Proceedings of Machine Learning Research vol 145:721–756, 2021 2nd Annual Conference on Mathematical and Scientific Machine Learning

Practical and Fast Momentum-Based Power Methods

Tahseen Rabbani TRABBANI@CS.UMD.EDU
Department of Computer Science
University of Maryland, College Park, MD

Apollo Jain APOLLO.JAIN@STRERESEARCH.COM
Systems and Technology Research
Sensors Division, Arlington, VA

Arjun Rajkumar RAJKUMAR@UMD.EDU
Department of Computer Science
University of Maryland, College Park, MD

Furong Huang FURONGH@CS.UMD.EDU
Department of Computer Science
University of Maryland, College Park, MD

Editors: Joan Bruna, Jan S Hesthaven, Lenka Zdeborova

Abstract

The power method is a classical algorithm with broad applications in machine learning tasks, including streaming PCA, spectral clustering, and low-rank matrix approximation. The distilled purpose of the vanilla power method is to determine the largest eigenvalue (in absolute modulus) and its eigenvector of a matrix. A momentum-based scheme can be used to accelerate the power method, but achieving an optimal convergence rate with existing algorithms critically relies on additional spectral information that is unavailable at run-time, and sub-optimal initializations can result in divergence. In this paper, we provide a pair of novel momentum-based power methods, which we call the *delayed momentum power method* (DMPower) and a streaming variant, the *delayed momentum streaming method* (DMStream). Our methods leverage inexact deflation and are capable of achieving near-optimal convergence with far less restrictive hyperparameter requirements. We provide convergence analyses for both algorithms through the lens of perturbation theory. Further, we experimentally demonstrate that DMPower routinely outperforms the vanilla power method and that both algorithms match the convergence speed of an oracle running existing accelerated methods with perfect spectral knowledge.

Keywords: matrix decomposition, PCA, power methods, momentum acceleration, streaming PCA

1. Introduction

Approximating the dominant eigenvector of a matrix $A \in \mathbb{R}^{d\times d}$ is a task common to many statistical and industrial applications. The vanilla power method is a simple and inexpensive algorithm for computing the dominant eigenvector v_1 of a matrix A. For an initial $q_0 \in \mathbb{R}^d$ non-orthogonal to v_1, the power method performs the following update eventually converging to v_1,

$$q_k = A^k q_{k-1}/||A^k q_{k-1}||. \tag{1}$$

Owing to its ease of implementation and modest assumptions for convergence, the power method has found its use in a variety of machine learning tasks. It can be used to assist the k-means algorithm

for class separation of large datasets, which is referred to as power iteration clustering (PIC) (Lin and Cohen, 2010a,b; Thang et al., 2013). It is also used in sparse PCA, which projects data onto sparse principal components, that is, components with small ℓ_0 norm (Journée et al., 2010; Yuan and Zhang, 2013). In particular, word-embedding matrices for NLP models can be dimensionally-reduced via sparse PCA (Gawalt et al., 2010; Drikvandi and Lawal, 2020).

De Sa et al. (De Sa et al., 2018) introduced the *power method with momentum*, abbreviated as Power+M along with a stochastic variant, Mini-Batch Power+M. Both of these methods outperform their vanilla counterparts (the stochastic version of equation 1 uses instead an unbiased estimate $\widehat{A}$ of A). Here, speed is measured in the sense of iteration complexity, i.e., the number of outer loop iterations/updates required to output a vector q_k with precision ϵ, i.e., $\sin^2\theta(q_k, v_1) \triangleq 1 - (q_k^\top v_1)^2 < \epsilon$. Recently, other algorithms have been developed which adopt a momentum-based scheme (Kim and Klabjan, 2020; Mai and Johansson, 2019). However, a notable drawback to Power+M, Mini-Batch Power+M, and other existing momentum-based methods is that achieving accelerated iteration complexity requires knowledge of λ_2, the second greatest eigenvalue of A, which is an impractical assumption. Specifically, the momentum coefficient β, a hyperparameter of momentum-based power methods, must be set near $\lambda_2^2/4$ to achieve improved iteration complexity over the vanilla power method.

In this work, we develop a scheme which enjoys a near-optimal acceleration without the strict spectral knowledge requirements of other momentum-based methods. Our scheme consists of two phases. In the *pre-momentum phase*, we run a vanilla (or stochastic for the online setting) power method to approximate the dominant eigenvector and an inexact Hotelling deflation (Saad, 2011) to estimate λ_2 and later assign $\beta \approx \lambda_2^2/4$ (our momentum coefficient). In the *momentum phase*, we run Power+M (or Mini-Batch Power+M for the online setting) with the near-optimal β assignment taken from the previous phase for the remaining iterations until convergence. Our main contributions, the *delayed momentum power method* (DMPower) and the *delayed momentum streaming power method* (DMStream), are realizations of this scheme.

Relaxing Spectral Knowledge As explained above, an optimal acceleration of momentum-based methods relies on the proper selection of β. Previous approaches rely on expensive guess-and-check auto-tuning, whereby the user chooses β, and at each round conducts many experimental iterations with 0.67β, 0.99β, β, 1.01β, etc., revising the coefficient after determining which adjustment results in the largest Rayleigh quotient (De Sa et al., 2018). To the best of our knowledge, DMPower and DMStream are the first momentum-based algorithms to approximate optimal β in a partially-adaptive manner while still benefiting from acceleration. We present an informal version of our major results.

Theorem 1 (Informal) *Let $\Delta = |\lambda_1 - \lambda_2|$ denote the absolute difference between the largest and second-largest eigenvalues. With high probability, our proposed practical* DMPower*, after an efficient pre-momentum warm-up stage, outputs an ϵ-close estimate of the leading eigenvector within the state-of-the-art $\mathcal{O}\left(\frac{1}{\sqrt{\Delta}}\log\left(\frac{1}{\epsilon}\right)\right)$ iteration complexity using a momentum acceleration, without requiring knowledge of λ_2 or hyperparameter selection for λ_2.* DMPower *is extended to* DMStream *in the streaming setting, with similar iteration complexity.*

In section 4 we provide the full version the above theorem along with a companion streaming theorem. Although the momentum phases utilize existing methods, neither of our algorithms are true hybrids; DMPower and DMStream are the first of their kind to utilize inexact Hotelling deflation

for second eigenvalue recovery with guarantees. We will show that the iteration complexity of the DMPower and DMStream momentum phases respectively match the iteration complexity of Power+M and Mini-Batch Power+M without having to assign β at initialization.

Our experiments show that DMPower converges faster than the vanilla power method to various precisions of estimation and for matrices with tighter eigengaps, $\Delta = 0.01, \Delta = 0.001$, our DMPower matches the convergence speed of Power+M running with optimal β. DMPower also outperforms than the vanilla power method and closely mimics the performance of optimal Power+M when employed in the unsupervised learning task of spectral clustering. Additionally, in comparison to another concurrent iteration method, the simultaneous power iteration, our experiments demonstrate that DMPower determines a more accurate approximation of λ_2. DMStream outputs a more accurate estimation of the dominant eigenvector for a variety of batch sizes when compared against Oja's algorithm and performs nearly identically to Mini-Batch Power+M initialized with optimal β.

Summary of Contributions (1) Our proposed algorithms DMPower and DMStream achieve close to optimal performance when compared against other momentum-based power methods, which have been initialized with a priori unknowable optimal hyperparameters. (2) To the best of our knowledge, we are to first to provide a convergence analysis of an inexact deflation receiving approximate dominant eigenvectors supplied by a power iteration, both in the deterministic setting (Lemma 12) and streaming setting (Proposition 21). (3) Our Proposition 7 provides a guarantee for acceleration if one can provide lower bounds on $|\lambda_1 - \lambda_2|$ and $|\lambda_2 - \lambda_3|$, which is a far more practical requirement at run-time. (4) For many estimation precision requirements, DMPower outperforms the state-of-the-art Lanczos algorithm in iteration complexity when factoring in the recommended number d of tri-diagonalization iterations needed for numerical stability of the Lanczos algorithm. Additionally, DMPower runs noticeably faster (in seconds) than the Lanczos algorithm.

2. Related Works

Speedy Deterministic Power Methods Variations of the vanilla power method intended to improve its iteration complexity of $\mathcal{O}(\frac{1}{\lambda_1-\lambda_2}\log\frac{1}{\epsilon})$ have been suggested. The Lanczos algorithm (Golub and Van Loan, 2012) itself may be thought of as a fast variation of the power method, which has iteration complexity $\mathcal{O}(\frac{1}{\sqrt{\lambda_1-\lambda_2}}\log\frac{1}{\epsilon})$ but only after a tri-diagonalization process which is expensive in high dimensions. Lei et al. (Lei et al., 2016) consider a coordinate-wise update with complexity $\mathcal{O}\big(\frac{\lambda_1}{\lambda_1-\lambda_2}\log\frac{\tan\theta_0}{\epsilon}\big)$. Based on the heavy ball method first studied by Polyak (Polyak, 1964), De Sa et al. (De Sa et al., 2018) first proposed the addition of a momentum term to accelerate the basic power method. The term is controlled by a momentum coefficient, β, which is a hyperparameter selected at initialization. Their method achieves $\mathcal{O}\big(\frac{1}{\sqrt{\lambda_1-\lambda_2}}\log\frac{1}{\epsilon}\big)$, but only with a precise selection of β requiring unrealistic spectral knowledge at run-time. Mai and Johannson (Mai and Johansson, 2019) created a momentum-based algorithm `NAPI` for solving the canonical correlation analysis (CCA) problem similar requiring impractical hyperparameter selection. In contrast, our proposed DMPower requires no spectral knowledge and achieves the same speed as existing momentum methods running optimally.

Block Iterations To enjoy the effects of acceleration, DMPower concurrently runs a second power iteration for a finite number of rounds and extracts an optimal momentum coefficient. This approach is reminiscent, though not the same as block power iterations such as the simultaneous power iteration for matrices (Trefethen and Bau III, 1997), extended to tensors by Wang and Lu

(Wang and Lu, 2017), which intends to recover multiple eigenvectors at once. The simultaneous power method for matrices is well-known to suffer from rounding errors (Golub and Van Loan, 2012)(Börm and Mehl, 2012). In contrast, our inexact deflation is experimentally shown to achieve greater accuracy in extracting the second eigenvalue, which is critically important when eigengaps are tight. The improved "practical" simultaneous iteration and QR algorithm (Börm and Mehl, 2012) compute a QR factorization followed by a reversed RQ computation, which requires $O(d^2)$ more flops than inexact deflation. Furthermore, these block iterations do not employ any acceleration schemes: the convergence rate of an ϵ-close approximation of v_1 in a 2-vector simultaneous iteration is $\mathcal{O}\big(\frac{1}{\min\{\lambda_1-\lambda_2,\lambda_2-\lambda_3\}}\log\frac{1}{\epsilon}\big)$.

Inexact Power Methods In the pre-momentum phase, DMPower relies on an inexact deflation step every round. This step may be regarded as a Hardt-Price noisy power method, which has been studied in differential privacy-preserving PCA (Hardt and Roth, 2013) (Kapralov and Talwar, 2013), but primarily as a meta-algorithm. The DMPower may be regarded then as one practical application of the noisy power method, with an added momentum phase and thus requiring novel analysis beyond the scope of existing noisy power method literature. Furthermore, most existing results on deflation schemes operate under the assumption that eigenvectors and eigenvalues have been exactly recovered, but in practice, only approximate eigenvectors are used. To the best of our knowledge, we are the first to provide a guarantee on obtaining a second eigenvalue through deflation using inaccurate dominant eigenvectors (Lemma 12 and Proposition 21). Our algorithm is the first in the class of momentum-based power methods to incorporate an acceleration scheme without the requirement of guessing an approximation of the second eigenvalue at runtime.

Streaming Methods DMStream is a streaming companion to DMPower, which instead uses unbiased estimates of the underlying covariance matrix A for all of its updates. The streaming setting assumes that A is either inaccessible or expensive to obtain, and several algorithms have been developed to address this situation (Mitliagkas et al., 2013; Shamir, 2015; Kim and Klabjan, 2020; Jain et al., 2016). The momentum phase of DMStream is built upon Mini-Batch Power+M (De Sa et al., 2018), whose iteration complexity is $\mathcal{O}\Big(\frac{1}{\sqrt{\lambda_1-\lambda_2}}\log\big(\frac{1}{\epsilon}\big)\Big)$, which is desirable in that it matches the offline state-of-the-art Lanczos complexity, but similar to Power+M, requires a momentum hyperparameter β close to $\lambda_2^2/4$. Just like DMPower, DMStream efficiently approximates λ_2 in its pre-momentum phase and then drops into a momentum phase where it enjoys the aforementioned optimal offline iteration complexity.

Gradient Descent Methods From the optimization perspective, leading eigenvector computation is equivalent to minimizing $-\mathbf{x}^\top A\mathbf{x}$ where $\mathbf{x}\in S^2$, i.e., the unit sphere. Although this problem is geodesically non-convex, it is possible to use gradient descent to solve this problem (Absil et al., 2009; Wen and Yin, 2013; Pitaval et al., 2015).In particular, the global convergence rate has been shown to be $\mathcal{O}\Big(\big(\frac{\lambda_1}{\lambda_1-\lambda_2}\big)^2\log\frac{1}{\epsilon}\Big)$, which is generally incomparable to the guarantee of DMPower (Xu et al., 2018). Conjugate gradient (CG) methods have also been employed to compute the dominant eigenvector. The Fletcher-Reeves Gradient Descent (FRGD) (Wang and Ye, 2020), a fully-adaptive momentum-based CG method, achieves a convergence rate of $\mathcal{O}\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)$, where $\kappa=\lambda_1/\lambda_n$ and thus this rate is also not directly comparable to the convergence rate of DMPower and other classical power methods, since they largely depend on the first eigengap.

3. Accelerated Momentum-Based Power Methods via Inexact Deflation

Problem Setup We first outline the setting and assumptions typical of deterministic momentum-based PCA. We discuss the streaming setting in section 3.2. Let $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ be data points. Our goal is to recover the top eigenvector and dominant eigenvalue of the symmetric PSD covariance matrix $A = \frac{1}{n}\sum_{i=1}^n x_i x_i^\top \in \mathbb{R}^{d\times d}$. We assume that A has eigenvalues $1 \geq \lambda_1 > \lambda_2 > \lambda_3 \geq \lambda_4 \geq \cdots \geq \lambda_d \geq 0$ with associated orthonormal eigenvectors $v_1, v_2, \ldots, v_d$. Unless noted otherwise, $\|\cdot\|$ refers to the 2-norm for vectors and matrices. We let $\Delta_{1,2} := \lambda_1 - \lambda_2$ and $\Delta_{2,3} := \lambda_2 - \lambda_3$. The vanilla power method, at each round $k = 1, 2, \ldots$ performs the following update,

$$q_k = \frac{A^k q_{k-1}}{\|A^k q_{k-1}\|} \tag{2}$$

$$\nu_k = q_k^\top A q_k \tag{3}$$

where q_0 is a random unit vector non-orthogonal to v_1. Under these conditions, $q_k \to v_1$ and $\nu_k \to \lambda_1$ at a geometric convergence rate, with ratio $\left(\frac{\lambda_2}{\lambda_1}\right)^2$. Here, error is measured as the sine squared of the angle $\theta(q_k, v_1)$ between our unit q_k and v_1, that is, $\sin^2\theta(q_k, v_1) \triangleq 1 - (q_k^\top v_1)^2$. The power method with momentum uses the alternative update,

$$q_k = \frac{Aq_{k-1} - \beta q_{k-2}}{\|Aq_{k-1} - \beta q_{k-2}\|} \tag{4}$$

where $q_{-1} = \mathbf{0}$ and β is the *momentum coefficient* chosen by the user at initialization. Sa et al. (De Sa et al., 2018) establish the following theorem and its corollary,

Theorem 2 (Convergence of Power+M (De Sa et al., 2018)) *Given a PSD matrix $A \in \mathbb{R}^{d\times d}$ with eigenvalues $1 \geq \lambda_1 > \lambda_2 \geq \lambda_3 \ldots \lambda_d \geq 0$, running with $\lambda_2 < 2\sqrt{\beta} \leq \lambda_1$ results in q_k with*

$$\sin^2\theta(q_k, v_1) = 1 - (q_k^\top v_1)^2 \leq \frac{4}{|q_0^\top v_1|^2}\left(\frac{2\sqrt{\beta}}{\lambda_1 + \sqrt{\lambda_1^2 - 4\beta}}\right)^{2k} \tag{5}$$

Corollary 3 ((De Sa et al., 2018)) *For $\epsilon \in (0,1)$ after $T = \mathcal{O}\left(\frac{\sqrt{\beta}}{\sqrt{\lambda_1^2 - 4\beta}}\log\frac{1}{\epsilon}\right)$ iterations,* $\sin^2\theta(q_T, v_1) \leq \epsilon$.

We have from these results that $\beta \in [\lambda_2^2/4, \lambda_1^2/4)$, and minimizing $\frac{\sqrt{\beta}}{\sqrt{\lambda_1 - 4\beta}}$ as in Corollary 3 over this interval results in an optimal assignment $\beta = \lambda_2^2/4$.

Practical Considerations The power method with momentum is capable of achieving a faster convergence rate than the vanilla power iteration. For $\beta = \lambda_2^2/4$, we have, as in Theorem 2, a geometric convergence with ratio $\left(\frac{\lambda_2}{\lambda_1 + \sqrt{\lambda_1^2 - \lambda_2^2}}\right)^2$, which is smaller and thus faster than the vanilla convergence ratio $\left(\frac{\lambda_2}{\lambda_1}\right)^2$. However, therein lies the impracticality of the algorithm: the user will generally not have knowledge of λ_2. Guessing the momentum coefficient $2\sqrt{\beta} > \lambda_1$ can result in extremely slow convergence and in some cases, divergence, as shown in Figure 1. In fact, there is currently no known convergence guarantee for $2\sqrt{\beta} > \lambda_1$; we notice that in the setting of Theorem 2 and its associated Corollary, that such selection of β results in an imaginary ratio, rendering the guarantee uninformative. For practical use, we should remove β as a hyperparameter.

Ultimately then, we must approximate λ_2. This naturally leads us to consider a concurrent/block iteration scheme that synchronously converges towards v_1 and v_2. From our approximations of v_2, we may obtain Rayleigh quotient estimations of λ_2. Accuracy is critical when the eigengap $\Delta_{1,2}$ is small, and since the simultaneous power iteration experiences rounding errors as shown in Figure 2, it is not appropriate for use. The more accurate *practical simultaneous power iteration* (Börm and Mehl, 2012) employs a thin QR factorization at each step, which is expensive. Our instinct, then, is to adopt a concurrent iteration which utilizes *matrix deflation* to approximate λ_2.

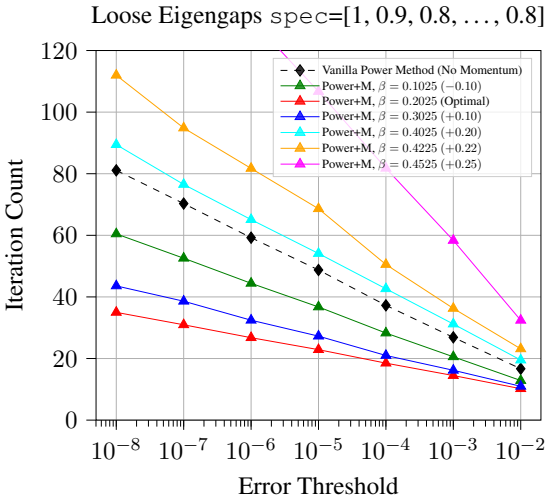


Figure 1: **Sub-optimal β Selection for Power+M**. We measure convergence speeds at various momentum assignments. The X-axis is the error threshold ϵ between approximates q_k, q_{k-1} of v_1 needed for Power+M to terminate, i.e., we run until $||q_k - q_{k-1}|| < \epsilon$. The Y-axis is the total number of iterations k needed to meet this condition according to the update equation (4). Averaged over 1000 runs, each time run on a random PSD $A \in \mathbb{R}^{10\times 10}$ with spectrum $\lambda_1 = 1, \lambda_2 = 0.9, \lambda_3 = 0.8$, and remaining eigenvalues set to 0.8. We observe that increased deviation from the optimal β assignment results in worsened and eventually divergent performance.

Approach for Smart Selection of β Deflation methods extract further eigenvalues along the spectrum (ordered in absolute modulus), once previous eigenvalues and eigenvectors are determined. Hotelling deflation (Golub and Van Loan, 2012) is one such scheme upon which we model our algorithms. Assume for our symmetric PSD A that it also has a positive second eigengap, i.e., $\lambda_2 > \lambda_3$. If we form the deflation matrix $B = A - \lambda_1 v_1 v_1^\top$, then for w_0 non-orthogonal to v_2, we have that $w_k = \frac{Bw_{k-1}}{||Bw_{k-1}||} \to v_2$ and $q_k^\top B q_k \to \lambda_2$ as $k \to \infty$. Clearly, we do not have access to λ_1 and v_1 (their approximation is the entire purpose of PCA), but at each round of a vanilla power iteration, we *do* have approximations ν_k and q_k as in update equations (2) and (3), so we may instead form an inexact deflation matrix $A - \nu_k q_k q_k^\top$.

One might wonder the implications of setting β as $q_J^\top A q_J$. Setting $\beta = q_J^\top A q_J$ will result in a momentum parameter converging towards λ_1, which would not lie in the convergence zone $[\lambda_2^2/4, \lambda_1^2/4)$ of Theorem 2. The next logical idea then would be to set $\beta = (q_J^\top A q_J)^2/4$, which would converge towards $\lambda_1^2/4$, but employing this coefficient very nearly follows the same dynamics as the vanilla power method (see Theorem 2) and hence, no improvement.

Our first proposed algorithm the *delayed momentum power method* (DMPower) is a realization of the above discussion and considerations, using inexact deflation to progressively approximate λ_2. Experimentally, DMPower outperforms the vanilla power method for all specified error thresholds, running near-optimally at tighter eigengaps $\Delta_{1,2} = 0.01$ and $\Delta_{1,2} = 0.001$. DMPower experiences decayed noise at each step, making it possible to establish a convergence guarantee, which we present in Theorem 4.

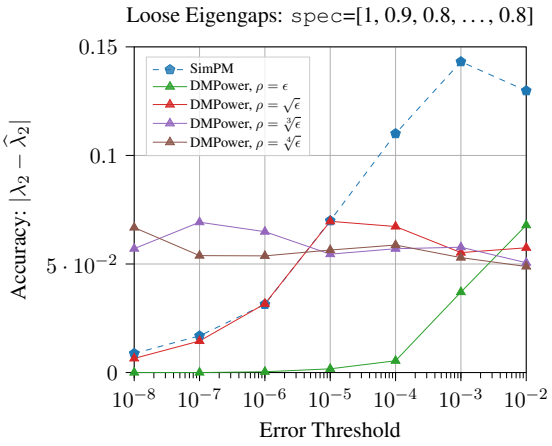


Figure 2: **Accuracy vs Simultaneous Power**. We measure the Rayleigh quotient accuracy in determining λ_2 between the Simultaneous Power Iteration (SimPM) and DMPower. The X-axis is the error threshold ϵ of the entire DMPower run. We report the true accuracy of $\widehat{\lambda}_2$ output by the pre-momentum phase, which is reflected in the Y-axis. Averaged over 1000 runs, each time run on a random PSD $A \in \mathbb{R}^{10\times 10}$. We varied ρ according to ϵ for convenience, but ρ does not depend on ϵ in implementation. We observe that DMPower is overall the most accurate, while SimPM is inaccurate at low thresholds.

3.1. Delayed Momentum Power Method (DMPower)

Intuitions of DMPower The delayed momentum power method, our primary contribution, experiences momentum-based acceleration after an initial waiting period with no required selection of β at initialization. As stated before, we use inexact Hotelling deflation to approximate $\lambda_2^2/4$. It is known that inexact deflation can succeed with controlled noise (Kapralov and Talwar, 2013; Hardt and Roth, 2013), and we prove in Appendix B that the inexact deflation step in DMPower satisfies such conditions. We will eventually obtain an estimate $\beta \in [\lambda_2^2/4, \lambda_1^2/4)$. Using this well-behaved β, we can transition to a momentum-based update exhibiting acceleration.

Overview Algorithm 1 describes DMPower. It proceeds as a routine vanilla power method, but at each step, our approximate top eigenvector and eigenvalue q_k and ν_k are used to form an inexact deflation matrix $(A-P) = A-\nu_k q_k q_k^\top$, which is run in a power-iterative manner on an initial vector w_0, which is non-orthogonal to v_2. We refer to this portion of DMPower as the *pre-momentum phase*. For a practical implementations, the first for-loop would exit once $|\mu_{k+1} - \mu_k| \leq \rho$, for some user-specified ρ, i.e., once our λ_2 approximates are close to each other. We discuss at length the practical and theoretical considerations of selecting ρ in section 4.3.

After achieving ρ-accuracy between our approximates μ_k, we set $\beta \leftarrow \mu_k^2/4$, and $q_0 \leftarrow q_j$ as our new initial vector. We then proceed to the *momentum phase*, which runs Power+M updates until ϵ-accuracy is achieved among the q_k. Notice that q_j has already made progress towards v_1. This greatly benefits our Power+M updates according to Theorem 2, where $\sin^2\theta(q_t, v_1)$ is limited at each step by the constant $\frac{4}{|q_j^\top v_1|}$. We provide a convergence guarantee in Theorem 4 for DMPower.

Complexity The vanilla power iteration costs $\mathcal{O}(d^2)$ flops per round, and since we are concurrently running two vanilla power methods in addition to a Rayleigh quotient, we perform two additional $\mathcal{O}(d^2)$ flops. Although we are not asymptotically increasing the time complexity, we justify the increased flops: we prove in Theorem 4 that for a fixed ρ, we will achieve our desired approximation of λ_2 after a finite number of rounds. Furthermore, we are only interested in the Rayleigh quotient approximations μ_j of λ_2 from each deflation step, and it is well-known that if $\|v_2 - w_k\| = \rho$, then $|\lambda_2 - \mu_k| = \mathcal{O}(\rho^2)$ (Trefethen and Bau III, 1997). That is, the Rayleigh quotient is a quadratically-accurate estimate of v_2.

Algorithm 1 Delayed Momentum Power Method (DMPower)

Require: $A \in \mathbb{R}^{d\times d}$ symmetric, unit $q_0 \in \mathbb{R}^d$, pre-momentum phase iterations J, momentum phase iterations K, unit $w_0 \in \mathbb{R}^d$

for $j = 1, 2, \ldots, J$ **do**
  $q_j \leftarrow Aq_{j-1}$
  $q_j \leftarrow q_j / \|q_j\|$
  $\nu_j \leftarrow q_j^\top Aq_j$ ▷ Rayleigh Quotient estimate of λ_1
  $P \leftarrow \nu_j q_j q_j^\top$
  $w_j \leftarrow (A - P)w_{j-1}$ ▷ Inexact deflation
  $w_j \leftarrow w_j / \|w_j\|$
  $\mu_j \leftarrow w_j^\top Aw_j$ ▷ Rayleigh Quotient estimate of λ_2
$\widehat{\lambda}_2 = \mu_J$
$\beta \leftarrow \widehat{\lambda}_2^2/4$ ▷ Approximated optimal momentum coefficient
$q_1 \leftarrow q_J$ ▷ Current estimate of v_1
$q_0 \leftarrow \mathbf{0}$
for $k = 1, 2, \ldots, K$ **do** ▷ while $\|q_k - q_{k-1}\| > \epsilon$
  $q_{k+1} \leftarrow Aq_k - \beta q_{k-1}$ ▷ Momentum update
  $q_{k+1} \leftarrow q_{k+1} / \|q_{k+1}\|$
  $\nu_k \leftarrow q_{k+1}^\top Aq_{k+1}$
return q_K, ν_K

3.2. Delayed Momentum Streaming Power Method (DMStream)

Overview We first review a typical setting for streaming PCA. Assume we have a stream of $\mathbb{R}^d$ inputs $x_1, x_2, \ldots$ drawn from some unknown distribution $\mathcal{D}$ with underlying covariance matrix A. We wish to recover the dominant eigenvector v_1 of A.

Streaming algorithms have risen to address this challenge, which instead use a different unbiased estimate $\widehat{A}_t = \frac{1}{n}\sum_{i=1}^n x_i x_i^\top$ of A at each round t to conduct their updates, where n is a fixed batch size, and the x_i's are selected in a uniformly random manner (Shamir, 2015; Jain et al., 2016; Mitliagkas et al., 2013; De Sa et al., 2018). In general, the total iteration complexity and runtime is dependent on several factors, including the variance $\mathbb{E}[(\widehat{A}_t - A) \otimes (\widehat{A}_t - A)]$ of each unbiased estimate (where $\otimes$ denotes the Kronecker product) and the batch size. The sample complexity is the total number of streaming inputs need overall to output an ϵ-close estimate of v_1 for $\epsilon < 1$.

Our second algorithm which we call the *delayed momentum streaming power method* (DMStream) shown in Algorithm 2 is a streaming companion to DMPower. Theorem 16 due to De Sa et. al De Sa et al. (2018) provides a convergence guarantee on a momentum-based streaming algorithm referred to as Mini-Batch Power+M. In particular, significant acceleration is experienced if $2\sqrt{\beta} \in [\lambda_2, \lambda_1)$, but as in the case of Power+M, selection of such a β at run-time is an impractical ask. The full convergence guarantee of Mini-Batch Power+M is provided in Theorem 16. Similar to DMPower, DMStream approximates a converging momentum coefficient in a pre-momentum phase and then uses that coefficient to accelerate convergence in a secondary momentum phase.

DMStream superficially resembles DMPower, instead using unbiased estimates for its updates. However, due to the noise introduced by estimation error of A by $\widehat{A}_t$ in conjunction with the noise

Algorithm 2 Streaming Delayed Momentum Power Method (DMStream)

Require: Streaming samples $x_1, x_2, \ldots, x_l \in \mathbb{R}^d$, batch size n, unit $q_0 \in \mathbb{R}^d$, pre-momentum phase iterations J, momentum phase iterations K, unit $w_0 \in \mathbb{R}^d$

for $j = 1, 2, \ldots, J$ **do**
  Generate unbiased estimate $\widehat{A}_j = \frac{1}{n}\sum_{i=(j-1)n+1}^{jn} x_i x_i^\top$
  $q_j \leftarrow \widehat{A}_j q_{j-1}$
  $q_j \leftarrow q_j / \|q_j\|$
  $\nu_j \leftarrow q_j^\top \widehat{A}_j q_j$ ▷ Rayleigh Quotient estimate of λ_1
  $P \leftarrow \nu_j q_j q_j^\top$
  $w_j \leftarrow (\widehat{A}_j - P) w_{j-1}$ ▷ Inexact deflation
  $w_j \leftarrow w_j / \|w_j\|$
  $\mu_j \leftarrow w_j^\top \widehat{A}_j w_j$ ▷ Rayleigh Quotient estimate of λ_2
$\hat{\lambda}_2 \leftarrow \mu_J$
$\beta \leftarrow \mu_J^2/4$ ▷ Approximated optimal momentum coefficient
$q_1 \leftarrow q_J$ ▷ Current estimate of v_1
$q_0 \leftarrow \mathbf{0}$
for $k = 1, 2, \ldots, K$ **do** ▷ while $\|q_k - q_{k-1}\| > \epsilon$
  Generate unbiased estimate $\widehat{A}_k = \frac{1}{n}\sum_{i=(k-1)n+1}^{kn} x_i x_i^\top$
  $q_{k+1} \leftarrow \widehat{A}_k q_k - \beta q_{k-1}$ ▷ Momentum update
  $q_{k+1} \leftarrow q_{k+1} / \|q_{k+1}\|$
  $\nu_k \leftarrow q_{k+1}^\top \widehat{A}_k q_{k+1}$
return q_K, ν_K

introduced by our imperfect estimations of v_1 by q_t, the inexact deflation step is more challenging to analyze and results in a distinct guarantee, which we provide in Theorem 5.

Complexity Each matrix-vector multiplication cost $\mathcal{O}(d^2)$ with three such multiplications in every round of the pre-momentum phase (power iteration, Hotelling iteration, and an inexact Rayleigh quotient). Akin to DMPower, we justify these increased FLOPS by noting that the pre-momentum phase will terminate in a finite number of rounds, which is shown in Theorem 5. We empirically observe that even a rough selection of β provides us with noticeable acceleration, resulting in lower iteration complexity overall, and thus a decreased total runtime when compared to conventional streaming PCA as in Algorithm 4.

4. Convergence Analysis

We now provide our two major convergence theorems. We adopt the same notations as in Algorithm 1 and Algorithm 2. Both algorithms are divided into J steps of a pre-momentum phase and K steps of a momentum phase. As a reminder, μ_j is the Rayleigh quotient approximation of λ_2 at step j and $\beta := \widehat{\lambda}^2/4 = \mu_J^2/4$. For notational convenience, we let $\theta_0 := \arccos|q_0^\top v_1|$.

4.1. Delayed Momentum Power Method (DMPower)

Theorem 4 (Convergence of DMPower) *Let J represent the number of steps in the pre-momentum phase and K the number of steps in the momentum phase as in Algorithm 1. Let $\epsilon < 1$ represent the desired error threshold of our v_1 estimates, i.e., $\sin^2\theta(q_t, v_1) < \epsilon$ and $\rho < \min\{1/2, \sqrt{\frac{\lambda_1-\lambda_2}{\lambda_2-\lambda_d}}\}$ represent the desired error threshold of our λ_2 estimates, i.e., $|\mu_k - \lambda_2| < \rho$. Further fix $\tau > 1$ and $\delta = \min\{\rho, \frac{1}{\tau\sqrt{d}}\}$. Then after*

$$J = \mathcal{O}\Big(\frac{1}{\lambda_1-\lambda_2}\log\frac{\tan^2\theta_0}{\delta(\lambda_2-\lambda_3)} + \frac{\lambda_2}{\lambda_2-\lambda_3}\log\frac{d\tau}{\rho}\Big), \tag{6}$$

$$K = \mathcal{O}\Big(\frac{\beta}{\sqrt{\lambda_1^2-4\beta^2}}\log\frac{1}{\epsilon}\Big) \tag{7}$$

pre-momentum and momentum steps, respectively, where $\beta = \widehat{\lambda}_2^2/4 = \mu_J^2/4$, with all but $\tau^{-\Omega(1)} + e^{-\Omega(d)}$ probability, DMPower *outputs a vector q_K with $\sin^2\theta(q_K, v_1) < \epsilon$.*

Remark 1: Our step count for Power+M convergence implicitly assumes $\lambda_2 \leq 2\sqrt{\beta_J}$, i.e., our β approximation lives on the right of $\lambda_2^2/4$. It is possible for $2\sqrt{\beta_J} \leq \lambda_2$, in which case we will still appreciate the effects of acceleration (see Theorem 23).
Remark 2: We require $J = \mathcal{O}\big(\frac{1}{\lambda_1-\lambda_2}\log\frac{\tan^2\theta_0}{\delta(\lambda_2-\lambda_3)} + \frac{\lambda_2}{\lambda_2-\lambda_3}\log\frac{d\tau}{\rho}\big)$ to achieve ρ-accuracy and a further $K = \mathcal{O}\big(\frac{\beta_J}{\sqrt{\lambda_1^2-4\nu_J^2}}\log\frac{1}{\epsilon}\big)$ steps to achieve ϵ-accuracy. In the momentum phase, we absorb q_J as our initial vector, which has already made convergent progress towards v_1. In practice, we will not need all $J + K$ iterations.

Proof Sketch. We divide DMPower by its pre-momentum and momentum phases. The full proof is deferred to Appendix C.

Pre-momentum phase: We regard the inexact deflation step as an exact deflation experiencing a perturbation every round. In Lemma 12, we show that this noise decays at every step and after $J_1 = \mathcal{O}\big(\frac{1}{\lambda_1-\lambda_2}\log\frac{\tan^2\theta_0}{\delta(\lambda_2-\lambda_3)}\big)$ steps, we achieve the Hardt-Price bounds (Hardt and Price, 2014) necessary for convergence of a noisy power method. The convergence rate for noisy power methods (Hardt and Price, 2014, Corollary 1.1) indicates that after an additional $J_2 = \mathcal{O}\big(\frac{\lambda_2}{\lambda_2-\lambda_3}\log\frac{d\tau}{\rho}\big)$ steps, we will reach our desired ρ-accuracy, so in total, we need $J = J_1 + J_2$ iterations to complete the pre-momentum phase. We set $\beta = \frac{\mu_J^2}{4}$ and proceed to the momentum phase.

Momentum phase: Now that our momentum $\widehat{\lambda}_2 = \mu_J^2/4$ coefficient is within the interval $[\lambda_2^2/4, \lambda_1^2/4)$, we may invoke Sa et al.'s Power+M convergence Theorem 4, which states that after $K = \mathcal{O}\Big(\frac{\beta_J}{\sqrt{\lambda_1^2-4\beta_J^2}}\log\big(\frac{1}{\epsilon}\big)\Big)$ steps of iteration on q_J (which we take to be our initial vector for Power+M) steps, we will have that $\sin^2\theta(q_{J+K}, v_1) < \epsilon$.

4.2. Delayed Momentum Streaming Power Method (DMStream)

Theorem 5 (Convergence of DMStream) *Let $\Sigma = \mathbb{E}[(\widehat{A}_t - A) \otimes (\widehat{A}_t - A)]$, where $\widehat{A}_j = \frac{1}{n}\sum_{i=1}^n x_i x_i^\top$ represents any unbiased estimate of A in DMStream with fixed batch size n and $\otimes$ denotes the Kronecker product. Assume we initialize with unit $q_0 \in \mathbb{R}^d$ where $d \gg 0$ and $|v_1^\top q_0| \geq 1/2$. Let $\theta_0 = \arccos|q_0^\top v_1|$. For any $\delta < 1$, $\epsilon < 1$, suppose*

$$||\Sigma|| \leq \frac{(\lambda_1^2-4\beta)\delta\epsilon}{256\sqrt{d}J} = \frac{(\lambda_1^2-4\beta)^{3/2}\delta\epsilon}{256\sqrt{d}\sqrt{\beta}}\log^{-1}\Big(\frac{32}{\delta\epsilon}\Big), \tag{8}$$

where J is the total number of pre-momentum steps we have fixed at runtime. Furthermore, we let $\rho < \min\{1/2, \sqrt{\frac{\lambda_1-\lambda_2}{\lambda_2-\lambda_d}}\}$ represent the error threshold of our λ_2 estimates, i.e., $|\mu_k - \lambda_2| < \rho$. Lastly, fix $\tau > 1$ and $\delta = \min\{\rho, \frac{1}{\tau\sqrt{d}}\}$. If our batch size n is chosen such that

$$\frac{n}{\log^4 n} = \mathcal{O}\Big(\frac{1/\gamma^2 \log d}{(\lambda_2 - \lambda_3)^2 d}\Big). \tag{9}$$

where $\gamma = \frac{\rho(\lambda_2-\lambda_3)}{10\tau\sqrt{d}}$, then after

$$J = \mathcal{O}\left(\frac{1}{\lambda_1 - \lambda_2} \log\left(\frac{\tan^2\theta_0 \tau\sqrt{d}}{\rho(\lambda_2 - \lambda_3)}\right) + \frac{\lambda_2}{\lambda_2 - \lambda_3} \log\frac{d\tau}{\rho}\right), \tag{10}$$

$$K = \frac{\sqrt{\beta}}{\sqrt{\lambda_1 - 4\beta}} \log\Big(\frac{32}{\delta\epsilon}\Big) \tag{11}$$

pre-momentum steps and momentum steps, respectively, with $(1-\frac{1}{n^2})(1-2\delta)(1-\tau^{-\Omega(1)}+e^{-\Omega(d)})$ probability DMStream outputs a vector q_K such that $\sin^2\angle(q_K, v_1) < \epsilon$.

Remark 1: Both phases have a probabilistic guarantee, whereas in DMPower, the momentum phase was deterministically guaranteed. The probabilistic parameter for the pre-momentum phase τ while it is δ for the momentum phase.
Remark 2: We assume a variance condition on our estimates in equation 8. While there are many sophisticated methods designed to reduce variance (Shamir, 2015; Partridge and Calvo, 1998; De Sa et al., 2018) via introduction of step sizes and anchor iterates, we do not explore these options in this paper. However, a simple strategy for reducing variance is to increase batch size n, since we have the relation $||\Sigma|| \leq \frac{\sigma^2}{s}$, where σ^2 is the variance of a single random sample.
Remark 3: The total sample complexity is $n(J + K)$.

Proof Sketch. We divide DMStream by its pre-momentum and momentum phases. The proof is deferred to Appendix D. *Pre-momentum phase:* There are two sources of noise in every round of this phase: H_t the estimation error associated to $(A - \widehat{A}_j)w_j$, and G_t the estimation error associated to $(\lambda_1 v_1 v_1^\top - \nu_j q_j q_j^\top)w_j$. Proposition 18 shows us how to control $||H_t||$ through batch size and Proposition 21 details $||G_t||$. We require $J_1 := \mathcal{O}\left(\frac{1}{\lambda_1-\lambda_2}\log\left(\frac{\tan^2\theta_0\tau\sqrt{d}}{\rho(\lambda_2-\lambda_3)}\right)\right)$ to achieve the Hardt-Price bounds and a further $J_2 := \mathcal{O}\big(\frac{\lambda_2}{\lambda_2-\lambda_3}\log\frac{d\tau}{\rho}\big)$ to acquire the appropriate β, for a total of $J = J_1 + J_2$ pre-momentum rounds.

Momentum phase: Now that our momentum coefficient $\beta = \widehat{\lambda}_2 = \mu_J^2/4$ coefficient is within the interval $[\lambda_2^2/4, \lambda_1^2/4)$, we may invoke Sa et al.'s streaming power convergence Theorem 16, to conclude that we need $K = \frac{\sqrt{\beta}}{\sqrt{\lambda_1-4\beta}}\log\big(\frac{32}{\delta\epsilon}\big)$ steps to complete the momentum phase.

4.3. Precision of Inexact Deflation

In this section we refer to variables and notations as listed in Algorithm 1 and Algorithm 2. We let $\Delta_{1,2} := |\lambda_1 - \lambda_2|$. Although one sets the number of iterations J for the pre-momentum phase at run-time, we are also interested in the accuracy of our λ_2 estimates, that is, $|\lambda_2 - \mu_j|$, since this will determine how quickly our momentum phase converges. In Theorem 4 and Theorem 5, we

discuss how many pre-momentum iterations J are needed to ensure $|\mu_j - \mu_{j-1}| \leq \rho$, where ρ is an error threshold which controls the accuracy of our final estimate $\widehat{\lambda}_2$. As such, for the remainder of this section we will look at a modification of DMPower and DMStream where ρ is provided as a hyperparameter and how it affects our overall convergence.

Effects of inaccurate approximation of λ_2 In Theorem 4 and Theorem 5, the overall convergence rate is dependent on our pre-momentum phase error $|\lambda_2 - \widehat{\lambda}_2| < \rho$, where $\widehat{\lambda}_2 = \mu_J$. Ultimately, we need our approximated momentum coefficient $\beta = \widehat{\lambda}_2^2/4 \in [\lambda_2^2/4, \lambda_1^2/4)$. In fact, even if $\beta < \lambda_2^2/4$, the momentum phase will still converge, and will still experience similar momentum effects as long as $|\beta - \lambda_2^2/4| < |\lambda_1^2/4 - \lambda_2^2/4|$, which is a generalization of Theorem 2, provided in Theorem 23. We establish a proposition suggesting how accurately $\widehat{\lambda}_2$ must approximate λ_2:

Proposition 6 *The momentum phase of DMPower set with momentum coefficient $\beta = \mu_J^2/4$ converges if and only if $|\lambda_2 - \widehat{\lambda}_2| \leq \Delta_{1,2}$.*

The proof is deferred to Appendix F. In lieu of fixed number of iterations for the pre-momentum phase, one may instead choose to exit the first for-loop if $|\mu_j - \mu_{j-1}| < \rho$, where we have now adopted ρ as a hyperparameter, which is a far less aggressive option than the guesswork required with randomly selecting a convergent β as in Power+M. We assume this termination condition for the remainder of our discussion.

By Theorem 4, the $\mu_j \to \lambda_2$, so we argue it is fair to assume $|\mu_j - \mu_{j-1}| \approx |\mu_j - \lambda_2|$. In which case, by the triangle inequality and Proposition 4.3, we have that $|\mu_j - \mu_{j-1}| < \rho$ if and only if $\rho \lesssim \frac{1}{2}\Delta_{1,2}$. Loosening ρ beyond this bound does not necessarily result in divergence, however, which we experimentally observe and discuss in the next section. Current state-of-the-art bounds do not say anything meaningful about how quickly we can expect to converge/diverge outside of this bound. According to Theorem 4, the looser we set ρ, the fewer iterations we can expect to run in the pre-momentum phase. However, the tighter we set ρ, the closer our β approaches $\lambda_2^2/4$, which is the optimal assignment for the momentum phase.

Practical selection of ρ Instead of setting ρ to an exceedingly small value close to machine precision, we experimentally demonstrate in Figure 3 that DMPower is successful in the setting where $\Delta_{1,2} = \Delta_{2,3}$ for a variety of ρ selections. In Figure 3 we have set $\rho = \sqrt[k]{\epsilon}$ for $k = 1, 2, 3, 4$ to demonstrate flexibility, but they are independent precision bounds; ρ depends only on $\Delta_{1,2}$ by Proposition 6, not on ϵ. We further observe that our DMPower outperforms the vanilla power method at nearly all error thresholds, and converges at a rate similar to optimal Power+M for tighter eigengaps according to Figure 3. In Figure 2 and Table 2, we demonstrate that setting ρ looser causes us to non-negligibly lose precision in our approximation of λ_2. Setting $\rho = \epsilon$ tighter increases our accuracy, but our iteration complexity worsens by a noticeable amount, especially in the medium eigengap setting.

If one is certain about lower bounds α_1, α_2 of $\Delta_{1,2}$ and $\Delta_{2,3}$, respectively, then we have the following result for DMPower:

Proposition 7 *Assume $\alpha_1 \leq \Delta_{1,2}$ and $\alpha_2 \leq \Delta_{2,3}$. Fix $\rho < \min\{1/2, \sqrt{\alpha_1}\}$, $\tau > 1$ and let $\theta_0 = \arccos|q_0^\top v_1|$, $\delta = \min\{\rho, \frac{1}{\tau\sqrt{d}}\}$. Then after*

$$J = \mathcal{O}\Big(\frac{1}{\alpha_1}\log\frac{\tan^2\theta_0}{\delta\alpha_2} + \frac{1}{\alpha_2}\log\frac{d\tau}{\rho}\Big) \tag{12}$$

pre-momentum phase steps, we output a vector w_J such that if $\mu_J = w_J A w_J^\top$, then $|\lambda_2 - \mu_J| < \rho^2$. Since $\rho^2 < \alpha_1$, our momentum phase will converge with all but $\tau^{-\Omega(1)} + e^{-\Omega(d)}$ probability.

Proof The size of J along with the probabilistic guarantee is a simple corollary of Theorem 4 in conjunction with Lemma 9 to relate $\sin(w_J, v_2)$ to $|\lambda_2 - \mu_J|$. ■

It is important to note the usefulness and practicality of Proposition 7: in other momentum-based methods, having a lower bound on $\Delta_{1,2}$ was not sufficient to guarantee convergence – one would still need the actual location of $[\lambda_2, \lambda_1)$ over [0,1].

5. Experiments

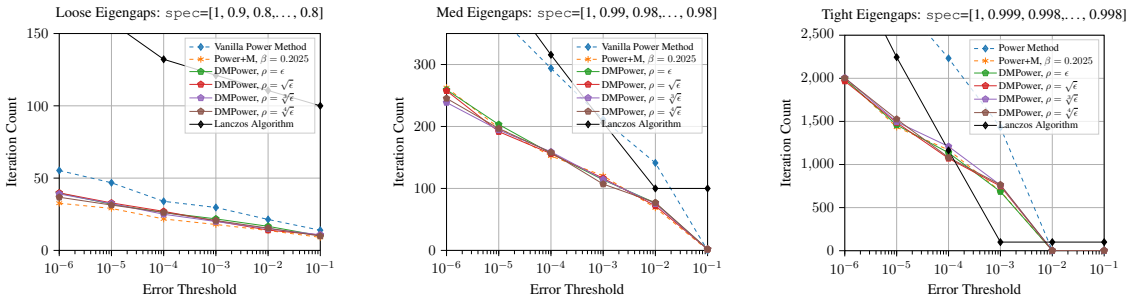
In this section, we discuss the set of experiments we conducted to measure the performance of DMPower and DMStream against a variety of common baseline algorithms. All the experiments were run on an Intel(R) Xeon(R) E5-1650 v4 machine with 32GiB RAM and Linux OS. The algorithms were implemented in Python using the Numpy and SciPy libraries. We refer the reader Appendix E for further experimental design and analyses, including the construction of the synthetic matrices and runtime (wall-time) comparisons.

5.1. DMPower Experiments

Experimental Setup In these experiments, we compare DMPower against the vanilla power method, Power+M with optimal assignment of β, and the Lanczos algorithm. In each experiment, we generate a random symmetric PSD $A \in \mathbb{R}^{100\times 100}$ with a fixed spectrum and conduct PCA using the methods listed above. In Figures 1 and 3, we record the number of iterations that are required to achieve the desired error-tolerance ϵ between our dominant eigenvalue approximates and then take an average over 1000 runs, each time generating a new symmetric PSD A with a specified spectrum. The initial vector q_0 is uniformly set across all methods for every run. Since we know the spectrum of these synthetic matrices, we can initialize Power+M to run with $\beta = \lambda_2^2/4$, the optimal momentum coefficient. In Figure 2, we measure the accuracy of Rayleigh quotient estimates to the Simultaneous Power Iteration rather than convergence speeds.

Iteration Complexity Results As shown in Figure 3, DMPower outperforms the vanilla power method at nearly all ϵ thresholds, across a variety of ρ settings. We set $\rho = \epsilon^{1/k}$ for $k = 1, 2, 3, 4$, but we must stress that ρ and ϵ are independent precision bounds; ρ depends on $\Delta_{1,2}$ by Proposition 6. Furthermore, at medium and tight eigengaps, observe DMPower performs nearly as well as optimal Power+M. When factoring in tridiagonalization iterations, DMPower outperforms the Lanczos algorithm, especially at tighter precisions (the Lanczos is well-known to suffer from numerical instability without corrective measures such as re-orthogonalization (Saad, 2011)). See Table 3 for a full set of raw data. We also measured the proportion of pre-momentum versus momentum phase iterations in Table 4, noting that for a variety of ρ settings, most iterations of DMPower are spent in the momentum phase.

Wall-time Performance At loose and medium eigengaps, Power+M (optimal) registers the lowest wall-time, owing to fewer matrix-vector computations. We remind the reader that this setting is only a baseline and not practically achievable. Of particular note is that at nearly all settings, DMPower runs noticeably faster than the Lanczos algorithm. We refer the reader to Table 5 for full data.



(*a*) $\Delta_{1,2} = \Delta_{2,3} = 0.1$ (*b*) $\Delta_{1,2} = \Delta_{2,3} = 0.01$ (*c*) $\Delta_{1,2} = \Delta_{2,3} = 0.001$

Figure 3: **Iteration Complexity Comparisons**. We compare DMPower (with different ρ settings) against the vanilla power method, Power+M with optimal β, and the Lanczos algorithm. The X-axis corresponds to ϵ, and the Y-axis measures performance by iteration count. Our algorithms demonstrate consistently favorable performance against the vanilla power method baseline and match optimal convergence speeds established by Power+M at medium and tight eigengap settings. *Note 1*: For the Lanczos algorithm, we start by taking 100 tri-diagonalization iterations, which is the recommended number for numerical stability. *Note 2:* We vary ρ according to ϵ for convenience, but they are independent precision thresholds. As stated in Proposition 6, ρ is dependent on $\Delta_{1,2}$.

5.2. DMStream Experiments

Experimental Setup We used a 50,000 sample subset of the MNIST dataset (LeCun et al., 1998), which is represented as a matrix of size 50000×784. The dataset was first pre-processed by centering and dividing the entire matrix by $\sigma\sqrt{784}$. We compared DMStream against Oja's algorithm (Oja, 1982) with varying step sizes, stochastic power iteration (Algorithm 4), and Mini-Batch Power+M set with optimal $\beta = \lambda_2^2/4$, where λ_2 is the second eigenvalue of the (processed) covariance matrix of MNIST. We measured performance by a commonly-used metric $\log_{10}\big(1 - \frac{||X^\top q_K||}{||X^\top v_1||}\big)$. We tested over a variety of batch sizes with $\rho = 0.1$ for DMStream. For each batch size, we ran 50 iterations and averaged the results over 10 runs.

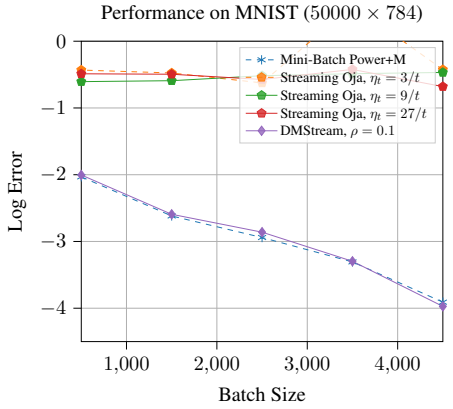


Figure 4: **Performance by batch size**. In this experiment we measure the performance of DMStream versus Oja's algorithm (with various step sizes) and Mini-Batch Power+M (optimal $\beta = \lambda_2^2/4$). We measure performance by the log error $\log_{10}(1 - \frac{||X^\top q_K||}{||X^\top v_1||})$. DMStream exhibits a consistent increase in accuracy as batch-size is increased and mimics the performance of optimal Mini-Batch Power+M.

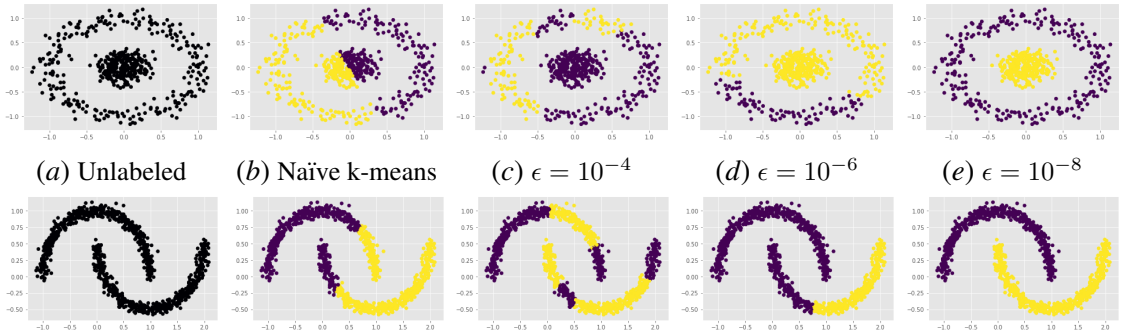
Results Our algorithm performs significantly better than Oja's algorithm for several step sizes and is as accurate as Mini-Batch Power+M initialized with optimal $\beta = \lambda_2^2/4$. As Table 6 demonstrates, our accuracy eventually ceases to improve for fixed batch size, but Figure 4 indicates noticeable improvement with increasing batch size, owing to reduced variance of our unbiased estimates $\widehat{A}_t$.

5.3. Application: Spectral Clustering

Overview Clustering is the unsupervised learning task of dividing a collection of data points into distinct groups or "clusters." The k-means algorithm (Lloyd, 1982) is a popular clustering method which has found ubiquitous use in machine learning, including social network analysis (Mishra et al., 2007), image processing (Shi and Malik, 2000), and other data mining tasks.

However, k-means is limited in effectiveness when applied to nonlinear data. Spectral clustering is an extension of k-means used to properly separate nonlinear data. Whereas k-means is applied directly on the data points $\{x_i\}_{i=1}^n$, spectral clustering first begins with a symmetric affinity matrix $A_{ij} = s(x_i, x_j)$ where s is a similarity function which could be Euclidean distance, for example. We define the diagonal matrix D where $D_{ii} = \sum_{j=1}^n A_{ij}$ and then form the normalized affinity matrix $W = D^{-1}A$. Spectral clustering then computes the top k components of W. These eigenvectors are supplied to the k-means algorithm to obtain the final separation result.

The power iteration may be used to find the eigenvectors of W, which is referred to as power iteration clustering (PIC). The deflation-based PIC algorithm (Thang et al., 2013) we use for our experiments is provided in Appendix H. In this section, we compare variants of the power iteration in carrying out PIC and demonstrate that DMPower is capable of faster eigenvector computation and more accurate data separation when compared against the vanilla power iteration.



(*a*) Unlabeled (*b*) Naïve k-means (*c*) $\epsilon = 10^{-4}$ (*d*) $\epsilon = 10^{-6}$ (*e*) $\epsilon = 10^{-8}$

(*f*) Unlabeled (*g*) Naïve k-means (*h*) $\epsilon = 10^{-4}$ (*i*) $\epsilon = 10^{-6}$ (*j*) $\epsilon = 10^{-8}$

Figure 5: **Performance of DMPM on Spectral Clustering.** We depict the performance of DMPM ($\rho = \sqrt[3]{\epsilon}$) on dividing a collection of data points into their natural, geometrically-partitioned clusters. The first series (a-e) is the concentric circles dataset while the second series (f-j) is the half-moons dataset. In both series, the first image depicts the unlabeled arrangement, the second depicts a naïve application of k-means without spectral clustering, while the last three images depict the performance of k-means assisted by spectral clustering over progressively tighter ϵ error thresholds (thus, requiring more accurate affinity matrix eigenvector approximations). As ϵ grows smaller, the data separation improves, eventually achieving perfect classification by $\epsilon = 10^{-8}$.

Experimental Setup We used two popular toy datasets for clustering: half-moons and concentric circles. The half-moons dataset was generated with 500 samples while the concentric circles set was generated using 1000 samples. We compared DMPower (with various ρ settings) against the vanilla power method and Power+M with optimal assignment of β. We follow the deflation-based approach of spectral clustering, which is outlined in Algorithm 5. Each method was used to recover the top two eigenvectors and eigenvalues of the normalized affinity matrix W of each dataset (our similarity function was pairwise ℓ_2 distance).

We use a practical implementation of the DMPower Algorithm 1: we exit the pre-momentum phase when $||w_{j+1} - w_j|| \leq \rho$, where ρ is a function of ϵ, and exit the momentum phase when $||q_{k-1} - q_j|| \leq \epsilon$. Similarly, for the vanilla power iteration Algorithm 3 and the alternative Power+M update in equation 4, we end the procedure once $||q_{k+1} - q_k|| \leq \epsilon$. Therefore, termination of these algorithms is governed by the closeness of the approximates.

Results DMPower, under all ρ settings, and for most error thresholds ϵ requires fewer iterations to recover the top two eigenvectors of the affinity matrices than the vanilla power method, see Table 6. Furthermore, DMPower with $\rho = \sqrt{\epsilon}, \sqrt[3]{\epsilon}$ performs similarly in both iteration complexity and accuracy when compared against Power+M with optimal $\beta = \lambda_2^2/4$ as reported in to Tables 7 and 8. Figure 5 depicts a progression of separation over the datasets, when DMPower ($\rho = \sqrt[3]{\epsilon}$) is used.

6. Conclusion

In summary, this paper introduces a new scheme for accelerating the vanilla and streaming power methods. The realization of this scheme is the delayed momentum power method (DMPower) and its streaming companion the delayed streaming momentum method (DMStream). DMPower is experimentally shown to outperform the vanilla power iteration and achieves iteration complexity similar to an existing accelerated method Power+M initialized with optimal hyperparameters. Empirically, it also outperforms the state-of-the-art Lanczos algorithm in both wall-time and iteration complexity under several regimes. DMStream is shown to vastly outperform all variations of Oja's algorithm and register close-to-optimal error when compared against the Mini-Batch Power+M initialized with an optimal momentum coefficient. We provide convergence guarantees for DMPower and DMStream using a mixture of perturbation theory and classical power iteration bounds. While other momentum-based methods rely on unrealistic spectral knowledge for acceleration, DMPower and DMStream are both practical and fast.

Acknowledgements

Rabbani thanks John Mattox and Mark Davis for helpful discussions. Huang is supported by a startup fund from the Department of Computer Science of the University of Maryland, National Science Foundation IIS-1850220 CRII Award 030742-00001, DOD-DARPA-Defense Advanced Research Projects Agency Guaranteeing AI Robustness against Deception (GARD), Laboratory for Physical Sciences at University of Maryland, and Adobe, Capital One and JP Morgan faculty fellowships.

References

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Steffen Börm and Christian Mehl. *Numerical methods for eigenvalue problems*. Walter de Gruyter, 2012.

C De Sa, Bryan He, Ioannis Mitliagkas, Christopher Ré, and Peng Xu. Accelerated stochastic power iteration. *Proceedings of machine learning research*, 84:58–67, 2018.

Reza Drikvandi and Olamide Lawal. Sparse principal component analysis for natural language processing. *Annals of Data Science*, pages 1–17, 2020.

Brian Gawalt, Youwei Zhang, and Laurent El Ghaoui. Sparse pca for text corpus summarization and exploration. In *NIPS 2010 Workshop on Low-Rank Matrix Approximation*. Citeseer, 2010.

Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU press, 2012.

Moritz Hardt and Eric Price. The noisy power method: A meta algorithm with applications. In *Advances in Neural Information Processing Systems*, pages 2861–2869, 2014.

Moritz Hardt and Aaron Roth. Beyond worst-case analysis in private singular vector computation. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 331–340, 2013.

Prateek Jain, Chi Jin, Sham M Kakade, Praneeth Netrapalli, and Aaron Sidford. Streaming pca: Matching matrix bernstein and near-optimal finite sample guarantees for oja's algorithm. In *Conference on learning theory*, pages 1147–1164, 2016.

Michel Journée, Yurii Nesterov, Peter Richtárik, and Rodolphe Sepulchre. Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research*, 11 (2), 2010.

Michael Kapralov and Kunal Talwar. On differentially private low rank approximation. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1395–1414. SIAM, 2013.

Cheolmin Kim and Diego Klabjan. Stochastic variance-reduced algorithms for pca with arbitrary mini-batch sizes. *Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Qi Lei, Kai Zhong, and Inderjit S Dhillon. Coordinate-wise power method. In *Advances in Neural Information Processing Systems*, pages 2064–2072, 2016.

Frank Lin and William W Cohen. Power iteration clustering. In *ICML*, 2010a.

Frank Lin and William W Cohen. A very fast method for clustering big text datasets. In *ECAI*, pages 303–308, 2010b.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

Vien V Mai and Mikael Johansson. Noisy accelerated power method for eigenproblems with applications. *IEEE Transactions on Signal Processing*, 67(12):3287–3299, 2019.

Nina Mishra, Robert Schreiber, Isabelle Stanton, and Robert E Tarjan. Clustering social networks. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 56–67. Springer, 2007.

Ioannis Mitliagkas, Constantine Caramanis, and Prateek Jain. Memory limited, streaming pca. In *Advances in neural information processing systems*, pages 2886–2894, 2013.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

Matthew Partridge and Rafael A Calvo. Fast dimensionality reduction and simple pca. *Intelligent data analysis*, 2(3):203–214, 1998.

Renaud-Alexandre Pitaval, Wei Dai, and Olav Tirkkonen. Convergence of gradient descent for low-rank matrix approximation. *IEEE Transactions on Information Theory*, 61(8):4451–4457, 2015.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical mathematics*, volume 37. Springer Science & Business Media, 2010.

Yousef Saad. *Numerical methods for large eigenvalue problems: revised edition*, volume 66. Siam, 2011.

Ohad Shamir. A stochastic pca and svd algorithm with an exponential convergence rate. In *International Conference on Machine Learning*, pages 144–152, 2015.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

Nguyen Duc Thang, Young-Koo Lee, Sungyoung Lee, et al. Deflation-based power iteration clustering. *Applied intelligence*, 39(2):367–385, 2013.

Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

Bao Wang and Qiang Ye. Stochastic gradient descent with nonlinear conjugate gradient-style adaptive momentum, 2020.

Po-An Wang and Chi-Jen Lu. Tensor decomposition via simultaneous power iteration. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3665–3673. JMLR. org, 2017.

Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1):397–434, 2013.

Zhiqiang Xu, Xin Cao, and Xin Gao. Convergence analysis of gradient descent for eigenvector computation. International Joint Conferences on Artificial Intelligence, 2018.

Xiao-Tong Yuan and Tong Zhang. Truncated power method for sparse eigenvalue problems. *Journal of Machine Learning Research*, 14(4), 2013.

Appendix A. Preliminaries and Facts

A.1. Vanilla Power Method

Unless noted otherwise, $\|\cdot\|$ refers to the 2-norm for vectors and the induced 2-norm for matrices. We recall the vanilla power method algorithm.

Algorithm 3 Fetch-Com

Require: $A \in \mathbb{R}^{d\times d}$ diagonalizable, initial vector $q_0 \in \mathbb{R}^d$, error threshold ϵ
Ensure: ϵ-accurate approximation of v_1, λ_1
1: **for** $k = 1, 2, \ldots$ **do** ▷ While $||q_k - v_1|| > \epsilon$
2: $\quad q_k \leftarrow \frac{Aq_{k-1}}{\|Aq_{k-1}\|}$
3: $\quad \nu_k \leftarrow q_k^\top A q_k$ ▷ Rayleigh Quotient
return q_k, ν_k

Although the termination condition relies on an ℓ_2 distance from v_1, this is often replaced with a sine squared error condition, that is, we exit the loop once $\sin^2(\theta_k) \triangleq 1 - (q_k^\top v_1)^2 > \epsilon$. We make use of both variations throughout this paper.

A.2. Power Iteration Bounds

The setting for the next two lemmas is the following: we let $A \in \mathbb{R}^{d\times d}$ be symmetric with spectrum $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$ and associated unit eigenvectors $v_1, v_2, \ldots, v_n$. Note that we do not need the presence of the second eigengap $|\lambda_2| > |\lambda_3|$ for these classical results. We will now establish several well-known inequalities regarding the accuracy of the vanilla power iteration.

Lemma 8 (Quarteroni et al. (2010), p. 194) *Let $q_0 \in \mathbb{R}^d$ such that ${q_0}^\top v_1 \neq 0$. We may write $q_0 = \sum_{i=1}^d c_i v_i$ since A is diagonalizable (being symmetric). Let $C = \left(\sum_{i=1}^d \left(\frac{c_i}{c_1}\right)^2\right)^{1/2}$. We have that*

$$\|\tilde{q_k} - v_1\| \leq C\left|\frac{\lambda_2}{\lambda_1}\right|^k \qquad k \geq 1 \tag{13}$$

where

$$\tilde{q_k} = \frac{q_k}{\|A^k q_0\|}\alpha_1 {\lambda_1}^k = v_1 + \sum_{i=2}^d \frac{c_i}{c_1}\left(\frac{\lambda_i}{\lambda_1}\right)^k v_i, \qquad k = 1, 2, \ldots \tag{14}$$

Remark. Since $\tilde{q_k}$ is nothing more than a scaled version of q_k convenient for analysis, as an abuse of notation, where 2-norm inequalites are invoked involving q_k and v_1, we assume we are working with $\tilde{q_k}$.

Lemma 9 (Golub and Van Loan (2012), p. 451) *Let $q_0 \in \mathbb{R}^n$ such that ${q_0}^\top v_1 \neq 0$. Let $q^k = \frac{A^k q_0}{\|A^k q_0\|}$ and $\nu_k = {q_k}^\top A q_k$, i.e., the basic power iteration approximation of v_1 and λ_1 after k steps. Define $\theta_k \in [0, \pi/2]$ by $\cos(\theta_k) = |{q_k}^\top v_1|$. For $k = 0, 1, 2, \ldots$, we have*

$$|\sin(\theta_k)| \leq \tan(\theta_0)\left|\frac{\lambda_2}{\lambda_1}\right|^k \tag{15}$$

$$|\lambda_1 - \nu_k| \leq \max_{2\leq i\leq d}|\lambda_1 - \lambda_i|\tan(\theta_0)^2\left|\frac{\lambda_2}{\lambda_1}\right|^{2k} \tag{16}$$

We now prove a more general, well-known fact: the Rayleigh quotient is a quadratically-accurate estimate when compared to the sine squared error.

Lemma 10 *Let $w_k \to v_j$ for $\{w_k\}_{i=1}^{\infty}$ a sequence of unit vectors and v_j a unit eigenvector of A associated to eigenvalue λ_j. Let η_k be the Rayleigh quotient of w_k and $\sin^2(\gamma_k) = 1 - (w_k^\top v_j)^2$. Then*

$$|\eta_k - \lambda_i| = \mathcal{O}\big(\sin^2(\gamma_k)\big) \tag{17}$$

Proof Express $w_k = \sum_{i=1}^d c_i v_i$, i.e., as a linear combination in the eigenbasis of A. We have then that

$$\eta_k - \lambda_j = w_k^\top A w_k - \lambda_j = \frac{\sum_{i=1}^d c_i^2 \lambda_i}{\sum_i^d c_i^2} - \lambda_j = \frac{\sum_{i=1}^d (\lambda_i - \lambda_j) c_i^2}{\sum_{i=1}^d c_i^2} \leq \max_{i \neq j} |\lambda_j - \lambda_i| \sum_{i \neq j}^d c_i^2 \tag{18}$$

Since $\sum_{i=1}^d c_i^2 = 1$ (w_k is unit), we have that $\sum\limits_{\substack{i=1 \\ i \neq j}}^{d} c_i^2 = 1 - c_j^2 = 1 - (w_k^\top v_j)^2 = \sin^2(\gamma_j)$. Therefore,

$$|\eta_k - \lambda_j| \leq \max_{i \neq j} |\lambda_j - \lambda_i| \sin^2(\gamma_k) \tag{19}$$

proving our claim. ■

Appendix B. Noisy Deflation

The following results will help us determine how many steps of inexact deflation we need to complete before we achieve small enough noise at each iteration to permit convergence. As usual, we let ν_k and q_k reflect our eigenvalue and eigenvector approximation at step k of a vanilla power iteration with matrix A. We first recall the formulation of a *noisy power method* (NPM). We consider an alternative update step of Algorithm 3:

$$q_k = \frac{A q_{k-1} + G_{k-1}}{\|A q_{k-1} + G_{k-1}\|} \tag{20}$$

Here, G_{k-1} is noise or a perturbation added at each round. The following is a powerful result on the conditions under which NPM can successfully converge:

Corollary 11 (Noisy Power Method Convergence Hardt and Price (2014)) *Let $k \leq p$. Let $U \in \mathbb{R}^{d \times k}$ represent the top k singular vectors of $B \in \mathbb{R}^{d \times d}$ and let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d$ denote its singular values. Suppose X_0 is an orthonormal basis of a random p-dimensional subspace. Further, suppose at every step of NPM we have*

$$5||G_\ell|| \leq \epsilon(\lambda_k - \lambda_{k-1}) \quad \textit{and} \quad 5||U^\top G_\ell|| \leq (\lambda_k - \lambda_{k-1}) \frac{\sqrt{p} - \sqrt{k-1}}{\tau \sqrt{d}} \tag{21}$$

for some fixed parameter τ and $\epsilon < 1/2$. Then with all but $\tau^{\Omega(p+1-k)} + e^{-\Omega(d)}$ probability, there exists an $L = \mathcal{O}\big(\frac{\lambda_k}{\lambda_k - \lambda_{k-1}} \log \frac{d\tau}{\epsilon}\big)$ so that after L steps we have $\big\|(I - X_L X_L^\top) U\big\| \leq \epsilon$.

In the context of Theorem 4, we have that $B = A - \lambda_1 v_1 v_1^\top$, $X_0 = w_0$, $U = v_2$, and G_ℓ is the error $||\lambda_1 v_1 v_1^\top - \nu_\ell q_\ell q_\ell^\top||$, that is, the "inexactness" of our deflation, and d is the dimension. We also note that in relation to A, the deflated matrix B has spectrum $\lambda_2 > \lambda_3 \geq \cdots \lambda_{n-1} \geq \lambda_n = 0$. In this setting, we have that $\left\|(I - X_L X_L^\top)U\right\| = \left\|(I - w_L w_L^\top)v_2)\right\| = \sqrt{1 - (w_L v_2^\top)^2}$, i.e., the sine error between v_2 and w_L. We will now provide an upper bound on $||G_\ell||$ and show it decays with every round. For notational convenience, we denote $\theta_0 := \arccos |q_0^\top v_1|$.

Lemma 12 *If we express $A - \nu_k q_k q_k^\top$ as $A - \lambda_1 v_1 v_1^\top + G_k$, where $G_k = \lambda_1 v_1 v_1^\top - \nu_\ell q_\ell q_\ell^\top$ is a perturbation reflecting the error of our eigenpair approximation, then $\|G_k\| = \mathcal{O}(\tan^2 \theta_0 |\frac{\lambda_2}{\lambda_1}|^k)$.*

Proof Let $q_k = v_1 + \xi_k$ and $\nu_k = \lambda_1 + \gamma_k$, that is, ξ_k and γ_k reflect the perturbations (error) associated with our power iterates. We may express our inexact deflation matrix as follows:

$$A - \nu_k q_k {q_k}^\top = A - (\lambda_1 + \gamma_k)(v_1 + \xi_k)(v_1 + \xi_k)^\top. \tag{22}$$

Expanding, we have that our deviation from $A - \lambda_1 v_1 v_1^\top$ is expressible as

$$G_k = \lambda_1(v_1 \xi_k^\top + \xi_k v_1^\top + \xi_k \xi_k^\top) + \gamma_k(v_1 v_1^\top + v_1 \xi_k^\top + \xi_k v_1^\top + \xi_k \xi_k^\top). \tag{23}$$

$$||G_k|| \leq |\lambda_1|(||v_1 \xi_k^\top|| + ||\xi_k v_1^\top|| + ||\xi_k \xi_k^\top||) + |\nu_k|(||v_1 v_1^\top|| + ||v_1 \xi_k^\top|| + ||\xi_k v_1^\top|| + ||\xi_k \xi_k^\top||) \tag{24}$$

Express the initial vector q_0 of the power iteration in the eigenbasis of A: $v_0 = \sum_{i=1}^n c_i v_i$. For the next step, we recall by the previous two lemmas that

$$|\lambda_1 - \nu_k| \leq \max_{2 \leq i \leq d} |\lambda_1 - \lambda_i| \tan^2(\theta_0) \left|\frac{\lambda_2}{\lambda_1}\right|^{2k}$$

and

$$||q_k - v_1|| \leq \left|\frac{\lambda_2}{\lambda_1}\right|^k \Big(\sum_{i=1}^d (\frac{c_i}{c_1})^2\Big)^{1/2}.$$

For brevity, we denote $C = \left(\sum_{i=1}^d [\frac{c_i}{c_1}]^2\right)^{1/2}$ and $D = \max_{2 \leq i \leq d} |\lambda_1 - \lambda_i| \tan^2(\theta_0)$. Now, it also well known that for $u, v \in \mathbb{R}^l$ that $||uv^t|| = |u^\top v|$. Therefore, in conjunction with application of the Cauchy-Schwarz inequality, we have that

$$||v_1 v_1^\top|| = 1 \tag{25}$$

$$||v_1 \xi_k^\top|| = ||\xi_k v_1^\top|| = |v_1^\top \xi_k| \leq |\xi_k| \leq C \left|\frac{\lambda_2}{\lambda_1}\right|^k \tag{26}$$

$$||\xi_k \xi_k^\top|| = |\xi_k^\top \xi_k| = |\xi_k|^2 \leq C^2 \left|\frac{\lambda_2}{\lambda_1}\right|^{2k} \tag{27}$$

Noting that $\lambda_1 \leq 1$, we have then that

$$||G_k|| \leq 2C \left|\frac{\lambda_2}{\lambda_1}\right|^k + C^2 \left|\frac{\lambda_2}{\lambda_1}\right|^{2k} + D \left|\frac{\lambda_2}{\lambda_1}\right|^{2k} (1 + 2C \left|\frac{\lambda_2}{\lambda_1}\right|^k + C^2 \left|\frac{\lambda_2}{\lambda_1}\right|^{2k}). \tag{28}$$

Thus, $||G_k|| = \mathcal{O}(\tan^2(\theta_0) |\frac{\lambda_2}{\lambda_1}|^k)$. ■

Lemma 13 *Let $v, w \in \mathbb{R}^d$ be of unit length, $A \in \mathbb{R}^{d\times d}$. We have then that*

$$|v^\top A w| \leq ||A||_F \tag{29}$$

Proof Applying the Cauchy-Schwarz inequality and sub-multiplicativity of the induced 2-norm for matrices,

$$|v^\top A w| \leq ||v|| \cdot ||A|| \cdot ||w||_2 = ||A||_2 \leq ||A||_F.$$

■

We may establish how many initial steps of inexact deflation we need to run before it is set on a path towards convergence. Our setting is the same as in the previous lemmas.

Theorem 14 *Let $A \in \mathbb{R}^{d\times d}$ be symmetric PSD and w_0 be the initial unit vector for inexact deflation such that ${w_0}^\top x_2 \neq 0$. Fix $\tau > 1$, $\rho < 1/2$ and $\delta = \min\{\rho, \frac{1}{\tau\sqrt{d}}\}$. Then after $T = \mathcal{O}\big(\frac{1}{\lambda_1-\lambda_2}\log(\frac{\tan^2\theta_0}{\delta(\lambda_2-\lambda_3)})\big)$ steps, the perturbations G_k achieve the Hardt-Price bounds in Equation 21.*

Proof Let $B = A - \lambda_1 v_1 {v_1}^\top$. For $k = 1, 2, \ldots$ we have that step k in the pre-momentum phase involves computing $q_k = \frac{A^k q_{k-1}}{\|A^k q_{k-1}\|}$, $\nu_k = q_k^\top A q_k$ and then an inexact deflation $(A - \nu_k q_k^\top q_k) w_{k-1}$. We have then that inexact deflation at step k is representable as

$$(B + G_k) w_{k-1} = B w_{k-1} + G_k w_{k-1} \tag{30}$$

where G_k is associated with the error $\big\|\lambda_1 v_1 {v_1}^\top - \nu_k q_k {q_k}^\top\big\|$. By Lemma 12, the fact the w_k are unit, and sub-multiplicativity, we have that

$$||G_k w_{k-1}|| \leq ||G_k|| \cdot ||w_{k-1}|| = ||G_k|| = \mathcal{O}\Big(\tan^2\theta_0 \left|\frac{\lambda_2}{\lambda_1}\right|^k\Big). \tag{31}$$

Reminding ourselves of the Hardt-Price bounds, we need any given perturbation to satisfy

$$5||G_k|| \leq \rho(\lambda_2 - \lambda_3) \quad \textit{and} \quad 5||G_k|| \leq (\lambda_2 - \lambda_3)\frac{1}{\tau\sqrt{d}} \tag{32}$$

Solving for k such that $\tan^2\theta_0 \left|\frac{\lambda_2}{\lambda_1}\right|^k$ satisfies these bounds, we arrive at our claim.

■

Appendix C. Proof of Theorem 4

We first outline the setting before proving our main result. We let $A \in \mathbb{R}^{d\times d}$ be symmetric PSD with spectrum

$$1 \geq \lambda_1 > \lambda_2 > \lambda_3 \geq \cdots \geq \lambda_d \geq 0. \tag{33}$$

Distinct to our setting is the presence of a positive eigengap between λ_2 and λ_3. We select $q_0 = \sum_{i=1}^n c_i v_i$ such that $|v_1^\top q_0| \neq 0$, which will be used for the vanilla and momentum power iterations, and $w_0 = \sum_{i=1}^d b_i v_i$ such that $|{v_2}^\top w_0| \neq 0$, which will be used for inexact deflation. Let $\theta_0 = \arccos|q_0^\top v_1|$. Lastly, in the first for-loop we add the theoretical termination condition `while` $|\lambda_2 - \mu_k| > \rho$.

Theorem 15 (Restating of Theorem 4) *Let J represent the number of steps in the pre-momentum phase and K the number of steps in the momentum phase as in Algorithm 1. Let $\epsilon < 1$ represent the desired error threshold of our v_1 estimates, i.e., $\sin^2\theta(q_t, v_1) < \epsilon$ and $\rho < \min\{1/2, \sqrt{\frac{\lambda_1-\lambda_2}{\lambda_2-\lambda_d}}\}$ represent the desired error threshold of our λ_2 estimates, i.e., $|\mu_k - \lambda_2| < \rho$. Select unit $q_0 \in \mathbb{R}^d$ and $w_0 \in \mathbb{R}^d$. Further fix $\tau > 1$ and $\delta = \min\{\rho, \frac{1}{\tau\sqrt{d}}\}$. Then after*

$$J = \mathcal{O}\Big(\frac{1}{\lambda_1-\lambda_2}\log\frac{\tan^2\theta_0}{\delta(\lambda_2-\lambda_3)} + \frac{\lambda_2}{\lambda_2-\lambda_3}\log\frac{d\tau}{\rho}\Big), \tag{34}$$

$$K = \mathcal{O}\Big(\frac{\beta}{\sqrt{\lambda_1^2-4\beta^2}}\log\frac{1}{\epsilon}\Big) \tag{35}$$

pre-momentum and momentum steps, respectively, where $\beta = \widehat{\lambda}_2^2/4 = \mu_J^2/4$, with all but $\tau^{-\Omega(1)} + e^{-\Omega(d)}$ probability, DMPower *outputs a vector q_K with*

$$\sin^2\theta(q_K, v_1) < \epsilon. \tag{36}$$

Proof We divide our proof into an analysis of the pre-momentum stage and then the momentum stage. Specifically, we will first establish how many iterations are needed to acquire a $\beta \in [\lambda_2^2/4, \lambda_1^2/4)$. Then, we will use this β to conduct Power+M updates and derive how many additional steps we will need to obtain a q_k with $1 - (q_k^\top v_1)^2 < \epsilon$.

Pre-momentum phase: By Theorem 14, after $J_1 = \mathcal{O}\big(\frac{1}{\lambda_1-\lambda_2}\log\frac{\tan^2\theta_0}{\delta(\lambda_2-\lambda_3)}\big)$ steps, we achieve the Hardt-Price bounds. Therefore, by Corollary 1.1 in (Hardt and Price, 2014), after a further $J_2 = \mathcal{O}\big(\frac{\lambda_2}{\lambda_2-\lambda_3}\log\frac{d\tau}{\rho}\big)$ iterations the w_t will converge to ρ-accuracy. That is, if we let $J = J_1 + J_2$ we have

$$\Big\|(I - w_J w_J^\top)v_2\Big\| = 1 - (w_J^\top v_2) = \sin(\gamma_J) < \rho \tag{37}$$

with all but $\tau^{-\Omega(1)} + e^{-\Omega(n)}$ probability. Therefore in conjunction with Lemma 10 we have that

$$|\lambda_2 - \mu_J| \le (\lambda_2 - \lambda_d)\sin^2(\gamma_J) < (\lambda_2 - \lambda_d)\rho^2 < \lambda_1 - \lambda_2 \tag{38}$$

so by Proposition 6 we obtain

$$\frac{1}{4}|\lambda_2^2 - \mu_J^2| < \frac{1}{4}|\lambda_1^2 - \lambda_2^2|. \tag{39}$$

We set $\beta = \frac{\mu_J^2}{4}$ as our momentum coefficient and proceed to the momentum phase.

Momentum phase: Our momentum coefficient is now within the interval of acceleration†, i.e., $\beta \in [\lambda_2^2/4, \lambda_1^2/4)$, so we may now invoke Theorem 2, which tell us that after

$$K = \mathcal{O}\left(\frac{\beta}{\sqrt{\lambda_1^2-4\beta^2}}\log\frac{1}{\epsilon}\right) \tag{40}$$

steps of Power+M iteration on q_J (which we now take to be our initial vector for Power+M), we will have that $1 - (q_{J+K}^\top v_1)^2 < \epsilon$, completing our proof.

† Subtly, we assume that $\mu_J \ge \lambda_2$. We discuss the case $\mu_J < \lambda_2$ in Theorem 23. ■

Algorithm 4 Stochastic Power Method/Streaming PCA

Require: Streaming inputs $x_1, x_2, \cdots \in \mathbb{R}^d$, batch size n, unit $q_0 \in \mathbb{R}^d$, iterations T, unit $q_0 \in \mathbb{R}^d$

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Generate unbiased estimate $\widehat{A}_t = \frac{1}{n}\sum_{i=(t-1)n+1}^{tn} x_i x_i^\top$
3: $q_t \leftarrow \widehat{A}_t q_{t-1}$
4: $q_t \leftarrow q_t / \|q_t\|$
5: $\nu_t \leftarrow q_t^\top \widehat{A}_t q_t$ ▷ Inexact Rayleigh quotient estimate of λ_1
return q_T, ν_T

Appendix D. Proof of Theorem 5

We re-outline the typical streaming setting: we have d-dimensional data points $x_1, x_2, \cdots \sim \mathcal{D}$, with underlying covariance matrix $A \in \mathbb{R}^{d\times d}$ with eigenvalues $1 \geq \lambda_1 > \lambda_2 > \lambda_3 \geq \lambda_4 \geq \cdots \geq \lambda_n \geq 0$. Presumably, it is too costly to access and/or store A, but we have access to a stream of inputs $x_1, x_2, \ldots$. Algorithm 4 is a conventional streaming PCA method designed to recover the principal components of A using only a sample of inputs. As an abuse of notation, they are not indexed in any particular order – we receive the data points in a uniformly random manner. At each round, we form an unbiased estimate $\widehat{A} = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top$, where n is a fixed batch size. We are interested in determining how many total samples are needed to output a vector q_t such that $\sin^2(q_t, v_1) < \epsilon$ for a fixed precision $\epsilon < 1$, which is also referred to as the sample complexity.

Similar to the vanilla power method, one may accelerate a conventional streaming PCA by attaching a momentum term. That is, our update step in Algorithm 4 would instead be

$$q_{j+1} \leftarrow \widehat{A}_j q_j - \beta q_{j-1} \tag{41}$$

where β is a momentum coefficient. De Sa et. al provide a guarantee for updates of this kind:

Theorem 16 (De Sa et al. (2018), Theorem 3) *Suppose we run Algorithm 4 with momentum updates as in equation 41. Let $\Sigma = \mathbb{E}[(\widehat{A}_j - A) \otimes (\widehat{A}_j - A)]$. Assume we initialize with unit $q_0 \in \mathbb{R}^d$ and with $|v_1^\top q_0| \geq 1/2$. For any $\delta < 1$ and $\epsilon < 1$, if $2\sqrt{\beta} \in [\lambda_2, \lambda_1)$ and*

$$J = \frac{\sqrt{\beta}}{\sqrt{\lambda_1 - 4\beta}} \log\Big(\frac{32}{\delta\epsilon}\Big) \;\; \text{and} \;\; ||\Sigma|| \leq \frac{(\lambda_1^2 - 4\beta)\delta\epsilon}{256\sqrt{d}J} = \frac{(\lambda_1^2 - 4\beta)^{3/2}\delta\epsilon}{256\sqrt{d}\sqrt{\beta}} \log^{-1}\Big(\frac{32}{\delta\epsilon}\Big), \tag{42}$$

then after J updates with probability at least $1 - 2\delta$, we have that $\sin^2(q_J, v_1) \leq \epsilon$.

Similar to Power+M, streaming PCA with momentum updates experiences noticeable acceleration. However, of particular note is the bounded variance condition and the initialization of β. The former has been addressed through further variance reduction techniques outlined in (De Sa et al., 2018), but current literature does not suggest how to intelligently set β so that it lies in the convergence interval $[\lambda_2^2/4, \lambda_1^2/4)$. Similar to our design of DMPower, our Algorithm 2 successively approximates a convergent β in a pre-momentum phase before dropping into a momentum phase. The convergence analysis of DMStream is more challenging, however, since we have two sources of noise: the estimation error $||A - \widehat{A}||$ and the estimation error of $||v_1 - q_t||$. The remainder of this section is devoted to providing upper bounds on both sources of noise.

We have the following noisy representation of $\widehat{A}x$ for $x \in \mathbb{R}^d$:

$$\widehat{A}x = Ax + H \tag{43}$$

where $H = (A - \widehat{A})x$. In light of Theorem 14, if we can control $||H||$, then we can produce a convergent noisy power method. To this end, we provide several results to aid our analysis, some of which are not proven here. The first result bounds the error in a single step of a streaming PCA:

Lemma 17 (Hardt and Price (2014), Lemma 3.5) *Let A be a covariance matrix as in the setting described above and $\widehat{A} = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top$ an empirical estimate based off a streaming batch $x_1, x_2, \ldots, x_n$. Consider the noisy representation as in equation 43. Then with all but $\mathcal{O}(1/n^2)$ probability*

$$||H|| \leq \sqrt{\frac{\log^4 n \log d}{n}} + \frac{1}{n^2} \quad \textit{and} \quad ||v_1^\top H|| \leq \sqrt{\frac{\log^4 n \log d}{n} + \frac{1}{n^2}} \tag{44}$$

We now consider $H_t = (A - \widehat{A}_t)q_{t-1}$, where $\widehat{A}_t$ and q_t are the unbiased estimate of A and v_1 respectively in round j as in Algorithm 4.

Proposition 18 (Hardt and Price (2014), Theorem 3.2) *Choose batch size n such that*

$$\frac{n}{\log^4 n} = \mathcal{O}\Big(\frac{1/\epsilon^2 \log d}{(\lambda_1 - \lambda_2)^2 d}\Big) \tag{45}$$

for $\epsilon < 1/2$ and we will have by Lemma 17 that

$$||H_t|| \leq \frac{\epsilon(\lambda_1 - \lambda_2)}{5} \quad \textit{and} \quad ||v_1^\top H_t|| \leq \frac{\lambda_1 - \lambda_2}{5\sqrt{d}}, \tag{46}$$

thereby satisfying the Hardt-Price bounds as in equation 21. Thus, by Theorem 14, after $T = \mathcal{O}(\log(d/\epsilon)/(1 - \lambda_2/\lambda_1))$ iterations, we have with all but $1 - \max\{1, T/n^2\}$ probability that Algorithm 2 outputs q_T such that $1 - (v_1^\top q_T)^2 < \epsilon$.

Now, consider the inexact update in Algorithm 2, $(\widehat{A}_t - \nu_t q_t q_t^\top)w_t$. For convenience, we let $B := A - \lambda_1 v_1 v_1^\top$, the exact deflation matrix. We may express this as

$$(\widehat{A}_t - \nu_t q_t q_t^\top)w_{t-1} = Bw_{t-1} + H_t + G_t \tag{47}$$

where $H_t = (A - \widehat{A}_t)w_{t-1}$ and $G_t = (\lambda_1 v_1 v_1^\top - \nu_t q_t q_t^\top)w_{t-1}$. By Proposition 18 we know how to control $||H_t||$, so we will now focus our attention on $||G_t||$. We will conduct analysis in the same spirit as Lemma 12, but first we will require a another pair of results. The first lemma demonstrates that if we satisfy the Hardt-Price bounds of equation 21, then $\tan\theta(v_1, q_t)$ decreases multiplicatively with each step of a noisy power method.

Lemma 19 (Hardt and Price (2014), Lemma 2.3 (modified)) *Let $v_1 \in \mathbb{R}^d$ be the dominant eigenvector with eigenvalue λ_1 of A, λ_2 the second dominant eigenvalue, $x \in \mathbb{R}^d$ a unit vector. Let $G \in \mathbb{R}^d$ and $\theta_0 = \arccos|v_1^\top x|$ satisfy*

$$4||v_1^\top G|| \leq (\lambda_1 - \lambda_2)\cos(\theta_0) \tag{48}$$

$$4||G|| \leq (\lambda_1 - \lambda_2)\epsilon \tag{49}$$

for some $\epsilon < 1$. Then

$$\tan\theta(v_1, Ax + G) \leq \max\{\epsilon, \max\{\epsilon, \Big(\frac{\lambda_2}{\lambda_1}\Big)^{1/4}\}\tan\theta_0\}. \tag{50}$$

We now provide a bound on the $\sin\theta(v_1, q_t)$ using Lemma 19.

Proposition 20 (Hardt and Price (2014), Theorem 2.4 (rephrased)) *Suppose q_0 is the initial vector supplied to a noisy power method, $\theta_0 = |v_1^\top q_0|$, and G_t is the noise experienced at round t. Further suppose that*

$$5||v_1^\top G_t|| \leq (\lambda_1 - \lambda_2)\cos\theta_0 \tag{51}$$

$$5||G_t|| \leq \epsilon(\lambda_1 - \lambda_2) \tag{52}$$

holds at every stage including and after round t for some $\epsilon < 1/2$. Then

$$\sin\theta(q_{t+k}, v_1) \leq \max\{\epsilon, (\lambda_2/\lambda_1)^{k/4}\tan\theta_0\} \tag{53}$$

Proof Let $l \geq t$. Since $|G_l||$ satisfies the modified Hardt-Price bounds of Lemma 19, we have that

$$\tan\theta(v_1, q_l) \leq \max\{\epsilon, \max\{\epsilon, \tan\theta_0\}\}. \tag{54}$$

For $\epsilon < 1/2$ we have that,

$$\cos\theta(v_1, q_l) \geq \min\{1 - \epsilon^2/2, \cos\theta_0\} \geq \frac{7}{8}\cos\theta_0, \tag{55}$$

which means that we may invoke Lemma 19 at every step $l \geq t$. This gives us

$$\tan(v_1, q_{l+1}) = \tan\theta(v_1, Aq_l + G) \leq \max\{\epsilon, \max\{\epsilon, \Big(\frac{\lambda_2}{\lambda_1}\Big)^{1/4}\}\tan\theta(v_1, q_l)\}. \tag{56}$$

Extending this inequality recursively for $l + k$ and noting that $\sin\theta(q_l, v_1) \leq \tan\theta(q_l, v_1)$ gives us our claim. ■

We are now prepared to conduct analysis on $||G_t||$ in equation 47.

Proposition 21 *Choose $\epsilon < 1/2$ and n such that*

$$\frac{n}{\log^4 n} = \mathcal{O}\Big(\frac{1/\epsilon^2 \log d}{(\lambda_1 - \lambda_2)^2 d}\Big). \tag{57}$$

. Let

$$\phi_t = |\lambda_1 - \lambda_d|\tan^2\theta(q_0, v_1)\left|\frac{\lambda_2}{\lambda_1}\right|^{2t} + \min\{\frac{\epsilon(\lambda_1 - \lambda_2)}{5}, \frac{\lambda_1 - \lambda_2}{5\sqrt{d}}\} \tag{58}$$

$$\psi_t = \sqrt{2 - 2\sqrt{(1 - \min\{1, \max\{\epsilon^2, (\lambda_2/\lambda_1)^{t/2}\tan^2\theta_0\})\}}}. \tag{59}$$

Then with all but $\mathcal{O}(1/n^2)$ probability, we have that $\|G_t\| = \mathcal{O}(\max\{\phi_t, \psi_t\})$.

Proof We first consider $|\lambda_1 - \nu_t| = |\lambda_1 - q_t^\top \widehat{A}_t q_t|$. Let $\theta_0 = q_0^\top v_1$. First observe that since $q_t \to v_1$, we have by Lemma 10 that $|\lambda_1 - q_t^\top A q_t| = \mathcal{O}(\sin^2(q_t, v_1))$. We have then that

$$|\lambda_1 - \nu_t| = |\lambda_1 - q_t^\top \widehat{A}_t q_1| = |\lambda_1 - q_t^\top A q_t + q_t^\top A q_t - q_t^\top \widehat{A}_t q_t| \tag{60}$$

$$\leq |\lambda_1 - q_t^\top A q_t| + |q_t^\top (A - \widehat{A}_t) q_t| \tag{61}$$

$$\leq |\lambda_1 - \lambda_d|\tan^2\theta(q_0, v_1)\left|\frac{\lambda_2}{\lambda_1}\right|^{2t} + \min\{\frac{\epsilon(\lambda_1 - \lambda_2)}{10}, \frac{\lambda_1 - \lambda_2}{10\sqrt{d}}\} \tag{62}$$

where the last inequality follows from applying the result on Rayleigh quotient approximation in Lemma 10 to the left term and Proposition 18 along with the Cauchy-Schwarz inequality to the right term.

We now examine $||v_1 - q_t||$. We have chosen n in such a way that we satisfy the Hardt-Price bounds of equation 21, therefore, we may invoke Proposition 20 and conclude that

$$||v_1 - q_t|| \leq \sqrt{2 - 2\sqrt{(1 - \min\{1, \max\{\epsilon^2, (\lambda_2/\lambda_1)^{t/2} \tan^2\theta_0\})\}}}, \tag{63}$$

where we have used the identity $||v_1 - q_t||^2 = 2 - 2\cos\theta(v_1, q_t)$ since unit v_1 and q_t are unit vectors. Following the exact same analysis as in the proof of Lemma 12, we will arrive at

$$||G_t|| = \mathcal{O}(\max\{|\lambda_1 - \nu_t|, ||v_1 - q_t||\}). \tag{64}$$

Taking the upper bounds on $||\lambda_1 - \nu_t||$ and $||v_1 - q_t||$ which we derived above gives us our result. ■

We are now prepared to prove our main convergence theorem for DMStream.

Theorem 22 (Restating of Theorem 5) *Let $\Sigma = \mathbb{E}[(\widehat{A}_t - A)\otimes(\widehat{A}_t - A)]$, where $\widehat{A}_j = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top$ represents any unbiased estimate of A in DMStream with fixed batch size n. Assume we initialize with a unit $q_0 \in \mathbb{R}^d$ where $d \gg 0$ and $|v_1^\top q_0| \geq 1/2$. Let $\theta_0 = \arccos|q_0^\top v_1|$. For any $\delta < 1$, $\epsilon < 1$, suppose*

$$||\Sigma|| \leq \frac{(\lambda_1^2 - 4\beta)\delta\epsilon}{256\sqrt{d}J} = \frac{(\lambda_1^2 - 4\beta)^{3/2}\delta\epsilon}{256\sqrt{d}\sqrt{\beta}} \log^{-1}\Big(\frac{32}{\delta\epsilon}\Big), \tag{65}$$

where J is the total number of pre-momentum steps we have fixed at runtime. Furthermore, we let $\rho < \min\{1/2, \sqrt{\frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_d}}\}$ represent the error threshold of our λ_2 estimates, i.e., $|\mu_k - \lambda_2| < \rho$. Lastly, fix $\tau > 1$ and $\delta = \min\{\rho, \frac{1}{\tau\sqrt{d}}\}$. If the batch size n is chosen such that

$$\frac{n}{\log^4 n} = \mathcal{O}\Big(\frac{1/\gamma^2 \log d}{(\lambda_2 - \lambda_3)^2 d}\Big). \tag{66}$$

where $\gamma = \frac{\rho(\lambda_2 - \lambda_3)}{10\tau\sqrt{d}}$, then after

$$J = \mathcal{O}\left(\frac{1}{\lambda_1 - \lambda_2}\log\left(\frac{\tan^2\theta_0 \tau\sqrt{d}}{\rho(\lambda_2 - \lambda_3)}\right) + \frac{\lambda_2}{\lambda_2 - \lambda_3}\log\frac{d\tau}{\rho}\right), \tag{67}$$

$$K = \frac{\sqrt{\beta}}{\sqrt{\lambda_1 - 4\beta}}\log\Big(\frac{32}{\delta\epsilon}\Big) \tag{68}$$

pre-momentum steps and momentum steps respectively, with $(1 - \frac{1}{n^2})(1 - 2\delta)(1 - \tau^{-\Omega(1)} + e^{-\Omega(d)})$ probability DMStream outputs a vector q_K such that

$$\sin^2\angle(q_K, v_1) < \epsilon. \tag{69}$$

Proof As a reminder, our exact deflation matrix B has spectrum $\lambda_2 > \lambda_3 \geq \lambda_4 \geq \cdots \lambda_n$ with respective eigenvectors $v_2, v_3, \ldots, v_n$. We will show that for our choice of n and J, that $||H_t|| + ||G_t||$ for all $t > J$ satisfy the Hardt-Price bounds, therefore allowing our inexact deflation to probabilistically succeed.

First we examine $||H_t||$. By our choice of n and since $d \gg 0$, we have by Proposition 18 that

$$||H_t|| \leq \frac{\gamma(\lambda_2 - \lambda_3)}{5} = \frac{\rho(\lambda_2 - \lambda_3)^2}{50\tau\sqrt{d}} < \frac{\rho(\lambda_2 - \lambda_3)}{10\tau\sqrt{d}}, \tag{70}$$

$$||v_2^\top H_t|| \leq ||H_t|| < \frac{\rho(\lambda_2 - \lambda_3)}{10\tau\sqrt{d}}. \tag{71}$$

Now, we will analyze $||G_t||$. By Proposition 21, we must consider both ϕ_t and ψ_t.
Case 1: $\phi_t > \psi_t$. For our choice of n we have that

$$\phi_t = |\lambda_1 - \lambda_d| \tan^2\theta(q_0, v_1) \left|\frac{\lambda_2}{\lambda_1}\right|^{2t} + \min\{\frac{\gamma(\lambda_1 - \lambda_2)}{5}, \frac{\lambda_1 - \lambda_2}{5\sqrt{d}}\} \tag{72}$$

$$= |\lambda_1 - \lambda_d| \tan^2\theta(q_0, v_1) \left|\frac{\lambda_2}{\lambda_1}\right|^{2t} + \frac{\rho(\lambda_2 - \lambda_3)}{50\tau\sqrt{d}} \tag{73}$$

$$< |\lambda_1 - \lambda_d| \tan^2\theta(q_0, v_1) \left|\frac{\lambda_2}{\lambda_1}\right|^{2t} + \frac{\rho(\lambda_2 - \lambda_3)}{20\tau\sqrt{d}}. \tag{74}$$

Solving for $|\lambda_1 - \lambda_d| \tan^2\theta_0 \left|\frac{\lambda_2}{\lambda_1}\right|^{2t} < \frac{\rho(\lambda_2 - \lambda_3)}{20\tau\sqrt{d}}$ we get

$$t = \mathcal{O}\left(\frac{1}{\lambda_1 - \lambda_2} \log\left(\frac{\tan^2\theta_0 \tau\sqrt{d}}{\rho(\lambda_2 - \lambda_3)}\right)\right), \tag{75}$$

therefore, in this many steps we will have that $\phi_t < \frac{\rho(\lambda_2 - \lambda_3)}{10\tau\sqrt{d}}$.
Case 2: $\psi_t \geq \phi_t$. If $\min\{1, \max\{\gamma^2, (\lambda_2/\lambda_1)^{t/2} \tan^2\theta_0\})\} = 1$ then $||G_t|| \leq \psi_t = 0$, i.e., we have 0 noise. So we consider the more interesting case where our minimum is $\max\{\gamma^2, (\lambda_2/\lambda_1)^{t/2} \tan^2\theta_0\}$. We first note that for any $\alpha \leq 1$, we have that

$$\psi_t = \sqrt{2 - 2\sqrt{1 - \alpha}} \leq \alpha. \tag{76}$$

If for all t our max is γ^2, we have that

$$||G_t|| \leq \psi_t \leq \gamma^2 \leq \frac{\rho^2(\lambda_2 - \lambda_3)^2}{100\tau^2 d} < \frac{\rho(\lambda_2 - \lambda_3)}{10\tau\sqrt{d}}. \tag{77}$$

Otherwise, solving for

$$\left|\frac{\lambda_2}{\lambda_1}\right|^{t/2} \tan^2\theta(q_0, v_1) < \frac{\rho(\lambda_2 - \lambda_3)}{10\tau\sqrt{d}} \tag{78}$$

we get

$$t = \mathcal{O}\left(\frac{1}{\lambda_1 - \lambda_2} \log\left(\frac{\tan^2\theta_0 \tau\sqrt{d}}{\rho(\lambda_2 - \lambda_3)}\right)\right). \tag{79}$$

Now, for our choice of n, after $J_1 := t = \mathcal{O}\left(\frac{1}{\lambda_1-\lambda_2}\log\left(\frac{\tan^2\theta_0\tau\sqrt{d}}{\rho(\lambda_2-\lambda_3)}\right)\right)$ steps, we have that for all $t > J_1$,

$$||H_t + G_t|| \leq ||H_t|| + ||G_t|| < \frac{\rho(\lambda_2-\lambda_3)}{10\tau\sqrt{d}} + \frac{\rho(\lambda_2-\lambda_3)}{10\tau\sqrt{d}} < \frac{\rho(\lambda_2-\lambda_3)}{5}. \tag{80}$$

and

$$||v_2(H_t + G_t)|| \leq ||H_t|| + ||G_t|| < \frac{\rho(\lambda_2-\lambda_3)}{10\tau\sqrt{d}} + \frac{\rho(\lambda_2-\lambda_3)}{10\tau\sqrt{d}} < \frac{\lambda_2-\lambda_3}{5\tau\sqrt{d}}, \tag{81}$$

thereby satisfying the Hardt-Price bounds of equation 21, so after a further $J_2 := \mathcal{O}\big(\frac{\lambda_2}{\lambda_2-\lambda_3}\log\frac{d\tau}{\rho}\big)$ steps, our pre-momentum phase, with a total of $J = J_1 + J_2$ steps, outputs a vector q_J such that $\sin\theta(q_J, v_1) < \rho$ with probability $(1-\frac{1}{n^2})(1-\tau^{-\Omega(1)}+e^{-\Omega(d)})$. Through dual application of Lemma 10 and Proposition 6, similar to the proof of Theorem 4, we have that $\beta = \mu_J^2/4 \in [\lambda_2^2/4, \lambda_2^2/4)$, therefore, we may proceed to the momentum phase.

By our assumptions on the variance of our unbiased estimates $\widehat{A}_t$ in relation to J, we have by Theorem 16 that after a further $K = \frac{\sqrt{\beta}}{\sqrt{\lambda_1-4\beta}}\log\left(\frac{32}{\delta\epsilon}\right)$ steps our entire algorithm outputs a vector q_{J+K} with $\sin^2\theta(q_{J+K}, v_1) < \epsilon$, with probability $(1-2\delta)$. By multiplying the probability of the pre-momentum phase succeeding with the probability of the momentum phase succeeding, our full claim follows. ■

Appendix E. Experimental Data

We provide raw experimental data used to generate various plots displayed in this paper (mostly related to iteration complexity) and include comparative wall-time benchmarks for vanilla power method, Power+M, DMPower, and the Lanczos algorithm.

Table 1: Iterations required for Power+M to converge at various sub-optimal and optimal β assignments. $\beta = \lambda_2^2/4 = 0.2025$ is the optimal momentum coefficient at this setting, where $\lambda_2 = 0.9$.

Sub-optimal β Selection for Power+M Data with Loose Eigengaps: `spec`=[1,0.9,0.8,...,0.8]							
Error Threshold (ϵ)	10e-9	10e-8	10e-7	10e-6	10e-5	10e-4	10e-3
Vanilla Power Method	81.097	70.309	59.230	48.746	37.263	26.873	16.671
Power+M, $\beta = 0.1025$	60.463	52.565	44.420	36.745	28.272	20.561	12.859
Power+M, $\beta = 0.2025$	**34.986**	**30.954**	**26.764**	**22.879**	**18.497**	**14.472**	**10.205**
Power+M, $\beta = 0.3025$	43.579	38.605	32.439	27.265	20.991	16.182	10.981
Power+M, $\beta = 0.4025$	89.440	76.524	65.099	54.101	42.651	31.200	19.501
Power+M, $\beta = 0.4225$	111.972	94.846	81.729	68.635	50.530	36.275	23.144
Power+M, $\beta = 0.4525$	179.062	155.476	130.219	106.71	81.776	58.348	32.412

Table 2: Absolute difference between final approximation of λ_2 and true value of λ_2.

Accuracy vs Simultaneous Power Data with Loose Eigengaps: `spec`=[1,0.9,0.8,...,0.8]							
Error Threshold (ϵ)	10e-9	10e-8	10e-7	10e-6	10e-5	10e-4	10e-3
Simultaneous PM	0.1723	0.1721	0.1684	0.1628	0.1466	0.1107	0.1126
DMPower, $\rho = \epsilon$	**0.0000**	**0.0000**	**0.0003**	**0.0017**	**0.0054**	**0.0370**	0.0678
DMPower, $\rho = \sqrt{\epsilon}$	0.0065	0.0145	0.0316	0.0696	0.0672	0.0552	0.0574
DMPower, $\rho = \sqrt[3]{\epsilon}$	0.0570	0.0692	0.0648	0.0545	0.0570	0.0577	0.0505
DMPower, $\rho = \sqrt[4]{\epsilon}$	0.0667	0.0538	0.0537	0.0564	0.0587	0.0529	**0.0488**

Table 3: Iterations for Vanilla PM, Power+M, and DMPower at `spec`=[1, 0.99, 0.98,..., 0.98].

Iteration Complexity Data with $A \in \mathbb{R}^{10\times 10}$						
Error Threshold (ϵ)	10e-2	10e-3	10e-4	10e-5	10e-6	10e-7
Vanilla Power Method	**2.0**	77.18	185.4	285.4	385.42	474.62
Power+M, $\beta = \lambda_2^2/4$	2.0	40.5	98.86	143.42	199.82	**231.74**
DMPower, $\rho = \epsilon$	3.0	44.5	104.7	139.9	197.08	243.84
DMPower, $\rho = \sqrt{\epsilon}$	3.0	33.86	93.7	153.48	192.86	249.52
DMPower, $\rho = \sqrt[3]{\epsilon}$	3.0	36.8	102.36	**138.82**	200.48	234.48
DMPower, $\rho = \sqrt[4]{\epsilon}$	3.0	37.7	**81.28**	144.02	**192.72**	243.58
Lanczos Algorithm	10.0	**10.0**	89.08	184.66	284.78	388.56

Iteration Complexity Data with $A \in \mathbb{R}^{100\times 100}$						
Error Threshold (ϵ)	10e-2	10e-3	10e-4	10e-5	10e-6	10e-7
Vanilla Power Method	**1.0**	141.18	211.34	293.94	378.1	472.98
Power+M, $\beta = \lambda_2^2/4$	2.0	**68.66**	120.4	**152.32**	197.84	262.8
DMPower, $\rho = \epsilon$	3.0	76.54	113.9	156.6	203.26	259.2
DMPower, $\rho = \sqrt{\epsilon}$	3.0	71.92	116.5	156.08	**191.64**	257.66
DMPower, $\rho = \sqrt[3]{\epsilon}$	3.0	75.44	114.6	159.08	194.34	**238.66**
DMPower, $\rho = \sqrt[4]{\epsilon}$	3.0	77.1	**107.04**	158.2	196.74	245.38
Lanczos Algorithm	100.0	100.0	206.06	315.62	431.12	551.76

Iteration Complexity Data with $A \in \mathbb{R}^{500\times 500}$						
Error Threshold (ϵ)	10e-2	10e-3	10e-4	10e-5	10e-6	10e-7
Vanilla Power Method	**1.0**	185.98	259.22	360.12	429.16	489.4
Power+M, $\beta = \lambda_2^2/4$	1.0	93.16	133.48	163.18	214.64	259.0
DMPower, $\rho = \epsilon$	2.0	102.94	133.18	163.98	**203.88**	264.72
DMPower, $\rho = \sqrt{\epsilon}$	2.0	**93.06**	**131.76**	171.62	207.24	252.4
DMPower, $\rho = \sqrt[3]{\epsilon}$	2.0	97.56	143.66	170.06	212.36	**252.24**
DMPower, $\rho = \sqrt[4]{\epsilon}$	2.0	94.12	137.5	**161.2**	213.24	262.36
Lanczos Algorithm	500.0	500.0	607.24	717.84	833.56	964.18

Table 4: Percentage of iterations in the *momentum phase* for `spec`=[1, 0.99, 0.98,..., 0.98] with $A \in \mathbb{R}^{100\times 100}$. DMPower across a variety of ρ settings spends the vast majority of its time in the less computationally-intensive momentum phase.

Phases Iteration Complexity Data with $A \in \mathbb{R}^{100\times 100}$				
Error Threshold (ϵ)	10e-4	10e-5	10e-6	10e-7
DMPower, $\rho = \epsilon$	64.72%	88.15%	99.06%	98.71%
DMPower, $\rho = \sqrt{\epsilon}$	99.6%	99.49%	99.28%	98.72%
DMPower, $\rho = \sqrt[3]{\epsilon}$	99.6%	99.49%	99.29%	98.75%

Table 5: Recorded wall-time convergence speeds (in nanoseconds) for vanilla power method, Power+M with optimal β assignment, and various settings of DMPower, and the Lanczos algorithm. Performed 1000 calls using random PSD of various sizes with fixed spectrum $\lambda_1 = 1, \lambda_2 = 0.99, \lambda_3 = 0.98$, and remaining eigenvalues set to 0.98. In several instances DMPower exhibits faster wall-time speeds than Power+M with optimal β assignment, consistently outperforms Lanczos, and markedly accelerates the vanilla power method at all error thresholds (ϵ) tighter than 0.1.

Wall-Time Performance Data with $A \in \mathbb{R}^{10\times 10}$						
Error Threshold (ϵ)	10e-2	10e-3	10e-4	10e-5	10e-6	10e-7
Vanilla Power Method	**67.32**	3186.6	7440.56	11497.34	15388.04	18846.76
Power+M, $\beta = \lambda_2^2/4$	101.54	1941.46	4539.84	**6140.58**	**8514.94**	**9984.74**
DMPower, $\rho = \epsilon$	251.28	2117.2	4813.14	6436.1	8982.4	11550.66
DMPower, $\rho = \sqrt{\epsilon}$	262.54	1696.62	4383.94	7066.94	8869.68	11470.54
DMPower, $\rho = \sqrt[3]{\epsilon}$	260.84	1837.14	4732.32	6508.14	9440.62	16254.96
DMPower, $\rho = \sqrt[4]{\epsilon}$	256.94	1878.86	**3830.36**	6586.34	8938.38	11294.3
Lanczos Algorithm	1463.06	**1471.86**	5818.92	7836.14	11705.58	22829.88

Wall-Time Performance Data with $A \in \mathbb{R}^{100\times 100}$						
Error Threshold (ϵ)	10e-2	10e-3	10e-4	10e-5	10e-6	10e-7
Vanilla Power Method	**107.24**	10027.98	16213.82	21231.46	29691.06	31801.24
Power+M, $\beta = \lambda_2^2/4$	132.2	**4890.36**	9701.76	**10840.92**	**13930.12**	21100.88
DMPower, $\rho = \epsilon$	432.06	5969.96	8910.08	11985.2	15986.18	**19431.32**
DMPower, $\rho = \sqrt{\epsilon}$	457.82	5765.98	9028.7	12397.06	15048.34	19432.9
DMPower, $\rho = \sqrt[3]{\epsilon}$	435.48	6113.44	9393.82	12575.0	15028.84	20037.1
DMPower, $\rho = \sqrt[4]{\epsilon}$	432.32	6094.92	**8143.68**	12136.16	16063.48	20166.24
Lanczos Algorithm	7724.94	6672.18	13832.8	21457.2	29447.52	39152.46

Wall-Time Performance Data with $A \in \mathbb{R}^{500\times 500}$						
Error Threshold (ϵ)	10e-2	10e-3	10e-4	10e-5	10e-6	10e-7
Vanilla Power Method	**307.9**	27496.5	35858.14	49056.32	66559.52	70891.42
Power+M, $\beta = \lambda_2^2/4$	337.26	**12912.76**	**20098.66**	24924.28	31590.88	49575.94
DMPower, $\rho = \epsilon$	1423.64	15366.22	20172.32	**24663.36**	**30818.32**	38748.8
DMPower, $\rho = \sqrt{\epsilon}$	2094.86	15868.62	20382.9	27008.76	30883.86	38207.12
DMPower, $\rho = \sqrt[3]{\epsilon}$	1490.0	15209.92	22283.78	26917.66	31475.8	**36617.02**
DMPower, $\rho = \sqrt[4]{\epsilon}$	1623.94	17126.1	20743.88	25229.04	32106.58	39925.98
Lanczos Algorithm	58438.68	58700.02	80682.24	91623.18	106078.32	126685.16

Table 6: Averaged log error, batch size = 500, with performance measured by $\log_{10}\left(1 - \frac{||X^\top q_K||}{||X^\top v_1||}\right)$. DMStream registers much better accuracy than Oja's algorithm and emulates the performance of optimal Mini-Batch Power+M. We notice that accuracy does not improve as we increase the number of epochs, which is commonly observed for streaming algorithm running with small batch sizes. We demonstrate in Figure 4 that increasing batch size results in improved accuracy.

Log Error Performance Data with batch size = 500					
Epochs	10	20	30	40	50
DMStream, $\rho = 0.1$	-1.900	-1.894	-1.983	-1.969	-1.959
DMStream, $\rho = 0.01$	-1.992	-1.908	-1.882	-1.949	-1.905
DMSteam, $\rho = 0.001$	-1.929	-1.9585	-1.936	-1.963	-1.973
Oja $\eta_t = 3/t$	-0.588	-0.599	-0.625	-0.565	-0.549
Oja $\eta_t = 9/t$	-0.629	-0.592	-0.599	-0.531	-0.638
Oja $\eta_t = 27/t$	-0.680	-0.668	-0.584	-0.599	-0.647
Oja $\eta_t = 81/t$	-0.590	-0.676	-0.527	-0.604	-0.665
Mini-Batch Power+M (optimal $\beta = \lambda_2^2/4$)	-1.881	-1.860	-1.964	-1.996	-1.966

Appendix F. Precision Bounds for Momentum

In our main result Theorem 4, we assume that $\beta \in [\lambda_2^2/4, \lambda_1^2/4)$, while it is possible that we select $\beta < \lambda_2^2/4$. This poses no problem: as long as β is near $\lambda_2^2/4$ we will experience similar acceleration effects. We state the more general version of Theorem 2 which reflects this fact and add our own modified condition, $\Delta_{2,3} := \lambda_2 - \lambda_3 > 0$.

Theorem 23 (Generalized Convergence of Power+M (De Sa et al., 2018)) *Given a PSD matrix $A \in \mathbb{R}^{d\times d}$ with eigenvalues $\lambda_1 > \lambda_2 > \lambda_3 \ldots \lambda_d \geq 0$ with associated orthonormal eigenvectors $v_1, v_2, \ldots, v_d$, for a unit $q_0 \in \mathbb{R}^n$ non-orthogonal to v_1, running Power+M with $\beta \leq \lambda_1$ results in q_k with*

$$\sin^2(\theta_k) = 1 - (q_k^\top v_1)^2 \leq \frac{1}{|q_0^\top v_1|^2} \cdot \begin{cases} 4\left(\frac{2\sqrt{\beta}}{\lambda_1 + \sqrt{\lambda_1^2 - 4\beta}}\right)^{2k}, & \lambda_2 < 2\sqrt{\beta} \\ \left(\frac{\lambda_2 + \sqrt{\lambda_2^2 - 4\beta}}{\lambda_1 + \sqrt{\lambda_1^2 - 4\beta}}\right)^{2k}, & \lambda_2 \geq 2\sqrt{\beta} \end{cases} \tag{82}$$

Ultimately then, our β can land on either side of $\frac{\lambda_2^2}{4}$ and we we will experience acceleration as long as we are "close." The next result establishes tolerance for our estimations $\widehat{\lambda}_2$ of λ_2.

Proposition 24 (Proposition 6 re-stated) *The momentum phase of DMPower set with momentum coefficient $\beta = \widehat{\lambda}_2^2/4 = \mu_J^2/4$ converges if and only if*

$$|\lambda_2 - \widehat{\lambda}_2| \leq \Delta_{1,2}. \tag{83}$$

Proof If $|\lambda_2 - \widehat{\lambda}_2| = |\lambda_2 - \mu_J| \leq \rho = \Delta_{1,2} = |\lambda_1 - \lambda_2|$, then

$$|\lambda_2 - \mu_J| \cdot |\lambda_2 + \mu_J| < |\lambda_1 - \lambda_2| \cdot |\lambda_1 + \lambda_2| \tag{84}$$

$$\Rightarrow |\lambda_1^2 - \mu_J^2| < |\lambda_1^2 - \lambda_2^2| \tag{85}$$

$$\Rightarrow \frac{1}{4}|\lambda_1^2 - \mu_J^2| < \frac{1}{4}|\lambda_1^2 - \lambda_2^2| \tag{86}$$

The first line of our inequality follows from the fact that $\mu_J > 0$ since A is PSD, therefore implicitly we have that $0 < \mu_J < \lambda_1$, implying $|\lambda_2 + \mu_J| < |\lambda_2 + \lambda_1|$. This shows we satisfy the constraint. To show that this is necessary and sufficient, we consider the case where $\rho > \Delta_{1,2}$. This allows for possible selection of $\mu_J > \lambda_1$, in which case we have that $\frac{1}{4}|{\lambda_2}^2 - \mu_k^2| > \frac{1}{4}|\lambda_1^2 - \lambda_2^2|$, which is outside of our guaranteed interval for convergence. ■

Appendix G. Data Matrix Generation for Non-Streaming Experiments

For every experiment, we ran variations of the power method on a random covariance matrix A with a fixed spectrum. We constructed such matrices using a synthetic singular value decomposition (SVD). Specifically, we begin with a diagonal $d \times d$ matrix $\Sigma = \text{diag}\{1, \sqrt{\lambda_2}, \sqrt{\lambda_3}, \sqrt{\lambda_4}, \ldots, \sqrt{\lambda_d}\}$. Notice that by default we set $\lambda_1 = 1$. In practice, we set $\sqrt{\lambda_3} = \sqrt{\lambda_4} = \cdots = \sqrt{\lambda_d}$ for simplicity. We then drew two random orthogonal matrices $U \in \mathbb{R}^{1000 \times d}$ and $V \in \mathbb{R}^{d \times d}$ from the Haar distribution. We then form the data matrix $X = dU\Sigma V^\top \in \mathbb{R}^{1000 \times d}$, from which we acquire our covariance matrix $A = \frac{1}{1000} XX^\top$ with spectrum $\text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_d\}$. For every run in our non-streaming experiments, we would generate a new covariance matrix with our desired spectrum using this construction.

Appendix H. Spectral Clustering & Experimental Data

We first describe provide and discuss the algorithm used for deflation-based power iteration clustering (Thang et al., 2013). We then provide the results of our spectral clustering experiments. We note that PowerIteration in step 3 of Algorithm 5 may be replaced with DMPower, Power+M, or any other variant. To recover the second eigenvector, DPIC uses a Schur complement deflation (Saad, 2011) on W once the leading eigenvector is computed. Briefly, deflating a matrix shifts its spectrum so that the second leading eigenvalue/eigenvector now becomes the leading eigenvalue/eigenvector, upon which a power iteration may be used again. Successive deflations allows one to recover as many eigenvectors as desired, although numerical instability is increased with each deflation.

Algorithm 5 Deflation-based Power Iteration Clustering (DPIC)

Require: normalized affinity matrix $W \in \mathbb{R}^{d \times d}$

$W_0 = W$

for $i = 1, 2$ **do** ▷ Top-2 component recovery

  $v_i = \text{PowerIteration}(W_{i-1})$ ▷ Return the leading eigenvector

  $W_i = W_{i-1} - \frac{W_{i-1} v_i v_i^\top W_{i-1}}{v_i^\top W_{i-1} v_i}$ ▷ Schur complement deflation

  $i \leftarrow i + 1$

Use k-means on eigenvector approximates v_1, v_2.

return C_1, C_2

Table 7: Proportion of data points correctly classified by spectral clustering combined with k-means. As the error-threshold (and therefore, accuracy) of the eigenvector output is tightened, accuracy improves, eventually achieving perfect classification. See Figure 5 for a visual depiction of data separation using DMPower ($\rho = \sqrt[3]{\epsilon}$). We further note that although the vanilla power method achieves perfect classification at lower error-thresholds, it is at the cost of significantly more iterations as indicated in Table 8.

Spectral Clustering Accuracy Data on Concentric Circles Dataset					
Error Threshold (ϵ)	10e-2	10e-4	10e-6	10e-8	10e-10
Vanilla Power Method	0.6953	0.7854	1.0000	1.0000	1.0000
Power+M, $\beta = \lambda_2^2/4$	0.6679	0.7158	0.7810	0.9886	1.0000
DMPower, $\rho = \epsilon$	0.6997	0.7215	0.8191	0.9869	1.0000
DMPower, $\rho = \sqrt{\epsilon}$	0.6908	0.6927	0.7779	0.9881	1.0000
DMPower, $\rho = \sqrt[3]{\epsilon}$	0.6770	0.7056	0.7657	0.9872	1.0000

Spectral Clustering Accuracy Data on Half Moons Dataset					
Error Threshold (ϵ)	10e-2	10e-4	10e-6	10e-8	10e-10
Vanilla Power Method	0.6164	0.7793	0.9808	1.0000	1.0000
Power+M, $\beta = \lambda_2^2/4$	0.5966	0.6362	0.7560	1.0000	1.0000
DMPower, $\rho = \epsilon$	0.6196	0.6616	0.8185	0.9696	1.0000
DMPower, $\rho = \sqrt{\epsilon}$	0.6132	0.6112	0.7368	1.0000	1.0000
DMPower, $\rho = \sqrt[3]{\epsilon}$	0.6084	0.6283	0.8054	1.0000	1.0000

Table 8: Iteration complexity required to recover principal components. DMPower requires significantly fewer iterations for eigenvector recovery when compared to the vanilla power method, and closely mimics the performance of Power+M with $\beta = \lambda_2^2/4$.

Spectral Clustering Iterations Data on Concentric Circles Dataset					
Error Threshold (ϵ)	10e-2	10e-4	10e-6	10e-8	10e-10
Vanilla Power Method	7.72	59.16	605.36	1457.56	2426.36
Power+M, $\beta = \lambda_2^2/4$	**3.00**	**9.24**	85.12	**642.76**	**1321.00**
DMPower, $\rho = \epsilon$	6.00	11.16	92.00	751.68	1449.08
DMPower, $\rho = \sqrt{\epsilon}$	5.00	11.20	**77.68**	694.20	1452.84
DMPower, $\rho = \sqrt[3]{\epsilon}$	5.00	10.64	78.48	642.92	1505.88

Spectral Clustering Iterations Data on Half Moons Dataset					
Error Threshold (ϵ)	10e-2	10e-4	10e-6	10e-8	10e-10
Vanilla Power Method	7.12	58.72	506.00	1061.68	1929.52
Power+M, $\beta = \lambda_2^2/4$	**3.00**	**11.20**	**84.84**	**601.12**	1226.72
DMPower, $\rho = \epsilon$	6.00	12.08	89.16	629.96	1192.68
DMPower, $\rho = \sqrt{\epsilon}$	5.00	11.56	94.88	661.20	1225.08
DMPower, $\rho = \sqrt[3]{\epsilon}$	5.00	11.52	92.64	711.00	**1180.72**