In most applications, it is this exponential model of the observed data which is the real object of interest, as it is this model which is used in forecasting and interpolation. Thus, the original DMD algorithm [28, 27] and the reformulation [30] end up being heuristics for finding approximate solutions of (1.2).

In agreement with the conclusions of [11], the optimized DMD, while more costly to compute, is more robust to additive noise than established heuristic methods based on eigensolvers, i.e. the exact DMD and its noise-corrected alternatives [12, 4]. It is also more flexible than the exact DMD, allowing for non-equispaced snapshots. While the optimized DMD does not fit directly into the optimization framework of [11], which is defined for $\mathcal{M}$ either a Stiefel manifold or a Grassmannian manifold, it can be computed efficiently using classical variable projection methods [16, 15, 4].

The DMD has been used in a variety of fields where the nature of the data can lead to corrupt and noisy measurements. This includes applications ranging from neuroscience [7] to video processing [17, 14] to fluid dynamics [28, 27, 18, 12]. Although the Frobenius norm used in the definition of the optimized DMD (1.2) is appealing due to its physical interpretability in many applications (energy, mass, etc.), it has significant flaws that can severely limit its applicability. Specifically, corrupt data or large noise fluctuations can lead to significant deformation of the DMD approximation of the data because the Frobenius norm implicitly assigns a very low probability to such outliers (see Section 2.3). In practice, these outliers are often removed from the data manually or using a black-box filtering approach like robust PCA [23, 32, 8]. However, such approaches ignore the structure of the DMD approximation and may introduce biases of their own. Further, it is desirable that DMD methods not only be robust to "noisy" outliers but also to non-exponential structure in the data.

Contributions. Here, we develop an automated approach to robust DMD. Specifically, we modify the optimized DMD definition (1.2) to incorporate ideas from the field of robust statistics [24, 20] in order to produce a decomposition that is significantly less sensitive to outliers in the data. Because the new problem formulation incorporates robust norms, many of the efficient strategies used in variable projection algorithms for problems defined in the Frobenius norm are no longer available. To

remedy this, we develop a number of algorithms based on modern variable projection methods [3, 2] which exploit the structure of the DMD for increased performance. In particular, we can incorporate nonsmooth features, such as regularizers and constraints, and scale to large problems using stochastic variance reduction techniques. This flexible architecture allows us to impose physically relevant constraints on the optimization that are critical for tasks such as future-state prediction. For instance, we can impose the constraint that the real parts of the DMD eigenvalues are nonpositive, thus ensuring that solutions do not grow to infinity when forecasting.

The effect of noise on the DMD is a well-studied area. Controlling for the bias of the exact DMD in the presence of additive noise was treated by [19] and [12]. A Bayesian formulation of the DMD was presented by [29]. This formulation is flexible enough to incorporate robust statistics but this was not a focus of that work. In [13], Dicle et al. presented a robust formulation of exact DMD type, which complements the current work.

The rest of this manuscript is organized as follows. In Section 2, we provide some necessary preliminaries from the DMD, robust statistics, and variable projection literature and we present our problem formulation. A detailed description of the algorithms we use to solve the robust DMD formulation follows in Section 3. We apply these methods to synthetic data in Section 4, demonstrating the effectiveness of the robust formulations. Finally, we provide some concluding remarks and describe possible future directions in Section 5.

2. Preliminaries.

In this section, we outline some of the precursors of this work and present our problem formulation.

2.1. Dynamic mode decomposition.

As mentioned above, the dynamic mode decomposition (DMD) corresponds to a best-fit linear dynamical model of the data.

Let $\mathbf{X} \in \mathbb{C}^{m\times n}$ be a snapshot matrix whose rows, $\mathbf{x}_j$, are samples of an n dimensional dynamical system at a set of m sample times t_j. If we suppose that the $\mathbf{x}_j$ arise from linear time dynamics, i.e.

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) \ ,$$

then

$$\mathbf{x}_j^\intercal = e^{t_j A}\mathbf{x}(0) \ .$$

Assuming a diagonalizable matrix A, this can be rewritten as

$$\mathbf{x}_j^\intercal = S\exp(t_j D)S^{-1}\mathbf{x}(0) \ ,$$

where D is a diagonal matrix made up of the eigenvalues of A and the columns of S are eigenvectors of A. We observe that each entry in $\mathbf{x}_j$ is then a linear combination of the terms $\exp(D_{11}t_j), \ldots, \exp(D_{nn}t_j)$. In the DMD setting, we make the further assumption that the samples, $\mathbf{x}_j$, project onto a relatively small number, $k \ll n$, of the eigenvectors. The optimized DMD problem is to then discover these eigenvalues and the coefficients of $\mathbf{x}(t)$ in the exponential basis based on the samples $\mathbf{x}_j$.

To be precise, for a given rank k, let $\boldsymbol{\alpha} \in \mathbb{C}^k$ be a vector of complex numbers corresponding to eigenvalues as in the above. We then define the matrix $\boldsymbol{\Phi}(\boldsymbol{\alpha}; \mathbf{t})$ by

$$\Phi_{ij}(\boldsymbol{\alpha}) = e^{\alpha_j t_i} \ . \tag{2.1}$$

When it is clear in context, we often drop the dependence of $\boldsymbol{\Phi}$ on $\boldsymbol{\alpha}$ and $\mathbf{t}$. Let $\mathbf{B} \in \mathbb{C}^{k\times n}$ be a matrix containing coefficients for each entry in $\mathbf{x}(t)$ in the exponential basis.

The so-called *optimized DMD* (see [9]) is defined to be the solution of the following optimization problem:

$$\min_{\boldsymbol{\alpha},\mathbf{B}} \frac{1}{2}\|\mathbf{X} - \boldsymbol{\Phi}(\boldsymbol{\alpha})\mathbf{B}\|_F^2 \,. \tag{2.2}$$

The problem (2.2) is a large, nonlinear least squares problem; in particular, it is non-convex and oscillatory (for complex-valued $\boldsymbol{\alpha}$). The classical variable projection framework provides an efficient method for computing a (local) solution.

2.2. Variable projection. Let

$$f_{\text{opt}}(\boldsymbol{\alpha},\mathbf{B}) = \frac{1}{2}\|\mathbf{X} - \boldsymbol{\Phi}(\boldsymbol{\alpha})\mathbf{B}\|_F^2 \,.$$

The classical variable projection (VP) framework is based on the observation that for a fixed $\boldsymbol{\alpha}$, it is easy to optimize f_{opt} in $\mathbf{B}$. In fact, for the least squares case, we have a closed form expression

$$\mathbf{B}(\boldsymbol{\alpha}) := \arg\min_{\mathbf{B}} f_{\text{opt}}(\boldsymbol{\alpha},\mathbf{B}) = \boldsymbol{\Phi}(\boldsymbol{\alpha})^\dagger \mathbf{X}, \tag{2.3}$$

where $\boldsymbol{\Phi}(\boldsymbol{\alpha})^\dagger$ denotes the Moore-Penrose pseudo-inverse of $\boldsymbol{\Phi}(\boldsymbol{\alpha})$. Let

$$\tilde{f}_{\text{opt}}(\boldsymbol{\alpha}) = \min_{\mathbf{B}} f_{\text{opt}}(\boldsymbol{\alpha},\mathbf{B}) := \frac{1}{2}\|\mathbf{X} - \boldsymbol{\Phi}(\boldsymbol{\alpha})\mathbf{B}(\boldsymbol{\alpha})\|_F^2.$$

The VP technique finds the minimizer of $\tilde{f}_{\text{opt}}(\boldsymbol{\alpha})$ using an iterative method.

First and second derivatives of $\tilde{f}$ with respect to $\boldsymbol{\alpha}$ are easily computed [6]:

$$\begin{aligned} \nabla_{\boldsymbol{\alpha}} \tilde{f}_{\text{opt}}(\boldsymbol{\alpha}) &= \partial_{\boldsymbol{\alpha}} f_{\text{opt}}|_{\boldsymbol{\alpha},\mathbf{B}(\boldsymbol{\alpha})} \\ \nabla^2_{\boldsymbol{\alpha}} \tilde{f}_{\text{opt}}(\boldsymbol{\alpha}) &= \left[\partial^2_{\boldsymbol{\alpha}} f_{\text{opt}} - \partial_{\boldsymbol{\alpha},\mathbf{B}} f_{\text{opt}} (\partial^2_{\mathbf{B}} f_{\text{opt}})^{-1} \partial_{\mathbf{B},\boldsymbol{\alpha}} f_{\text{opt}}\right]\Big|_{\boldsymbol{\alpha},\mathbf{B}(\boldsymbol{\alpha})} . \end{aligned} \tag{2.4}$$

These formulas allow first- and second-order methods to be directly applied to $\tilde{f}_{\text{opt}}$, including steepest descent, BFGS, and Newton's method. The matrix $\mathbf{B}(\boldsymbol{\alpha})$ is updated every time $\boldsymbol{\alpha}$ changes. Gauss-Newton and Levenberg-Marquardt (LM) have been classically used for exponential fitting; these methods do not use the Hessian in (2.4), opting for simpler approximations. The method was used for exponential fitting by [16].

While VP originally referred to least-squares projection (using the closed-form solution $\mathbf{B}(\boldsymbol{\alpha})$ in (2.3)), follow-up work considered more general loss functions, using the term *projection* to refer to partial minimization [3, 2].

For practitioners, the optimized DMD may be less familiar than exact DMD [30]. We favor the optimized DMD for its performance on data with additive noise (see [4]) and its flexibility. In particular, the optimized formulation enables the contributions of the current work. For a review of the DMD and its applications, see [30] and [22].

2.3. Robust Formulations. The optimized DMD problem (2.2) is formulated using the least-squares error norm, which is equivalent to assuming a Gaussian model on the errors between predicted and observed data:

$$\mathbf{X} = \boldsymbol{\Phi}(\boldsymbol{\alpha})\mathbf{B} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim N(0,\sigma^2 I).$$

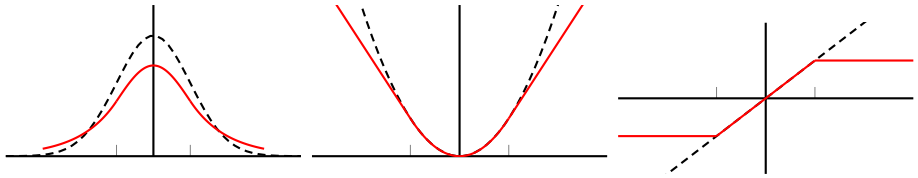

FIG. 1. *Gaussian (black dash) and Huber (red solid) Densities, Negative Log Likelihoods, and Influence Functions.*

This error model, and the corresponding formulation, are vulnerable to outliers in the data. Both DMD and optimized DMD are known to be sensitive to outliers, so in practice data are 'pre-cleaned' before applying these approaches.

In many domains, formulations based on robust statics have become the method of choice for dealing with contaminated data. Two common approaches are

- to replace the LS penalty with one that penalizes deviations less harshly and
- to solve an extended problem that explicitly identifies outliers while fitting the model.

The first approach, often called M-estimation [20, 24], is illustrated in Figure 1. Replacing the least squares penalty by the Huber penalty

$$\rho(z) = \begin{cases} \frac{1}{2}|z|^2 & \text{if } |z| \le \kappa \\ \kappa|z| - \frac{1}{2}\kappa^2 & \text{if } |r| > \kappa \end{cases}$$

corresponds to choosing the solid red penalty rather than the dotted black least squares penalty in the center panel of Figure 1. This corresponds to modeling errors $\boldsymbol{\epsilon}$ using the density $\exp(-\rho)$, which has heavier tails than the Gaussian (see left panel of Figure 1). Heavier tails means deviations (i.e. larger residuals) are more likely than under the Gaussian model, and so observations that deviate from the norm have less *influence*, i.e. effect on the fitted parameters $(\boldsymbol{\alpha}, \mathbf{B})$ than under the Gaussian model (see right panel of Figure 1). The M-estimator-DMD problem can be written as

$$\min_{\boldsymbol{\alpha},\mathbf{B}} \sum_{j=1}^{n} \rho(X_{\cdot j} - \boldsymbol{\Phi}(\boldsymbol{\alpha})\mathbf{B}_{\cdot j}) := \sum_{j=1}^{n} \rho_j(\boldsymbol{\alpha}, \mathbf{B}),$$

where the sum is run across columns, denoted $X_{\cdot j}$ and $\mathbf{B}_{\cdot j}$.

Another approach, called trimmed estimation, builds on M-estimation by coupling explicit outlier identification/removal with model fitting. The trimmed DMD formulation for any penalty ρ is given by

$$\min_{\boldsymbol{\alpha},\mathbf{B}} \sum_{l=1}^{h} \rho_{j_l}(\boldsymbol{\alpha}, \mathbf{B}), \tag{2.5}$$

where $\rho_{j_1}(\boldsymbol{\alpha}, \mathbf{B}) \le \cdots \le \rho_{j_h}(\boldsymbol{\alpha}, \mathbf{B})$ are the first h order statistics of the objective values and $\{j_1, \ldots, j_h\} \subseteq \{1, \ldots, n\}$. Interpreting the loss ρ_j as the negative log likelihood of the jth observed column, it is clear that trimming jointly fits a likelihood model while simultaneously eliminating the influence of all low-likelihood observations. An equivalent formulation to (2.5) replaces the order statistics with explicit weights

$$\min_{\boldsymbol{\alpha},\mathbf{B},\mathbf{w}} \sum_{j=1}^{n} w_j \rho_j(\boldsymbol{\alpha}, \mathbf{B}), \quad 0 \le w_j \le 1, \quad \mathbf{1}^\intercal \mathbf{w} = h. \tag{2.6}$$

The reader should verify that (2.6) and (2.5) are equivalent.

Trimmed M-estimators were initially introduced by [26] in the context of least-squares regression. The author's original motivation was to develop linear regression estimators that have a high breakdown point (in this case 50%) and good statistical efficiency (in this case $n^{-1/2}$)[2]. For a number of years, the difficulty of efficiently optimizing trimmed problems limited their application. However, recent work has made it possible to efficiently apply trimming to general models [33, 1]. We show how to incorporate trimming into the robust DMD framework below.

2.4. Regularization. Optimized DMD allows prior knowledge to be incorporated into the optimization formulation, either through constraints on variables, or regularization terms.

For example, in exponential fitting problems like the DMD, the real parts of the $\boldsymbol{\alpha}$ parameters affect the ability of the discovered model to forecast because they determine the exponential growth rate of $\boldsymbol{\Phi}(\boldsymbol{\alpha})$. A natural regularization is to place an upper bound on the real parts of $\boldsymbol{\alpha}$, i.e. to impose the constraint $\text{real}(\boldsymbol{\alpha}) \leq \gamma$ with γ chosen by the user.

We write the constraint as follows:

$$r(\boldsymbol{\alpha}) = \begin{cases} 0 & \text{if real}(\boldsymbol{\alpha}) \leq \gamma \\ \infty & \text{if real}(\boldsymbol{\alpha}) > \gamma. \end{cases} \tag{2.7}$$

This is a simple convex function but it is not smooth. Fortunately, there are simple iterative algorithms based on proximal operators which can handle such penalties.

DEFINITION 2.1 (Proximal Operator). *A proximal operator can be associated to any proper, lower semi-continuous, convex function defined on a Hilbert space $\mathcal{V}$. Let φ be such a function. Then,*

$$\text{prox}_{\varphi}(\mathbf{v}) = \arg\min_{\mathbf{x}\in\mathcal{V}} \left(\varphi(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|_2^2 \right).$$

While the evaluation of the proximal operator entails an optimization problem, there are many common and important penalties for which there is an explicit, easy-to-evaluate formula. The penalty r above admits a trivial proximal operator (see [10]): entry-wise projection of each component of $\boldsymbol{\alpha}$ onto the shifted left half-plane in $\mathbb{C}$. Because this operation is simple to compute, we call $r(\boldsymbol{\alpha})$ a "prox-friendly" regularizer. The VP framework proposed in this manuscript can incorporate both prox-friendly and smooth regularizers on the $\boldsymbol{\alpha}$ parameters; see subsection 3.3 for details.

Constraints and penalties can also be imposed on the matrix $\mathbf{B}$. We assume that only smooth, separable regularization penalties can be used; and in this case, the regularization is added to the function g.

2.5. Problem formulation. Let $q(\mathbf{B})$ and $r(\boldsymbol{\alpha})$ be convex regularization terms. We formulate the general robust DMD problem as follows:

$$\min_{\boldsymbol{\alpha},\mathbf{B},\mathbf{w}} f(\boldsymbol{\alpha},\mathbf{B},\mathbf{w}) := g(\boldsymbol{\alpha},\mathbf{B},\mathbf{w}) + r(\boldsymbol{\alpha}) + s(\mathbf{w}) \,, \tag{2.8}$$

[2]Breakdown refers to the percentage of outlying points which can be added to a dataset before the resulting M-estimator can change in an unbounded way.

where $r(\boldsymbol{\alpha})$ encodes optional regularization functions (or constraints) for $\boldsymbol{\alpha}$ (see Section 2.4) and

$$g(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w}) = \sum_{j=1}^{n} w_j \rho\left(X_{\cdot j} - \boldsymbol{\Phi}(\boldsymbol{\alpha})\mathbf{B}_{\cdot j}\right) + q(\mathbf{B}_{\cdot j}) \tag{2.9}$$

with ρ any differentiable penalty, $q(\mathbf{B}_{\cdot j})$ any regularizer for columns of $\mathbf{B}$, and $s(\mathbf{w})$ encoding the capped simplex constraints:

$$s(\mathbf{w}) = \begin{cases} 0 & \text{if } 0 \le w_j \le 1,\ \mathbf{1}^\intercal \mathbf{w} = h \\ \infty & \text{else.} \end{cases} \tag{2.10}$$

These constraints are explained in Section 2.3. The $\mathbf{w}$ variables select the best-fit h columns of the data, and only use those values to update $\boldsymbol{\alpha}$. Since each $w_j \in [0,1]$ rather than $\{0,1\}$, the solutions do not have to be integral. However, for any fixed $(\mathbf{B}, \boldsymbol{\alpha})$ there exists a vertex solution, since the subproblem in $\mathbf{w}$ with the other variables fixed is a linear program. The function $s(\mathbf{w})$ admits a simple proximal operator, which is the projection onto the intersection of the h-simplex with the unit cube[3]. Setting $h = n$ forces $w_j = 1$ for each column, eliminating trimming completely, and reducing (2.8) to a simpler regularized M-estimation form of DMD.

Our numerical examples use constraints for $\boldsymbol{\alpha}$, but do not regularize $\mathbf{B}$, that is, $q(\mathbf{B}_{\cdot j}) \equiv 0$. However, we consider separable penalties q in the algorithmic description to preserve the generality of (2.8).

Remark 2.2. Observe that (2.8) captures the standard optimized DMD, where $\rho(\cdot) = \|\cdot\|_F^2$, $q(\mathbf{B}_{\cdot j}) \equiv 0$, $r(\boldsymbol{\alpha}) \equiv 0$ and $h = n$.

3. Methods. In this section, we develop numerical approaches for (2.8). While (2.8) is in principle a non-linear and non-convex problem in $nk + k + n$ variables, the variable projection framework decouples this into relatively simple convex optimizations over nk of these variables, the 'inner' problem, and a non-linear, non-convex value function optimization problem in the remaining $n + k$ variables, the 'outer' problem. We detail this in subsection 3.1, including sufficient conditions on the regularizers and penalties in the robust formulation. In subsection 3.2, we provide some explicit formulas for the gradients of the relevant objective functions. Suitable algorithms for both smooth and non-smooth regularizers are presented in subsection 3.3. For large n and m, the simple, convex optimizations which arise can become a computational bottleneck. In subsection 3.4, we develop a stochastic variance reduction (SVRG) algorithm that accelerates each iteration for the 'outer' problem by exploiting the structure of the 'inner' problem, enabling scalability of the approach to large problems. Specifically, only a subset of the 'inner' problems need to be evaluated at each iteration to get noisy gradients of the outer problem. Numerical studies illustrating the utility of both robust DMD formulations, and of the SVRG acceleration, are presented later on in Section 4.

3.1. Variable projection. To compute the robust optimized DMD, we apply the variable projection (VP) technique to the optimization problem (2.8). Define the reduced function $\tilde{f}$ and implicit solution $\mathbf{B}(\boldsymbol{\alpha})$ by

$$\begin{aligned} \tilde{f}(\boldsymbol{\alpha}, \mathbf{w}) &= \min_{\mathbf{B}} f(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w}) \ , \\ \mathbf{B}(\boldsymbol{\alpha}, \mathbf{w}) &= \arg\min_{\mathbf{B}} f(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w}) \ , \end{aligned} \tag{3.1}$$

[3] This set is called the *capped simplex*, and admits fast projections [1].

where f is as defined in (2.8).

We refer to partially minimizing f over $\mathbf{B} \in \mathbb{C}^{k\times n}$ as the *inner problem* and minimizing $\tilde{f}$ over $\boldsymbol{\alpha} \in \mathbb{C}^k$ and $\mathbf{w} \in \mathbb{R}^n$ as the *outer problem*. We leave the trimming parameters $\mathbf{w}$ as part of the outer problem to leave the inner problem as both smooth and convex, making it easier to develop provably convergent variable projection algorithms and their stochastic extensions. The general VP strategy is to apply an iterative method to the outer problem, computing a (local) minimizer of the reduced function $\tilde{f}$. In each such iteration, we must solve the inner problem over $\mathbf{B}$. When f is convex and smooth with respect to $\mathbf{B}$, fast optimization algorithms can be applied to the inner problem. Moreover, the inner problem is embarrassingly parallelizable, as will be clear in the next subsection.

For the outer problem, we require gradient information for $\tilde{f}$ with respect to $\boldsymbol{\alpha}$ and $\mathbf{w}$. The gradient formula (2.4) holds for a very broad problem class. For example, as long as f is strongly convex with respect to $\mathbf{B}$, the result holds for any convex regularizer on $\mathbf{B}$ [31]. If the regularizer is finite-valued, the strong convexity of f with respect to $\mathbf{B}$ is not necessary, and we have an alternative set of conditions [25, Theorem 10.58]:

1. $g(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w})$ is level-bounded in $\mathbf{B}$ locally uniformly in $\boldsymbol{\alpha}$; i.e., for any compact subset of $\boldsymbol{\alpha}$, the union of sublevel sets $\{\mathbf{B} : g(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w}) \leq \gamma\}$ is bounded.
2. The gradient of $g(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w})$ exists and is continuous for all $(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w})$.
3. $\mathbf{B}(\boldsymbol{\alpha}, \mathbf{w})$ is unique.

Several assumptions on g, $\boldsymbol{\Phi}$, and q (see (2.9)) can be made to ensure these conditions hold. E.g.:

- If g is differentiable, convex, and has compact level sets with respect to $\mathbf{B}$, and $\boldsymbol{\Phi}(\boldsymbol{\alpha})$ has full rank, then the result holds.
- For any convex g, strong convexity of q also ensures the result without any assumptions on $\boldsymbol{\Phi}(\boldsymbol{\alpha})$.

The gradient formula (2.4) is valid for all of the examples in the paper and takes the form:

$$\nabla \tilde{f}(\boldsymbol{\alpha}, \mathbf{w}) = \partial_{\boldsymbol{\alpha},\mathbf{w}} f(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w})|_{\boldsymbol{\alpha},\mathbf{B}(\boldsymbol{\alpha},\mathbf{w}),\mathbf{w}} \,. \tag{3.2}$$

See subsection 3.2 for more explicit gradient formulas.

Solving (2.8) requires optimization procedures for both the inner and outer problems. We outline some deterministic algorithms in the subsection 3.3 and then present a a stochastic variance reduction algorithm in subsection 3.4.

3.2. Gradient formulas. In order to apply the algorithms proposed in this manuscript, we need to compute the gradient of the penalty function (2.9) with respect to $\boldsymbol{\alpha}$, $\mathbf{B}$, and $\mathbf{w}$.

In all of the optimization methods, we treat the real and imaginary components of α_j and B_{ji} as independent, real-valued parameters. However, for the sake of compactness, we write derivative formulas in the Wirtinger sense, computing partial derivatives with respect to the complex variables. Consider a complex number $z = x + \mathbf{i}y$. The derivatives for the real components can be recovered from the formulas

$$\frac{\partial}{\partial z} = \frac{1}{2}\left(\frac{\partial}{\partial x} - \mathbf{i}\frac{\partial}{\partial y}\right). \tag{3.3}$$

DEFINITION 3.1 (Wirtinger derivative). *Let $\psi(z)$ be a function of z which can be written as $\psi(z) = \Psi(z, \bar{z})$ where Ψ is differentiable with respect to both z and $\bar{z}$. The*

Wirtinger derivative of ψ is then the partial derivative of Ψ with respect to z, treating $\bar{z}$ as a constant.

For example, the Huber penalty may be written as

$$\rho(z) = P(z, \bar{z}; \kappa) = \begin{cases} \kappa\sqrt{z\bar{z}} - \frac{1}{2}\kappa^2, & |z| \geq \kappa \\ \frac{1}{2} z\bar{z}, & |z| < \kappa \end{cases} .$$

The Wirtinger derivative of the Huber penalty is then

$$\rho'(z) = \frac{\partial}{\partial z} P(z, \bar{z}; \kappa) = \begin{cases} \dfrac{\kappa\bar{z}}{2\sqrt{z\bar{z}}}, & |z| < \kappa \\ \frac{1}{2}\bar{z}, & |z| \geq \kappa \end{cases} .$$

Once the derivative of ρ is known, then the gradients of g with respect to $\boldsymbol{\alpha}$, $\mathbf{w}$, and $\mathbf{B}$ can then be computed using the chain rule. Gradients of the reduced function $\tilde{f}$ can then be obtained via (3.2). For notational convenience, we define a matrix-valued penalty function

$$\boldsymbol{\rho}(A) := \begin{bmatrix} \rho(A_{1,1}) & \cdots & \rho(A_{1,n}) \\ \vdots & \ddots & \vdots \\ \rho(A_{m,1}) & \cdots & \rho(A_{m,n}) \end{bmatrix} .$$

In this notation, we can write

$$g(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w}) = \mathbf{1}^\intercal \boldsymbol{\rho}(\mathbf{X} - \boldsymbol{\Phi}(\boldsymbol{\alpha})\mathbf{B})\mathbf{w} + q(\mathbf{B}) , \tag{3.4}$$

where

$$q(\mathbf{B}) = \sum_{j=1}^{n} q(\mathbf{B}_{\cdot j}) .$$

We then have:

$$\begin{aligned} \nabla_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w}) &= -\operatorname{diag}\left[\mathbf{B}\operatorname{Diag}(\mathbf{w})\boldsymbol{\rho}'(\mathbf{X} - \boldsymbol{\Phi}\mathbf{B})^\intercal (\operatorname{Diag}(\mathbf{t})\boldsymbol{\Phi})\right] \\ \nabla_{\mathbf{B}} g(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w}) &= -\boldsymbol{\Phi}^\intercal \boldsymbol{\rho}'(\mathbf{X} - \boldsymbol{\Phi}\mathbf{B})\operatorname{Diag}(\mathbf{w}) + \nabla q(\mathbf{B}) \\ \nabla_{\mathbf{B}_{\cdot j}} g(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w}) &= -\boldsymbol{\Phi}^\intercal \boldsymbol{\rho}'(\mathbf{X}_{\cdot j} - \boldsymbol{\Phi}\mathbf{B}_{\cdot j}) w_j + \nabla q(\mathbf{B}_{\cdot j}) \\ \nabla_{\mathbf{w}} g(\boldsymbol{\alpha}, \mathbf{B}, \mathbf{w}) &= \boldsymbol{\rho}(\mathbf{X} - \boldsymbol{\Phi}\mathbf{B})^\intercal \mathbf{1}, \end{aligned} \tag{3.5}$$

where we define

$$\operatorname{diag}(A) := \begin{bmatrix} a_{11} \\ \vdots \\ \\ \vdots \\ a_{nn} \end{bmatrix} , \qquad \operatorname{Diag}(a) := \begin{bmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_n \end{bmatrix} .$$

3.3. Deterministic algorithms. For the algorithms below, we assume that g in (2.8) is convex with respect to $\mathbf{B}$; recall that g is continuously differentiable with respect to $\mathbf{B}$, $\boldsymbol{\alpha}$, and $\mathbf{w}$. We note that the function f may not necessarily be smooth, depending on the regularizer $r(\boldsymbol{\alpha})$.

Observe that the inner problem decouples into n independent subproblems of dimension m:

$$\mathbf{B}_{\cdot j}(\boldsymbol{\alpha}, \mathbf{w}) = \arg\min_{\mathbf{b}} \quad w_j \rho\left(X_{\cdot j} - \boldsymbol{\Phi}(\boldsymbol{\alpha})\mathbf{b}\right) + q(\mathbf{b}), \quad j = 1, \ldots, n. \tag{3.6}$$

We use BFGS to solve each of these subproblems, since the dimension of each problem is relatively small and BFGS gives a superlinear convergence rate while using only gradient information. Further, the values of the vectors $\mathbf{b}_j$ are independent of each other so that these solves can be performed in parallel.

The selection of the outer solver depends on the regularizers. When r in (2.8) is continuously differentiable, we can also use BFGS as our outer solver, resulting in Algorithm 3.1. When r is non-smooth but admits an efficient prox operator, a first order method such as the proximal gradient method or its accelerations, such as FISTA [5], can be used instead; see Algorithm 3.2 for a simple prox-gradient implementation. Proximal gradient requires a rule for selecting a step size, $\eta_{\boldsymbol{\alpha}}$. We use a backtracking line search in practice but other methods are available.

Algorithm 3.1 VP using BFGS for outer problem (smooth r).

Input: $\boldsymbol{\alpha}^0$, $\mathbf{B}^0$, $\mathbf{w}^0$, $H^0_{\boldsymbol{\alpha}} = I$, $\nu = 0$.

1: **while** not converged **do**
2: **for** $j = 1, \ldots, n$ **do**
3: $\mathbf{B}^{\nu+1}_{\cdot j} \leftarrow \arg\min_{\mathbf{b}} w_j^\nu \rho\left(X_{\cdot j} - \boldsymbol{\Phi}(\boldsymbol{\alpha}^\nu)\mathbf{b}\right) + q(\mathbf{b})$
4: $\mathbf{w}^{\nu+1} \leftarrow$ weights update
5: $f^\nu_{\boldsymbol{\alpha}} \leftarrow f(\boldsymbol{\alpha}^\nu, \mathbf{B}^{\nu+1}, \mathbf{w}^{\nu+1})$
6: $g^\nu_{\boldsymbol{\alpha}} \leftarrow \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}^\nu, \mathbf{B}^{\nu+1}, \mathbf{w}^{\nu+1})$
7: **if** $\nu \geq 1$ **then**
8: $s^\nu \leftarrow f^\nu_{\boldsymbol{\alpha}} - f^{\nu-1}_{\boldsymbol{\alpha}}$
9: $y^\nu \leftarrow g^\nu_{\boldsymbol{\alpha}} - g^{\nu-1}_{\boldsymbol{\alpha}}$
10: $\beta^\nu \leftarrow (\langle s^\nu, y^\nu \rangle)^{-1}$
11: $H^\nu_{\boldsymbol{\alpha}} \leftarrow \left[I - \beta^\nu (s^\nu)(y^\nu)^\intercal\right] H^{\nu-1} \left[I - \beta^\nu (y^\nu)(s^\nu)^\intercal\right] + \beta (s^\nu)(s^\nu)^\intercal$
12: $\boldsymbol{\alpha}^{\nu+1} \leftarrow \text{LineSearch}(\boldsymbol{\alpha}^\nu - \eta_{\boldsymbol{\alpha}} H^\nu_{\boldsymbol{\alpha}} g^\nu_{\boldsymbol{\alpha}})$
13: $\nu \leftarrow \nu + 1$

Output: $\boldsymbol{\alpha}^\nu$, $\mathbf{B}^\nu$.

Algorithm 3.2 VP using proximal gradient for outer problem (prox-friendly r).

Input: $\boldsymbol{\alpha}^0$, $\mathbf{B}^0$, $\mathbf{w}^0$, $\nu = 0$.

1: **while** not converged **do**
2: **for** $j = 1, \ldots, n$ **do**
3: $\mathbf{B}^{\nu+1}_{\cdot j} \leftarrow \arg\min_{\mathbf{b}} w_j^\nu \rho\left(X_{\cdot j} - \boldsymbol{\Phi}(\boldsymbol{\alpha}^\nu)\mathbf{b}\right) + q(\mathbf{b})$
4: $\mathbf{w}^{\nu+1} \leftarrow$ weights update
5: $\boldsymbol{\alpha}^{\nu+1} \leftarrow \text{prox}_{\eta_{\boldsymbol{\alpha}} r}\left(\boldsymbol{\alpha}^\nu - \eta_{\boldsymbol{\alpha}} \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}^\nu, \mathbf{B}^{\nu+1}, \mathbf{w}^{\nu+1})\right)$
6: $\nu \leftarrow \nu + 1$

Output: $\boldsymbol{\alpha}^\nu$, $\mathbf{B}^\nu$.

There are different ways to update the weights $\mathbf{w}$, see line 4 in Algorithms 3.1 and 3.2. We let ν denote the iteration counter. Define

$$\rho_j^\nu = \rho\left(X_{\cdot j} - \boldsymbol{\Phi}(\boldsymbol{\alpha}^\nu)\mathbf{b}_j^{\nu+1}\right).$$

The objective with respect to $\mathbf{w}$ is given by

$$\min_{\mathbf{w}} \sum_{j=1}^{n} w_j \rho_j^{\nu} + s(\mathbf{w}),$$

where s encodes the weight constraints (2.10). The simplest update rule is to set $w_j = 1$ if ρ_j^{ν} is one of the h smallest, and 0 otherwise [33]; this corresponds to partial minimization in $\mathbf{w}$ at every step. A less aggressive strategy is to use proximal updates on $\mathbf{w}$,

$$\mathbf{w}^{\nu+1} = \mathrm{prox}_{\eta_{\mathbf{w}} s}\left(\mathbf{w}^{\nu} - \eta_{\mathbf{w}} \nabla_{\mathbf{w}} f(\boldsymbol{\alpha}^{\nu}, \mathbf{B}^{\nu+1}, \mathbf{w}^{\nu})\right)$$

where any step size $\eta_{\mathbf{w}} > 0$ can be used [1]. We use the former simple rule as the default in the algorithm. When $h = n$, trimming is turned off, and all weights are identically equal to 1.

Algorithm 3.3 SVRG for DMD

Input: $\boldsymbol{\alpha}^0$, $\mathbf{B}^0$, $\mathbf{w}^0$

Initialize $\nu = 0$, $\zeta_j = \nabla f_j\left(\boldsymbol{\alpha}^0, \mathbf{w}^0\right)$ for $j = 1, 2, \ldots, n$, and $\zeta = \frac{1}{n}\sum_{j=1}^{n} \zeta_j$
while not converged **do**
  Uniformly sample $I^{\nu} \subset \{1, 2, \ldots, n\}$, such that $|I^{\nu}| = \tau$
  Sample $J^{\nu} \in \{0, 1\}$, such that $P(J = 1) \ll P(J = 0)$.
  for $j \in I_{\nu}$ **do**
    $\mathbf{B}_{\cdot j}^{\nu+1} \leftarrow \arg\min_{\mathbf{b}} w_j^{\nu} \rho(X_{\cdot,j} - \boldsymbol{\Phi}(\boldsymbol{\alpha}^{\nu})\mathbf{b}) + q(\mathbf{b})$
    $\zeta_j^{+} \leftarrow \nabla \tilde{g}_j\left(\boldsymbol{\alpha}^{\nu}, \mathbf{w}\right)$
  if $J = 1$ **then**
    $\mathbf{w}^{\nu+1} \leftarrow$ weights update
  else
    $\mathbf{w}^{\nu+1} \leftarrow \mathbf{w}^{\nu}$
  $\boldsymbol{\alpha}^{\nu+1} \leftarrow \mathrm{prox}_{\eta_{\boldsymbol{\alpha}} r}\left(\boldsymbol{\alpha}^{\nu} - \eta_{\boldsymbol{\alpha}}\left[\frac{1}{\tau}\sum_{j \in I^{\nu}}\left(\zeta_j^{+} - \zeta_j\right) + \zeta\right]\right)$
  $\eta_{\boldsymbol{\alpha}} \leftarrow$ step size update
  $\zeta_j \leftarrow \zeta_j^{+}$ for $j \in I^{\nu}$
  $\zeta \leftarrow \frac{1}{n}\sum_{j=1}^{n} \zeta_j$
  $\nu \leftarrow \nu + 1$

Output: $\boldsymbol{\alpha}^{\nu}$, $\mathbf{B}^{\nu}$

3.4. A scalable stochastic algorithm. In DMD applications, n represents the number of spatial variables, and is often much larger than the dimension of the outer problem, k. In step 2 of Algorithms 3.1 and 3.2, we must solve n subproblems of dimension k for which gradient evaluations have $O(mk)$ cost (see (3.5)). For large n and m, this is a computational bottleneck.

We use stochastic methods to scale the approach. The basic idea is to partially minimize over a random sample of τ columns of $\mathbf{B}$, with $\tau \ll n$; the resulting (scaled) gradient is an unbiased estimate of $\nabla_{\boldsymbol{\alpha}} \tilde{f}$. More precisely, define

$$\mathbf{B}_{\cdot j}(\boldsymbol{\alpha}, \mathbf{w}) = \arg\min_{\mathbf{b}} w_j \rho\left(X_{\cdot j} - \boldsymbol{\Phi}(\boldsymbol{\alpha})\mathbf{b}\right) + q(\mathbf{b}),$$
$$\tilde{g}_j(\boldsymbol{\alpha}, \mathbf{w}) = w_j \rho(X_{\cdot j} - \boldsymbol{\Phi}(\boldsymbol{\alpha})\mathbf{B}_{\cdot j}(\boldsymbol{\alpha}, \mathbf{w})) + q(\mathbf{B}_{\cdot j}(\boldsymbol{\alpha}, \mathbf{w})).$$

Then we have

$$\tilde{f}(\boldsymbol{\alpha}, \mathbf{w}) = \sum_{j=1}^{n} \tilde{g}_j(\boldsymbol{\alpha}, \mathbf{w}) + r(\boldsymbol{\alpha}) + s(\mathbf{w}).$$

This is a classical setting for stochastic methods. In each iteration, we can use a subset of $\tilde{g}_j$ to calculate the approximate gradient for the smooth part of $\tilde{f}$ in order to reduce the computational burden. Here we use SVRG [21] as our stochastic solver for the outer problem; the full details are given in Algorithm 3.3.

SVRG is chosen in contrast with stochastic proximal gradient (SPG). Stochastic proximal gradient (SPG) has no convergence theory, though it is frequently used in practice. A clear practical downside of using SPG is that it requires a diminishing step size and its performance is sensitive to parameters that guide step size selection. For SVRG, we may use a constant step size. Convergence of SVRG is analyzed for the nonconvex case, with and without trimming, by [1]. The trimming weights $\mathbf{w}$ require full passes through the data, and this is why the $\mathbf{w}$ update (lines 8-11 of Algorithm 3.3) is done rarely. A numerical study showing the impact of SVRG compared to full gradient methods is summarized in Figure 6 of Section 4.

Remark 3.2. This stochastic approach is an alternative to using a dimensionality reduction based on projecting onto SVD modes [4] or using an optimized but fixed subsampling of the columns [18]. With the method of Algorithm 3.3, none of the data is discarded or filtered by the cost reduction procedure.

4. Synthetic examples. In this section, we demonstrate the effectiveness of robust penalties in handling outliers on a pair of synthetic test cases with known solution. These examples are drawn from the additive noise study of [12] and represent two cases in which additive noise presents a challenge: recovering purely oscillatory dynamics and recovering a decaying mode in a system with a growing mode. In [4], the optimized DMD was demonstrated to improve significantly over the biases of the exact DMD for the case of additive Gaussian noise. Here, we show that the robust DMD can also handle significant outliers. We also demonstrate the effectiveness of the SVRG-based randomized algorithm, Algorithm 3.3, by comparing its performance on a medium-sized problem with the performance of the proximal-gradient-based algorithm, i.e. Algorithm 3.2, and the performance of SPG.

4.1. A simple periodic example. Let $\mathbf{x}(t)$ be the solution of a two dimensional linear system with the following dynamics

$$\dot{\mathbf{x}} = \begin{pmatrix} 1 & -2 \\ 1 & -1 \end{pmatrix} \mathbf{x} \,. \tag{4.1}$$

We use the initial condition $\mathbf{x}(0) = (1, 0.1)^\intercal$ and take snapshots

$$\mathbf{x}_j = \mathbf{x}(j\Delta t) + \sigma \mathbf{g}_j + \mu \mathbf{s}_j \,,$$

where $\Delta t = 0.1$, σ and μ are prescribed noise levels, $\mathbf{g}_j$ is a vector whose entries are drawn from a standard normal distribution, and $\mathbf{s}_j$ is a vector whose entries are the product of a Bernoulli trial with small expectation p and a standard normal (corresponding to sparse noise). The snapshots are therefore corrupted with a base level of noise σ and sparse "spikes" of size μ with firing rate p. A sample time series for this example can be found in Figure 2.

The $k = 2$ eigenvalues of the system matrix in (4.1) are $\pm\mathbf{i}$, corresponding to sinusoidal dynamics in time. In Figure 3, we plot the median (over 200 random trials) of the l^1-norm error in the approximations of these eigenvalues using three different methods: the exact DMD of [30]; the optimized DMD as defined in (2.2);

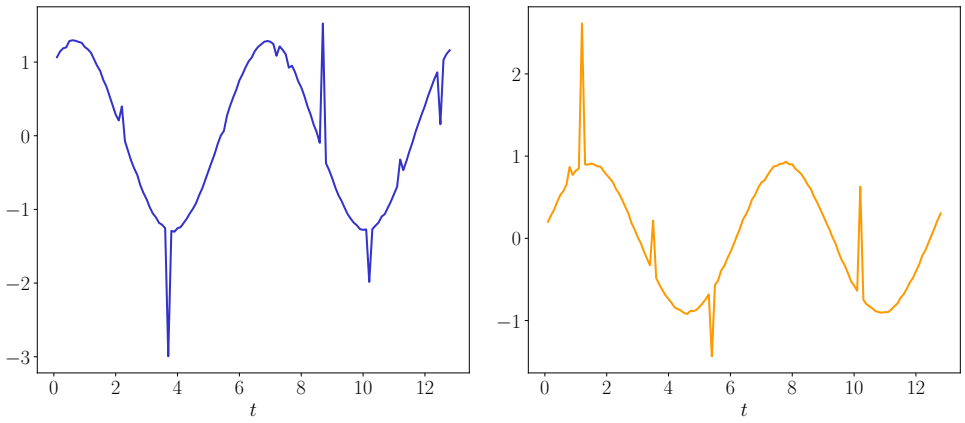

FIG. 2. *Sample time series of $x_1(t)$ and $x_2(t)$ for the simple periodic example, with background noise of size $\sigma = 10^{-2}$ and spikes of size $\mu = 1$ added at $p = 5\%$ of the snapshots for each channel.*

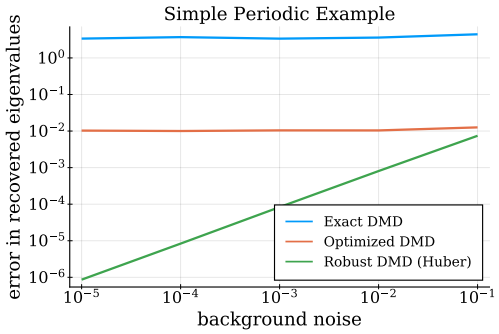

FIG. 3. *Median error in the computed eigenvalues over 200 runs. The background noise σ varies while the size of the spikes is fixed at $\mu = 1$ and the firing rate is fixed at $p = 5\%$.*

and the robust DMD as defined in (2.8), with ρ the Huber norm and $h = n = 2$ (no trimming). Each trial consists of the first 128 snapshots with additive noise. We bound the maximum exponential growth rate by setting $\gamma = 1$ in the regularizer $r(\boldsymbol{\alpha})$ (see (2.7)). The level of the background noise, σ, varies over the experiments and the size and firing rate of the spikes are fixed at $\mu = 1$ and $p = 5\%$, respectively. We set the Huber parameter using knowledge of the problem set-up, i.e. $\kappa = 5\sigma$, but in a real data setting this parameter would have to be estimated or chosen adaptively. While the optimized DMD improves over the exact DMD, the error does not decrease as the level of the background noise decreases. We therefore see the effect of the sparse outliers using the optimized DMD. For the robust formulation, on the other hand, the accuracy of the eigenvalues is determined by the level of the background noise, so that the outliers are not biasing the computed eigenvalues.

4.2. An example with hidden dynamics. In the case that a signal contains some rapidly decaying components it can be more difficult to identify the dynamics, particularly in the presence of sensor noise [12]. We consider a signal composed of two sinusoidal forms which are translating, with one growing and one decaying, i.e.

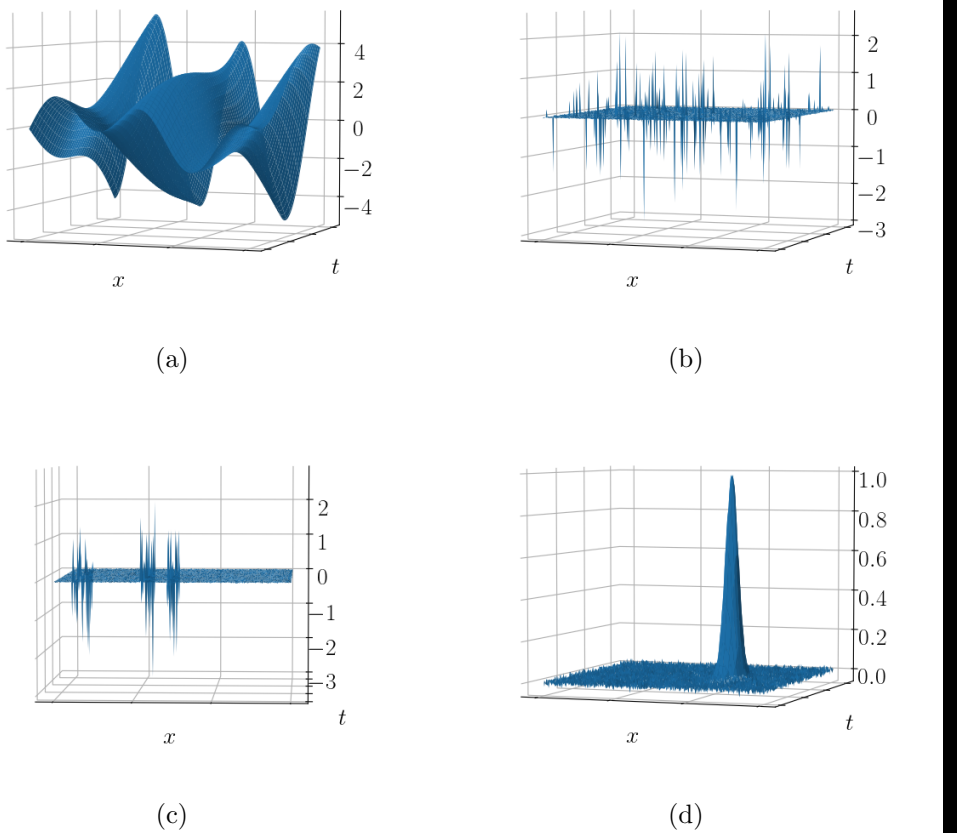

FIG. 4. *A surface plot of the data for the hidden dynamics example and surface plots of a sample of each type of noise we consider.*

$$x(y,t) = \sin(k_1 y - \omega_1 t)e^{\gamma_1 t} + \sin(k_2 y - \omega_2 t)e^{\gamma_2 t}\,, \tag{4.2}$$

where $k_1 = 1$, $\omega_1 = 1$, $\gamma_1 = 1$, $k_2 = 0.4$, $\omega_2 = 3.7$, and $\gamma_2 = -0.2$ (following settings used by [12]). This signal has $k = 4$ continuous time eigenvalues given by $\gamma_1 \pm i\omega_1$ and $\gamma_2 \pm i\omega_2$. We set the domain of y to be $[0, 15]$ and use 300 equispaced points, y_j, to discretize. For the time domain, we set $\Delta t = \pi/(2^8 - 2)$ so that the number of snapshots we use, $m = 2^7$, covers $[0, \pi/2]$. We denote the vector of discrete values $x(y_j, t)$ by $\mathbf{x}(t)$. See Figure 4(a) for a surface plot of this data.

We consider three different types of perturbations of the data. The first perturbation adds background noise and spikes, as in the previous example, i.e. the snapshots are given by

$$\mathbf{x}_j^{(1)} = \mathbf{x}(j\Delta t) + \sigma \mathbf{g}_j + \mu \mathbf{s}_j\,,$$

where σ and μ are prescribed noise levels, $\mathbf{g}_j$ is a vector whose entries are drawn from

a standard normal distribution, and $\mathbf{s}_j$ is a vector whose entries are the product of a Bernoulli trial with small expectation p and a standard normal. See Figure 4(b) for a sample plot of this "sparse noise" pattern. The second perturbation we consider adds background noise and spikes which are confined to specific entries of $\mathbf{x}_j$, i.e.

$$\mathbf{x}_j^{(2)} = \mathbf{x}(j\Delta t) + \sigma \mathbf{g}_j + \mu \tilde{\mathbf{s}}_j \,,$$

where $\mathbf{g}_j$, σ, and μ are as above and the $\tilde{\mathbf{s}}_j$ are sparse vectors which have the same sparsity pattern for all j and nonzero entries drawn from a standard normal distribution (this corresponds to having a few broken sensors recording the data). We plot a sample of this "broken sensor" noise pattern in Figure 4(c). The third perturbation we consider adds background noise and a localized bump to the data, i.e.

$$\left[\mathbf{x}_j^{(3)}\right]_i = x(y_i, j\Delta t) + \sigma\mathcal{N}(0,1) + A\exp\left(-\left(\frac{y_b - y_i}{w\Delta y}\right)^2 - \left(\frac{t_b - j\Delta t}{w\Delta t}\right)^2\right) ,$$

where σ is as above, $\mathcal{N}(0,1)$ denotes a number drawn from the standard normal distribution, A determines the maximum height of the bump, w determines the "width" of the bump, and y_b and t_b determine the center of the bump in space and time (this corresponds to having some non-exponential dynamics in the data). In Figure 4(d), we plot a sample of this "bump" noise pattern.

In Figure 5, we plot the median (over 20 random trials) of the l^1-norm error in the approximations of the eigenvalues using four different methods: the exact DMD of [30]; the optimized DMD as defined in (2.2); the robust DMD as defined in (2.8), with ρ the Huber norm and $h = n = 300$ (no trimming); and the robust DMD with ρ the standard Frobenius norm and $h = 0.8n = 240$ (trimming). Each trial consists of the first 128 snapshots with additive noise. We bound the maximum exponential growth rate by setting $\gamma = 2$ in the regularizer $r(\boldsymbol{\alpha})$ (see (2.7)). The level of the background noise, σ, varies over the experiments. For the "sparse noise" and "broken sensor" snapshots, the size of the spikes is fixed at $\mu = 1$ and the density is fixed at $p = 5\%$, i.e. 5% of the entries are corrupted for the "sparse noise" example and 5% of the sensors are corrupted for the "broken sensor" example. For the "bump" snapshots, the height of the bump is fixed at $A = 1$ and the width at $w = 10$. We set the Huber parameter using knowledge of the problem set-up, i.e. $\kappa = 5\sigma$, but in a real data setting this parameter would have to be estimated or chosen adaptively.

With sparse noise, as in Figure 5(a), the results for the exact DMD, optimized DMD, and Huber norm-based robust DMD are consistent with the simple periodic example. The Huber norm formulation is the only one which is able to take advantage of the lower levels of background noise. The trimming formulation provides very little advantage for this example, as any sensor can be affected by the outliers. In contrast, we see that the trimming formulation is able to out-perform the Huber formulation for the broken sensor example (see Figure 5(b)), as the algorithm is able to adaptively remove the broken sensors from the data. In Figure 5(c), we plot the results for the bump data, which display some interesting behavior. Here, the optimized DMD performs worse than it did for the other noise sources, perhaps due to an attempted fit of the smooth bump. For all but the highest background noise level, the Huber and trimming formulations show a significant advantage over the optimized DMD and exact DMD, with the trimming formulation performing the best. The trimming

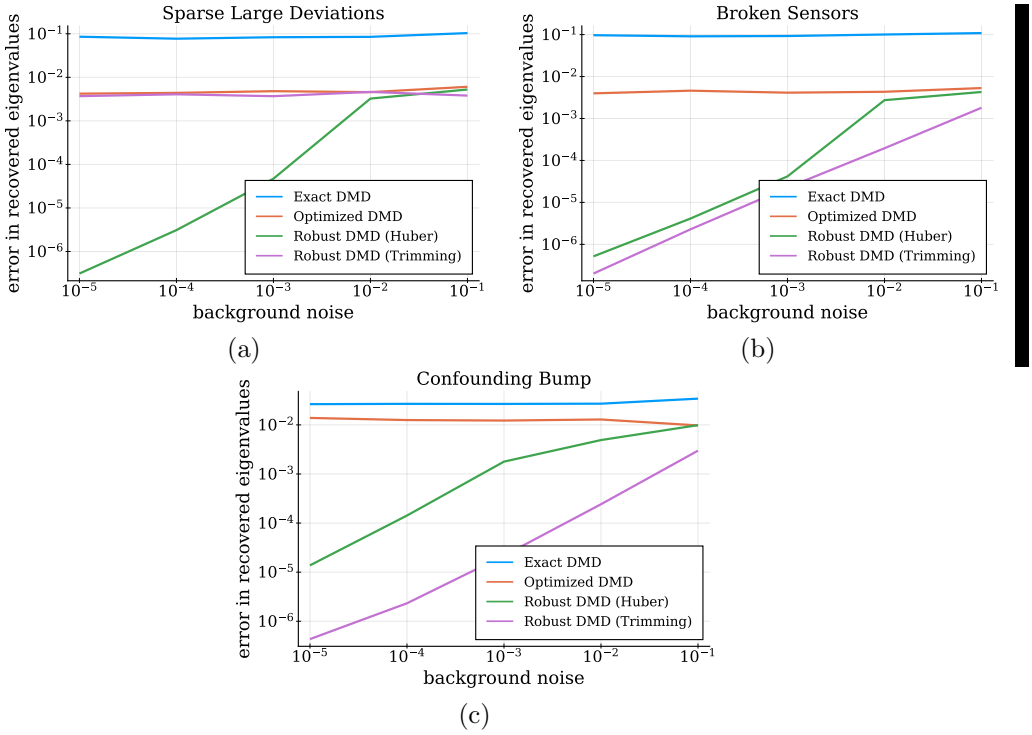

FIG. 5. *Median error in the computed eigenvalues over 20 runs. The background noise σ varies while the size of the spikes is fixed at $\mu = 1$ and the firing rate is fixed at $p = 5\%$ for the "sparse noise" and "broken sensor" examples and the height is fixed at $A = 1$ and the width at $w = 10$ for the "bump" example.*

formulation therefore presents an attractive solution for data with unknown, localized deviations from the exponential basis of the DMD, especially given that the inner problem for trimming with the Frobenius penalty can be solved rapidly. Of course, trimming can be combined with a Huber (or other robust) penalty for increased robustness to outliers.

4.3. Scalability demonstration. As noted in subsection 3.4, the inner problem becomes a computational bottleneck for large dimensional problems (large n and m). Algorithm 3.3 proposes an acceleration where noisy gradient values are obtained for the outer problem by only solving a fraction of the inner subproblems at each step.

In Figure 6, we solve a synthetic problem with dimension $m = 512$, $n = 1000$, and $k = 3$ using 3 different methods: proximal gradient (PG) with backtracking line search, as in Algorithm 3.2; stochastic proximal gradient (SPG); and the proposed SVRG algorithm. As noted above, SPG requires a diminishing step size; we choose the attenuation schedule

$$\eta_{\boldsymbol{\alpha}}^{\nu} = \frac{\eta_{\boldsymbol{\alpha}}^{0}}{\text{floor}(\nu/K) + 1},$$

with $\eta_{\boldsymbol{\alpha}}^{0} = 10^{-7}$ and $K = 100$. For SVRG, we use a constant step size $\eta_{\boldsymbol{\alpha}}^{0}$, same as our choice for SPG, and set the parameter $\tau = 10$. Comparing the algorithms by

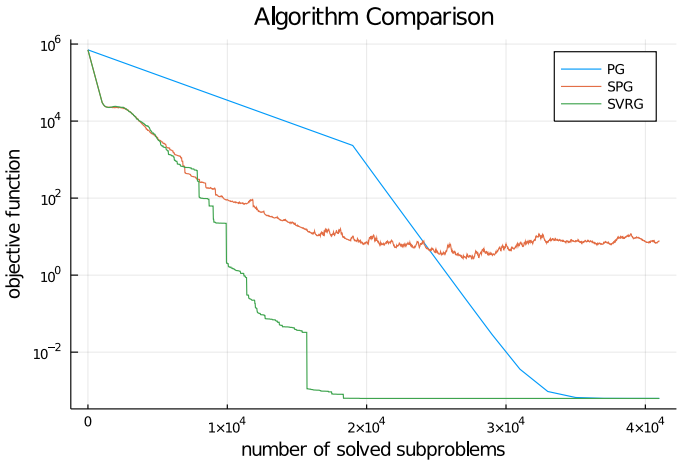

FIG. 6. *Comparative performance of SVRG, Stochastic Proximal Gradient (SPG) method and Proximal Gradient (PG) method using the same data set. PG is a robust method but requires many subproblem solves at each iteration (when using a line search). Both SPG and SVRG are much more efficient at the start, but SPG can easily stagnate; SVRG has the best theoretical rates and empirical performance with respect to the requisite number of subproblem solves.*

the total number of inner subproblems solved (number of optimizations to compute $\mathbf{b}_j(\boldsymbol{\alpha}, \mathbf{w})$ for some j) we see that SVRG is the most efficient method and is less noisy than SPG (Figure 6).

5. Conclusion and future directions. We have presented an optimization framework and a suite of numerical algorithms for computing the dynamic mode decomposition with robust penalties and parameter constraints. This framework allows for improved performance of the DMD in a number of settings. In the presence of sparse noise or non-exponential structure, the use of robust penalties significantly decreases the bias in the computed eigenvalues. When using the DMD to perform future state prediction, adding the constraint that the eigenvalues lie in the left half-plane increases the stability of the extrapolation. The algorithms presented are capable of solving small to medium-sized problems in seconds on a laptop (e.g. the problem of size $m = 512$, $n = 1000$, and $k = 3$ of subsection 4.3 takes a few seconds on a laptop) and scale well to higher-dimensional problems due to their intrinsic parallelism and the efficiency of the SVRG approach. In contrast with previous approaches, the SVRG increases efficiency without throwing out data or incidentally filtering it. For DMD practitioners, these features of the new framework and algorithms presented here will enable the analysis of larger, noisier, and more complex data sets than previously possible. The software used for these calculations is available in the open-source package `RobustDMD`[4].

The present work can be extended in a number of ways. Because the inner solve completely decouples over the columns of $\mathbf{X}$ and $\mathbf{B}$, the algorithms presented above immediately generalize to data-sets with missing entries and even data which are collected asynchronously across sensors. While the global nature of an optimized DMD fit has advantages in terms of the quality of the recovered eigenvalues, it implicitly rules out process noise. However, including process noise or a known forcing term

[4] https://github.com/UW-AMO/RobustDMD.jl

would be useful in many applications. Incorporating such terms into this optimization framework is ongoing work and results will be reported at a later date. We also note that much of the above applies to dimensionality reduction using any parameterized family of time dynamics, not just exponentials. For such an application, many of the algorithms above could be easily adapted, so long as gradient formulas are available.

REFERENCES

[1] A. Aravkin and D. Davis, *Trimmed statistical estimation via variance reduction*, Mathematics of Operations Research, 45 (2020), pp. 292–322.

[2] A. Y. Aravkin, D. Drusvyatskiy, and T. van Leeuwen, *Efficient quadratic penalization through the partial minimization technique*, IEEE Transactions on Automatic Control, (2017).

[3] A. Y. Aravkin and T. Van Leeuwen, *Estimating nuisance parameters in inverse problems*, Inverse Problems, 28 (2012), p. 115016.

[4] T. Askham and J. N. Kutz, *Variable projection methods for an optimized dynamic mode decomposition*, SIAM Journal on Applied Dynamical Systems, 17 (2018), pp. 380–416.

[5] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM journal on imaging sciences, 2 (2009), pp. 183–202.

[6] B. M. Bell and J. V. Burke, *Algorithmic differentiation of implicit functions and optimal values*, Advances in Automatic Differentiation, (2008), pp. 67–77.

[7] B. W. Brunton, L. A. Johnson, J. G. Ojemann, and J. N. Kutz, *Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition*, Journal of Neuroscience Methods, 258 (2016), pp. 1–15.

[8] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust principal component analysis?*, Journal of the ACM (JACM), 58 (2011), p. 11.

[9] K. K. Chen, J. H. Tu, and C. W. Rowley, *Variants of dynamic mode decomposition: boundary condition, koopman, and fourier analyses*, Journal of nonlinear science, 22 (2012), pp. 887–915.

[10] P. L. Combettes and J.-C. Pesquet, *Proximal splitting methods in signal processing*, in Fixed-point algorithms for inverse problems in science and engineering, Springer, 2011, pp. 185–212.

[11] J. P. Cunningham and Z. Ghahramani, *Linear dimensionality reduction: survey, insights, and generalizations.*, Journal of Machine Learning Research, 16 (2015), pp. 2859–2900.

[12] S. T. Dawson, M. S. Hemati, M. O. Williams, and C. W. Rowley, *Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition*, Experiments in Fluids, 57 (2016), pp. 1–19.

[13] C. Dicle, H. Mansour, D. Tian, M. Benosman, and A. Vetro, *Robust low rank dynamic mode decomposition for compressed domain crowd and traffic flow analysis*, in Multimedia and Expo (ICME), 2016 IEEE International Conference on, IEEE, 2016, pp. 1–6.

[14] N. B. Erichson, S. L. Brunton, and J. N. Kutz, *Compressed dynamic mode decomposition for real-time object detection*, Preprint, (2015).

[15] G. Golub and V. Pereyra, *Separable nonlinear least squares: the variable projection method and its applications*, Inverse problems, 19 (2003), p. R1.

[16] G. H. Golub and R. J. LeVeque, *Extensions and uses of the variable projection algorithm for solving nonlinear least squares problems*, in Proceedings of the 1979 Army Numerical Analsysis and Computers Conference, 1979.

[17] J. Grosek and J. N. Kutz, *Dynamic Mode Decomposition for Real-Time Background/Foreground Separation in Video*, arXiv preprint, arXiv:1404.7592, (2014).

[18] F. Guéniat, L. Mathelin, and L. R. Pastur, *A dynamic mode decomposition approach for large and arbitrarily sampled systems*, Physics of Fluids, 27 (2015), p. 025113, https://doi.org/10.1063/1.4908073, http://dx.doi.org/10.1063/1.4908073, https://arxiv.org/abs/http://dx.doi.org/10.1063/1.4908073.

[19] M. S. Hemati, C. W. Rowley, E. A. Deem, and L. N. Cattafesta, *De-biasing the dynamic mode decomposition for applied koopman spectral analysis of noisy datasets*, Theoretical and Computational Fluid Dynamics, (2017), pp. 1–20.

[20] P. J. Huber, *Robust statistics*, in International Encyclopedia of Statistical Science, Springer, 2011, pp. 1248–1251.

[21] R. Johnson and T. Zhang, *Accelerating stochastic gradient descent using predictive variance*

reduction, in Advances in neural information processing systems, 2013, pp. 315–323.
[22] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*, SIAM, 2016.
[23] N. Locantore, J. Marron, D. Simpson, N. Tripoli, J. Zhang, K. Cohen, G. Boente, R. Fraiman, B. Brumback, C. Croux, et al., *Robust principal component analysis for functional data*, Test, 8 (1999), pp. 1–73.
[24] R. Maronna, R. D. Martin, and V. Yohai, *Robust statistics*, John Wiley & Sons, Chichester. ISBN, 2006.
[25] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*, vol. 317, Springer Science & Business Media, 2009.
[26] P. J. Rousseeuw, *Multivariate estimation with high breakdown point*, Mathematical statistics and applications, 8 (1985), pp. 283–297.
[27] P. J. Schmid, *Dynamic mode decomposition of numerical and experimental data*, Journal of fluid mechanics, 656 (2010), pp. 5–28.
[28] P. J. Schmid and J. Sesterhenn, *Dynamic mode decomposition of numerical and experimental data*, in 61st Annual Meeting of the APS Division of Fluid Dynamics, American Physical Society, Nov. 2008.
[29] N. Takeishi, Y. Kawahara, Y. Tabei, and T. Yairi, *Bayesian dynamic mode decomposition*, in Proc. of the 26th Intl Joint Conf. on Artificial Intelligence (IJCAI), 2017, pp. 2814–2821.
[30] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, *On dynamic mode decomposition: Theory and applications*, Journal of Computational Dynamics, 1 (2014), pp. 391–421.
[31] T. van Leeuwen and A. Aravkin, *Nonsmooth variable projection*, SIAM Journal of Scientific Computing (to appear), (2021).
[32] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, *Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization*, in Advances in neural information processing systems, 2009, pp. 2080–2088.
[33] E. Yang, A. C. Lozano, and A. Aravkin, *A general family of trimmed estimators for robust high-dimensional data analysis*, Electronic Journal of Statistics, 12 (2018), pp. 3519–3553.