Journal of Computational and Applied Mathematics 308 (2016) 98–116

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Matrix-free Krylov iteration for implicit convolution of numerically low-rank data

Alex Breuer [a,*], Andrew Lumsdaine [b]

[a] Army Research Laboratory, United States

[b] Indiana University, United States

ARTICLE INFO

Article history:
Received 26 June 2015
Received in revised form 10 December 2015

Keywords:
Low-rank approximation
Convolution
SVD
PCA

ABSTRACT

Evaluating the response of a linear shift-invariant system is a problem that occurs frequently in a wide variety of science and engineering problems. Calculating the system response via a convolution may be done efficiently with Fourier transforms. When one must compute the response of one system to m input signals, or the response of m systems to one signal, it may be the case that one may approximate all system responses without having to compute all m Fourier transforms. This can lead to substantial computational savings. Rather than process each point individually, one may only process basis vectors that span the output data space. However, to get a low-error approximation, it is necessary that the output vectors have low numerical rank if they were assembled into a matrix. We develop theory that shows how the singular value decay of a matrix ΦA that is a product of a convolution operator Φ and an arbitrary matrix A depends in a linear fashion on the singular value decays of Φ and A. We propose gap-rank, a measure of the relative numerical rank of a matrix. We show that convolution cannot destroy the numerical low-rank-ness of ΦA data with only modest assumptions. We then develop a new method that exploits low-rank problems with block Golub–Kahan iteration in a Krylov subspace to approximate the low-rank problem. Our method can exploit parallelism in both the individual convolutions and the linear algebra operations in the block Golub–Kahan algorithm. We present numerical examples from signal and image processing that show the low error and scalability of our method.

Published by Elsevier B.V.

1. Introduction

Linear shift-invariant (LSI) systems find application in numerous scientific and engineering domains, from computational imaging to geophysics to control theory. The widespread use of LSI systems is owed in part to one particular advantage: the response of an LSI system to an input is defined as the convolution of the impulse response of the LSI with that input, and convolution may be computed efficiently.

More formally, if $f[t]$ is the discrete impulse response of the LSI system and $a[t]$ is the input, then the system response $c[t]$ is

$$c[t] = (f * a)[t] \tag{1}$$

where $*$ is the convolution operator as defined in (3).

* Corresponding author.
E-mail addresses: alexander.m.breuer.civ@mail.mil (A. Breuer), lums@cs.indiana.edu (A. Lumsdaine).

http://dx.doi.org/10.1016/j.cam.2016.05.005
0377-0427/Published by Elsevier B.V.

Convolution is equivalent to point-wise multiplication in the frequency domain. Thus, (1) becomes

$$\tilde{c}(\omega) = \tilde{f}(\omega)\tilde{a}(\omega)$$

where $\tilde{c}(\omega)$ is the Discrete Fourier transform (DFT) of $c[t]$, $\tilde{f}(\omega)$ is the DFT of $f[t]$, and $\tilde{a}(\omega)$ is the DFT of $a[t]$. The Discrete Fourier transform may be computed for a signal with n samples with complexity $O(n \log n)$, and calculating the system response of one LSI system to one input is an easily-solved problem.

Nevertheless, it may be the case that one is presented with m such signals $a[t]$, or m LSI systems. In either case, one may compute m DFTs to get all system responses. When either the input signals or the LSI systems exhibit a significant degree of linear dependence, it is possible to approximate all m system responses with far fewer than m DFT operations. Input signals with significant linear dependence may result from a process with few degrees of freedom and/or repeated initial conditions. For example, data from experiments which repeatedly load a physical structure with near-identical loadings may be expected to exhibit a high degree of linear dependence. We call data which exhibits a large degree of linear dependence a good candidate for low-rank approximation. By that, we mean all m signals may be approximated using linear combinations of $k \ll m$ basis vectors.

1.1. Low-rank matrix approximation

When data is presented as a series of vectors $a_i \in \mathbb{C}^n$ for $1 \leq i \leq m$ and the vectors a_i exhibit a large degree of linear dependence, the vectors a_i may be approximated with vectors $\hat{a}_i^{(k)}$ that belong to a space $\mathcal{S}$ that itself has dimension $k \ll m$. The goodness of the approximate vectors $\hat{a}_i^{(k)}$ may be measured with a variety of metrics, but the sum-of-squares difference is a natural measure that has intuitive geometric interpretations. When normalized by the sum of squared norms of all vectors, it gives the percentage of "energy" of the original a_i that is represented by the approximate $\hat{a}_i^{(k)}$. The normalized sum-of-squares approximation error of $\epsilon_A^{(k)}$ is just

$$\epsilon_A^{(k)} = \sqrt{\frac{\sum_{i=1}^{m} \|\hat{a}_i^{(k)} - a_i\|^2}{\sum_{i=1}^{m} \|a_i\|^2}}.$$

In matrix form, this becomes $A := [a_1\ a_2\ \cdots\ a_m]$, $\hat{A}^{(k)} := [\hat{a}_1^{(k)}\ \hat{a}_2^{(k)}\ \cdots\ \hat{a}_m^{(k)}]$, and $\epsilon_A^{(k)} = \|\hat{A}^{(k)} - A\|_F^2 / \|A\|_F^2$ — the normalized Frobenius norm error. We seek an approximation that sets $\epsilon_A^{(k)}$ as small as possible.

The vectors a_i may be approximated with projections derived from truncating the singular value decomposition (SVD) of A. These SVD-based approaches are already common in many signal processing domains [1–4]. When the data are nearly low-rank — that is, they are *numerically* low-rank, singular values of A decay rapidly. This property may be exploited to produce low-rank approximations that have small error against the original, high-dimensional data [5,6,3,7,8]. Computation of singular triplets may be costly, especially as k grows [9,10]. Less-costly low-rank approximation alternatives to the SVD are Krylov subspace methods [11–15]. Substantial cost savings may be realized with only small sacrifices in Frobenius norm approximation error.

The linear dependence of data a_i is not immediately useful for dimension reduction when the data a_i are subjected to a subsequent convolution step: $c_i[t] = (f * a_i)[t]$. Note that convolution may be expressed in matrix form as $C = \Phi A$, where Φ is a (possibly complex) convolution matrix. Cyclic convolution can be expressed as a matrix [16], and non-cyclic convolution may be obtained from cyclic convolution with appropriate zero-padding of the signals.

1.2. Principal difficulties of low-rank approximation of C

We want to approximate C with a low-rank matrix, but there are two principal difficulties we face:

- we may only know that A is a good candidate for low-rank approximation — we do not know that C is — and
- C is defined in terms of its factors A and Φ, but we need to avoid directly forming C.

Addressing these two challenges is the chief contribution of this paper.

Our first principal contribution is that we provide mechanisms for determining if C is amenable to low-rank approximation via its spectrum. The spectrum of the matrix C depends on both A and Φ. Even if A is a good candidate for low-rank approximation, it is not clear that C will be. It is easy to construct a Φ such that C has a flat spectrum and is a poor candidate for low-rank approximation — e.g. set Φ to be the Moore–Penrose pseudoinverse [17, Def. 3.2] of A. Nevertheless, we will show in Section 2.2 that the matrix C is a good candidate for low-rank approximation with only modest restrictions on the spectrum of Φ.

Our second principal contribution is a modification of the Golub–Kahan algorithm that allows for low-rank approximation of C without ever directly forming either C or Φ. This method avoids redundant DFT operations. The Golub–Kahan algorithm, a Krylov subspace method, only interacts with its input matrix through matrix–vector products,

so it is only necessary that one may compute Cx and C^*x. In practice, one almost never forms the convolution matrix Φ explicitly; rather, the fast Fourier transform algorithm [18] is used to compute DFT operations, and implicitly evaluate Φx for some x [17]. One may compute Cx and C^*x using explicit matrix–vector products on A and using the DFT to compute matrix–vector products on Φ. Thus, one may approximate C in a Krylov subspace without ever explicitly forming C.

1.3. Notation

We introduce some notation that we will use in the proceeding discussion. First, we formally define a *Krylov subspace*.

Definition 1. Given a square matrix $G \in \mathbb{C}^{n\times n}$ and a matrix $X_0 \in \mathbb{C}^{n\times s}$ (hereafter called a "block vector"), the **Krylov subspace** of length i defined by G and X_0 is given by

$$\mathcal{K}_i(G, X_0) := \operatorname{span}\{X_0, GX_0, G^2X_0, \ldots, G^{i-1}X_0\}.$$

For rectangular matrices C, we may set $G = CC^*$ or $G_T = C^*C$, and we consider the two Krylov subspaces $\mathcal{K}_i(G, X_0)$ and $\mathcal{K}_i(G_T, C^*X_0)$.

As the bulk of our analysis revolves around singular values, we write $\sigma_i(A)$ to denote the ith singular value of the matrix A, and order singular values as $\sigma_1(A) \geq \sigma_2(A) \geq \cdots \geq \sigma_{\text{inf}}(A)$. We write $\sigma(A)$, without a subscript, to denote the entire sequence of singular values, ordered as $\sigma(A) := [\sigma_1(A)\ \sigma_2(A)\ \ldots\ \sigma_{\text{inf}}(A)]$. The notation $\sigma^{\uparrow}(A)$ reverses the ordering of the sequence: $\sigma^{\uparrow}(A) := [\sigma_{\text{inf}}(A)\ \ldots\ \sigma_2(A)\ \sigma_1(A)]$. We assume that our matrices are finite-dimensional, though low-rank approximation could be applied to any bounded operator, in theory. Therefore, A is a $n \times m$ matrix, Φ is $n \times n$, and both have only finitely-many singular values.

We will use majorization [19] to compare two sequences of singular values; we define majorization.

Definition 2. Let $\boldsymbol{x} = x_1 \geq x_2 \geq \cdots \geq x_n$ and $\boldsymbol{y} = y_1 \geq y_2 \geq \cdots \geq y_n$ be two sequences of real numbers. Then we say that $\boldsymbol{x}$ **majorizes** $\boldsymbol{y}$ — written as $\boldsymbol{x} \prec \boldsymbol{y}$ — if

$$\sum_{i=1}^{k} x_i \leq \sum_{i=1}^{k} y_i \tag{2}$$

for all $k < n$ and

$$\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i.$$

We say that $\boldsymbol{x}$ **weakly majorizes** $\boldsymbol{y}$ — written as $\boldsymbol{x} \prec_w \boldsymbol{y}$ — if (2) holds for $k \leq n$.

We write

$$\sigma(A) - \sigma(B) := [(\sigma_1(A) - \sigma_1(B))\ (\sigma_2(A) - \sigma_2(B))\ \ldots\ (\sigma_{\text{inf}}(A) - \sigma_{\text{inf}}(B))].$$

We define the product $\sigma(A)\sigma(B)$ as

$$\sigma(A)\sigma(B) := [(\sigma_1(A)\sigma_1(B))\ (\sigma_2(A)\sigma_2(B))\ \ldots\ (\sigma_{\text{inf}}(A)\sigma_{\text{inf}}(B))].$$

The Frobenius norm is defined in terms of singular values $\|A\|_F = \sqrt{\sum_{i=1}^{N} \sigma_i(A)^2}$, where A has N singular values.

To measure the angles between subspaces, we use principal angles (cf. [20, Definition 2.1 and Theorem 2.1]). We write the principal angles between subspaces colspan$\{X\}$ and colspan$\{Y\}$, where X and Y have orthonormal columns, as

$$\theta(X, Y) := [\cos^{-1}(\sigma_1(X^*Y))\ \cos^{-1}(\sigma_2(X^*Y))\ \ldots] := \cos^{-1}(\sigma(X^*Y)).$$

Note that when either X or Y has only one column, then there is only one principal angle.

We write Ψ as the unitary matrix that gives the DFT operation [17, p. 323]. For a time or spatial-domain vector x, we write $\tilde{x} := \Psi x$ to mean its frequency-domain representation.

Finally, we define convolution. We have assumed that all our signals are finite in length; these correspond to columns of A. Abusing notation slightly, we define $a_i[j] = 0$ if $j < 0$ or $a_i[j] > m$; this could be interpreted as zero padding used to obtain non-circular convolution. We then write the convolution of two finitely-supported functions as

$$(f * a_i)[t] = \sum_{j=-m}^{m} f[j]a_i[t - j] \tag{3}$$

for $0 \leq t \leq m$. This denotes the time-domain convolution of f and a_i as used in (1).

2. On the suitability of C for low-rank approximation

We have said that an important contribution of this paper is that we show when C is a good candidate for low-rank approximation given modest assumptions on Φ and A. To see that, we must first characterize more precisely what it means

for a matrix to be a good candidate for low-rank approximation. First, we will see that the best-case normalized Frobenius norm error for an orthogonal projection of dimension k is defined by the first k left singular values. We will examine how the decay of singular values characterizes those matrices that are good candidates for low-rank approximation.

We then present gap-rank: a measure of the decay of singular values. This is our contribution that measures how well some matrix M may be approximated with respect to the normalized sum-of-squares error.

2.1. Decay of singular values and bounding the optimal normalized Frobenius norm error

It is well known the smallest Frobenius norm error $\epsilon^{(k)}$ may be conveniently defined in terms of the SVD of A; in fact, $\epsilon^{(k)} = \sqrt{\sum_{i=k+1}^{\min\{n,m\}} \sigma_i(A)}/\|A\|_F$. It follows that when $\sigma_1(A) \gg \sigma_k(A)$, the best-case normalized Frobenius norm error will be small. This fact may directly characterize those matrices that are good candidates for low-rank approximation: if $\sqrt{(\sum_{i=k+1}^{\min\{n,m\}} \sigma_i(A)^2)/(\sum_{i=1}^{\min\{n,m\}} \sigma_i(A)^2)}$ is tiny for some small $k \ll \min\{n, m\}$, then A is a good candidate for low-rank approximation. This is equivalent to saying that the bulk of the Frobenius norm of A is accounted for by only a small number of singular values.

This characterization is fine for matrices where one has information about all of the singular values of the matrix which we wish to approximate. However, we are presented with only factors of C, and we cannot assume any knowledge of the spectrum of C. We address this by developing the gap-rank measure, which we use to reason about how well C may be approximated with a low-rank matrix. Gap-rank measures the value $(m - k)(\sigma_k(A)/\sigma_1(A))$, which bounds $\epsilon^{(k)} = \sqrt{\sum_{i=k+1}^{\min\{m,n\}} \sigma_i(A)}/\|A\|_F \leq (m - k)(\sigma_k(A)/\sigma_1(A))$ from above.

2.2. The gap-rank measure

Characterizing how well a matrix may be rank-k approximated in terms of only its first k singular values is convenient for the matrix C. We only assume that we know the singular values of the two factors of C. In order to reason about how well $C = \Phi A$ may be approximated in terms of the singular values of A and Φ, we present gap-rank.

Definition 3. Let A be an arbitrary $n \times m$ matrix with singular value decomposition $A = U\Sigma V^*$. Let τ be a real number with $0 \leq \tau \leq 1$. Then we define the **gap-rank**, written gap-rank$\{A, \tau\}$, of A as

$$\text{gap-rank}\{A, \tau\} \coloneqq \begin{cases} \min_i\{\sigma_i(A)/\sigma_1(A) \leq \tau\} & \text{if } \exists \sigma_i(A) \text{ s.t. } \sigma_i(A)/\sigma_1(A) \leq \tau \\ \min\{n, m\} & \text{otherwise.} \end{cases}$$

As is evident by its definition, gap-rank is simply the part of the scaled ratio $(m - k)\sigma_i(A)/\sigma_k(A)$ that depends on the singular values of A. Thus, when the $n \times m$ matrix A has small gap-rank$\{A, \tau\}$ for some tiny τ, A will be a good candidate for low-rank approximation.

As one might expect, the gap-rank of C depends on the gap-ranks of Φ and A. To reason about the gap-rank of C, we present the following theorem. This theorem shows that matrices of the form ΦA, where at least one of A or Φ are good candidates for low-rank approximation, will inherit that property to some degree. When and how C will be a good candidate for low-rank approximation is based on the relationship between $\sigma_1(\Phi A)$, $\sigma_1(A)$ and the whole sequence $\sigma(\Phi)$.

Theorem 1. *Let Φ and A be arbitrary $n \times n$ and $n \times m$ matrices. Pick some τ. Suppose that $\sigma_1(\Phi A) \geq \sigma_1(\Phi)\sigma_p(A)$ for some $p \geq$ gap-rank$\{A, \tau\}$. Then*

$$\text{gap-rank}\{\Phi A, \tau\} \leq 2p - 1.$$

Proof. Pick τ and set

$$p = \operatorname*{argmin}_{i \geq \text{gap-rank}\{A,\tau\}} \{\sigma_1(\Phi A) \geq \sigma_1(\Phi)\sigma_i(A)\}.$$

Such a p is guaranteed to exist due to Theorem 10.30 and 10.1 in [19] which respectively provide that $\sigma(A)\sigma^{\uparrow}(B) \prec_w \sigma(AB)$ and $x \prec_w y \Rightarrow x_1^{\uparrow} \leq y_1^{\uparrow}$.

Let $A = U\Sigma V^*$ be the SVD of A and $\Phi = X\Upsilon Y^*$ be the SVD of Φ. We can partition the SVDs as

$$A = [U_1\ U_2] \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} [V_1^*\ V_2^*]$$

where $\Sigma_1 = \text{diag}(\sigma_1(A), \sigma_2(A), \ldots, \sigma_{p-1}(A))$, and for Φ,

$$\Phi = [X_1\ X_2] \begin{bmatrix} \Upsilon_1 & \\ & \Upsilon_2 \end{bmatrix} [Y_1^*\ Y_2^*]$$

where $\Upsilon_1 = \operatorname{diag}(\sigma_1(\Phi), \sigma_2(\Phi), \ldots, \sigma_{p-1}(\Phi))$. Moreover, we can write the above partitionings as sums: $A = U_1\Sigma_1 V_1^* + U_2\Sigma_2 V_2^*$ and $\Phi = X_1\Upsilon_1 Y_1^* + X_2\Upsilon_2 Y_2^*$.

Let $A_1 = U_1\Sigma_1 V_1^*, A_2 = U_2\Sigma_2 V_2^*, \Phi_1 = X_1\Upsilon_1 Y_1^*$ and $\Phi_2 = X_2\Upsilon_2 Y_2^*$, and then

$$\Phi A = \Phi_1 A_1 + \Phi_1 A_2 + \Phi_2 A_1 + \Phi_2 A_2. \tag{4}$$

Also, notice that $\operatorname{rank}\{A_1\} = \operatorname{rank}\{\Phi_1\} = p - 1$.

From this point, we seek to bound the rank and largest singular values of the matrices in the right-hand side of (4).

First, we have lower bounds for $\sigma_1(\Phi_1 A_1 + \Phi_1 A_2 + \Phi_2 A_1 + \Phi_2 A_2)$ from our assumption on p: we have assumed

$$\sigma_1(\Phi A) \geq \sigma_1(\Phi)\sigma_p(A).$$

Now we seek to bound $\sigma_{2p-1}(\Phi A) = \sigma_{2p-1}(\Phi_1 A_1 + \Phi_1 A_2 + \Phi_2 A_1 + \Phi_2 A_2)$ from above. First, we need to determine the rank of $\Phi_1(A_1 + A_2) + \Phi_2 A_1$. Using the subadditivity of matrix rank and Sylvester's Theorem on rank [19, Theorem 2.6], which gives $\operatorname{rank}\{DE\} \leq \min\{\operatorname{rank}\{D\}, \operatorname{rank}\{E\}\}$, we get

$$\begin{aligned}\operatorname{rank}\{\Phi_1(A_1 + A_2) + \Phi_2 A_1\} &\leq \operatorname{rank}\{\Phi_1(A_1 + A_2)\} + \operatorname{rank}\{\Phi_2 A_1\} \\ &\leq \min\{\operatorname{rank}\{\Phi_1\}, \operatorname{rank}\{A_1 + A_2\}\} + \min\{\operatorname{rank}\{\Phi_2\}, \operatorname{rank}\{A_1\}\} \\ &\leq 2p - 2\end{aligned}$$

since $\operatorname{rank}\{\Phi_1\} = \operatorname{rank}\{A_1\} = p - 1$. Therefore, $\Phi_1 A + \Phi_2 A_1$ has at most $2p - 2$ nonzero singular values, and $\sigma_{2p-1}(\Phi_1 A_1 + \Phi_1 A_2 + \Phi_2 A_1) = 0$.

We can now bound singular value $\sigma_{2p-1}(\Phi A)$. Weyl's inequality [17] on perturbation gives

$$\sigma_i(D) - \sigma_1(E) \leq \sigma_i(D + E) \leq \sigma_i(D) + \sigma_1(E), \tag{5}$$

but we only need the upper bound $\sigma_i(D + E) \leq \sigma_i(D) + \sigma_1(E)$ in this proof. We apply the right-hand side of (5): we set $E := \Phi_2 A_2$ and $D := \Phi_1 A_1 + \Phi_1 A_2 + \Phi_2 A_1 = \Phi_1(A_1 + A_2) + \Phi_2 A_1$. Clearly, $\sigma_1(\Phi_2 A_2) \leq \sigma_p(A)\sigma_p(\Phi)$ due to the construction of Φ_2 and A_2: $\sigma_1(\Phi_2) = \sigma_p(\Phi)$ and $\sigma_1(A_2) = \sigma_p(A)$. Then

$$\begin{aligned}\sigma_{2p-1}(\Phi_1 A_1 + \Phi_1 A_2 + \Phi_2 A_1 + \Phi_2 A_2) &\leq \sigma_{2p-1}(\Phi_1 A_1 + \Phi_1 A_2 + \Phi_2 A_1) + \sigma_1(\Phi_2 A_2) \\ &\leq 0 + \sigma_p(A)\sigma_p(\Phi)\end{aligned}$$

follows. Now,

$$\frac{\sigma_{2p-1}(\Phi_1 A_1 + \Phi_1 A_2 + \Phi_2 A_1 + \Phi_2 A_2)}{\sigma_1(\Phi_1 A_1 + \Phi_1 A_2 + \Phi_2 A_1 + \Phi_2 A_2)} \leq \frac{\sigma_p(A)\sigma_p(\Phi)}{\sigma_1(A)\sigma_p(\Phi)} = \frac{\sigma_p(A)}{\sigma_1(A)}. \tag{6}$$

By the definition of gap-rank$\{A, \tau\}$, the right-hand side of (6) is not greater than τ. Then

$$\frac{\sigma_{2p-1}(\Phi A)}{\sigma_1(\Phi A)} = \frac{\sigma_{2p-1}(\Phi_1 A_1 + \Phi_1 A_2 + \Phi_2 A_1 + \Phi_2 A_2)}{\sigma_1(\Phi_1 A_1 + \Phi_1 A_2 + \Phi_2 A_1 + \Phi_2 A_2)} \leq \tau,$$

which completes the proof. □

We point out that Theorem 1 is an upper bound on gap-rank$\{\Phi A, \tau\}$. The transformation of A with Φ to form C may result in C having a smaller gap-rank than either A or Φ. The following theorem addresses the $\hat{\tau}$ that satisfies gap-rank$\{\Phi A, \hat{\tau}\} = p$ when $\sigma_1(\Phi)\sigma_p(A) \leq \sigma_1(\Phi A)$; this $\hat{\tau}$ may have $\hat{\tau} \ll \tau$ for the τ that satisfies gap-rank$\{A, \tau\} = p$.

Theorem 2. *Suppose there exists some p such that $\sigma_1(\Phi)\sigma_p(A) \geq \sigma_1(\Phi A)$. Further, let $\hat{\tau} = \sigma_p(\Phi)\sigma_1(A)/\sigma_1(\Phi)\sigma_p(A)$. Then*

$$\text{gap-rank}\{\Phi A, \hat{\tau}\} \leq p.$$

Proof. Beginning with (4), we get $\Phi_1 A = \Phi_1(A_1 + A_2)$. Applying the right-hand side of Weyl's inequality (5) with $D := \Phi A_1$ and $E := \Phi A_2$ gives

$$\begin{aligned}\sigma_p(\Phi A_1 + \Phi A_2) &\leq \sigma_p(\Phi A_1) + \sigma_1(\Phi A_2) \\ &\leq 0 + \sigma_1(\Phi)\sigma_p(A)\end{aligned}$$

as $\operatorname{rank}\{\Phi A_1\} = \min\{\operatorname{rank}\{\Phi\}, \operatorname{rank}\{A_1\}\} \leq p - 1 \Rightarrow \sigma_p(\Phi A_1) = 0$ and $\sigma_1(\Phi A_2) \leq \sigma_1(\Phi)\sigma_1(A_2) = \sigma_1(\Phi)\sigma_p(A)$. Since we supposed that $\sigma_1(\Phi A) \geq \sigma_1(\Phi)\sigma_p(A)$,

$$\frac{\sigma_p(\Phi A)}{\sigma_1(\Phi A)} \leq \frac{\sigma_p(\Phi)\sigma_1(A)}{\sigma_1(\Phi)\sigma_p(A)} = \hat{\tau}$$

follows as desired. □

Remark 1. We note that one may choose p such that it satisfies $\sigma_p(\Phi)\sigma_1(A) \leq \sigma_1(\Phi A)$. Then the right-hand side of (6) becomes $\sigma_p(\Phi)/\sigma_1(\Phi)$, and one may bound gap-rank$\{\Phi A, \tau\}$ with gap-rank$\{\Phi, \tau\}$. This also applies to Theorem 2, where $\hat{\tau}$ becomes $\hat{\tau} := \sigma_1(\Phi)\sigma_p(A)/\sigma_p(\Phi)\sigma_1(A)$. So for both Theorems 1 and 2, the order of Φ and A can be reversed in the expressions without loss of generality.

The spectrum of ΦA may decay faster than those of Φ or A. This is summarized in the following proposition.

Proposition 1. *Let A and Φ be arbitrary $n \times n$ complex matrices; when A is not $n \times n$, then it can be zero-padded without loss of generality. Let $A = U\Sigma V^*$ be the SVD of A and $\Phi = X\Upsilon Y^*$ be the SVD of Φ. Let $\mathfrak{s} = \{|\sigma_i(A)\sigma_i(\Phi)(y_i^* u_i)| \text{ for } 1 \leq i \leq n\}$. Then*

$$\mathfrak{s} \prec_w \sigma(\Phi A).$$

Proof. Let $A = U\Sigma V^*$ be the SVD of A and $\Phi = X\Upsilon Y^*$ be the SVD of Φ. Since colspan$\{X\}$ = colspan$\{\Phi\}$ and colspan$\{U\}$ = colspan$\{A\}$,

$$\sigma(\Phi A) = \sigma(X^*\Phi A V) = \sigma(\Upsilon Y^* U\Sigma). \tag{7}$$

Let the operator $d(A) := \{a_{ii} \mid 1 \leq i \leq n\}$ return a sequence that is simply the diagonal of the input matrix A. Note that $d(\Upsilon Y^* U\Sigma) = \{\upsilon_i \sigma_i (y_i^* u_i) \mid 1 \leq i \leq n\}$, where y_i and u_i are columns of Y and U, respectively. By Theorem 10.19 in [19], $|d(A)| \prec_w \sigma(A)$, and noting $\mathfrak{s} = |d(\Upsilon Y^* U\Sigma)|$ and $\sigma(\Phi A) = \sigma(\Upsilon Y^* U\Sigma)$ yields $\mathfrak{s} \prec_w \sigma(\Phi A)$ as desired. □

Together, Theorem 1 and Proposition 1 show how the singular value decay of ΦA depends on the interaction of Φ and A. Proposition 1 shows that when the left singular vectors of A are the same as the right singular vectors of Φ, then $|y_i^* u_i| = 1$ and $\sigma(C) = \sigma(\Phi)\sigma(A)$. Therefore, when the left and right singular vector spaces of Φ and A have $|y_i^* u_i|$ close to 1, then singular values of C will decay faster than those of either Φ or A. Theorem 1 shows how the gap-rank of ΦA depends on how much larger $\sigma_1(\Phi A)$ is than $\sigma_1(A)$. When $\sigma_1(\Phi A) \gg \sigma_1(A)$ and singular values of both Φ and A decay rapidly, then the parameter p from Theorem 1 will not be much larger than gap-rank$\{A, \tau\}$.

We present an example that illustrates the application of both Theorem 1 and Proposition 1 to a straightforward problem; Theorem 2 will be demonstrated in Section 4.3.2.

Example 1. Let $A = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{200})$ be a 200×200 diagonal matrix with singular values σ_i drawn from a Pareto distribution with shape factor $\alpha = 0.3$. We examine the gap-ranks of products ΦA for three different Φ to show how Theorem 1 and Proposition 1 predict decay of the spectrum of ΦA based on the p from Theorem 1 and $u_i^* y_i$ from Proposition 1. These three Φ are:

- $\Phi_{\text{diag}} = \text{diag}\{\upsilon_1, \upsilon_2, \ldots, \upsilon_{\text{inf}}\}$ where υ_i are from the same distribution as singular values of A,
- a random matrix $\Phi_{\text{random}} = \phi_{ij}$ with ϕ_{ij} drawn from a Normal distribution with $\mu = 2$ and standard deviation $\sigma = 0.1$, and
- Φ is the inverse of A, i.e. $\Phi_{A^{-1}} = A^{-1}$; therefore $\Phi_{A^{-1}} A$ is the identity matrix.

We first examine the amenability of each matrix to low-rank approximation. The singular values of A and these three Φ are shown in Fig. 1. The gap-ranks are shown in Fig. 2. Note that A has rapid decay of singular values, which implies that the ratio $\sigma_k(A)/\sigma_1(A)$ can be made tiny for a small k. This translates to a small gap-rank even for tiny values of τ (τ is on the x-axis of the plots in Fig. 2): gap-rank$\{A, 10^{-6}\} = 16$. The matrices $\Phi_{A^{-1}}$ and Φ_{diag} also have rapidly-decaying singular values which makes them good candidates for low-rank approximation.

However, a central problem of attempting to approximate the product of two matrices is that $C = \Phi A$ may not be a good candidate even when both A and Φ are. It is evident in Fig. 2 that $\Phi_{A^{-1}} A = I$ is not a good candidate for low-rank approximation. Indeed, the identity matrix, which has all singular values equal, is the worst-case for low-rank approximation error. However, both $\Phi_{\text{diag}} A$ and Φ_{random} are good candidates. To see how Theorem 1 and Proposition 1 predict this behavior, we show the values of $\sigma_1(\Phi A)$, p from Theorem 1 and $y_1^* u_1$ from Proposition 1 in Table 1. We point out that Theorem 2 does not show any additional tightness, as $\sigma_1(\Phi)\sigma_3(A)/\sigma_3(\Phi)\sigma_1(A) > 1$ in all 3 cases. We will show examples in Section 4.3.2 which demonstrate the utility of Theorem 2. Note that the value of $\sigma_1(A) = 6.22 \times 10^9$.

We consider $\Phi_{\text{diag}} A$. Since $|y_i^* u_i| = 1$ for all i, $\Phi_{\text{diag}} A$, we may apply Proposition 1 and see that singular values of $\Phi_{\text{diag}} A$ decay at least as fast as the product of two Pareto distributed variables.

We consider $\Phi_{\text{random}} A$. For Φ_{random}, $\sigma_i(\Phi_{\text{random}}) \leq 10$ for $i \geq 2$; this and $\sigma_1(\Phi_{\text{random}} A)/\sigma_1(A) \geq 10$ imply that p from Theorem 1 is not greater than 3. Then the gap-rank of $\Phi_{\text{random}} A$ is not greater than the gap-rank of A when gap-rank$\{A, \tau\} \geq 5$.

We consider $\Phi_{A^{-1}} A$: for $\Phi_{A^{-1}}$, $\sigma(\Phi_{A^{-1}} A) = 1 = \sigma_1(A)\sigma_{\text{inf}}(\Phi_{A^{-1}})$, so $p = 200$.

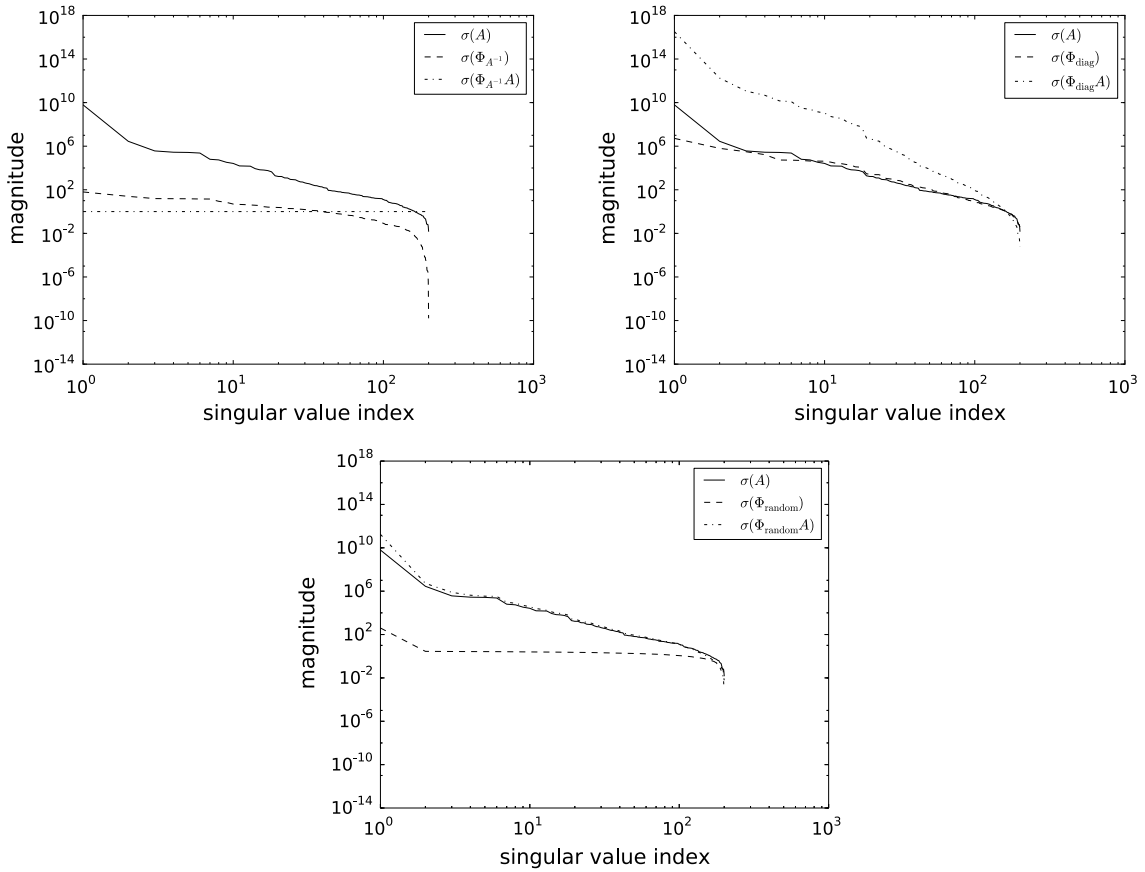

Fig. 1. Spectra for $\Phi_{A^{-1}}A$ (top left), $\Phi_{\text{diag}}A$ (top right) and $\Phi_{\text{random}}A$ (bottom). Note the rapid decay of singular values of A. This suggests that A is a good candidate for low-rank approximation. Some of the Φ also have rapid decay of singular values and are good candidates, such as $\Phi_{A^{-1}}$ and Φ_{diag}.

Table 1
Singular values of ΦA, p from Theorem 1 and $\langle u_1, y_1 \rangle$ from Proposition 1.

Matrix	$\sigma_1(\Phi A)$	min p such that $\sigma_1(\Phi A) \geq \sigma_1(A)\sigma_p(\Phi)$	$y_1^* u_1$
$\Phi_{\text{diag}}A$	3.24×10^{16}	2	1
$\Phi_{\text{random}}A$	1.75×10^{11}	2	−0.07
$\Phi_{\text{A}^{-1}}A$	1	200	0

3. Krylov subspace projections for low-rank approximation

As demonstrated in the previous section, singular vector spaces are optimal for minimizing the Frobenius norm error of a rank-k approximation of a matrix C. When the matrix C has a rapidly-decaying spectrum, the normalized error $\epsilon_A^{(k)}$ will shrink rapidly with only modest k. For such matrices, it has been observed that projection through a short Krylov subspace produces sum-of-squares error not much larger than the optimal singular vector projection [11,15]. The principal advantage of using a short Krylov subspace instead of a singular vector space is that the Krylov subspace will be substantially less expensive to generate, c.f. [13].

The Golub–Kahan algorithm [21] (identical to the Lanczos bidiagonalization in [11]) can generate orthonormal bases Q_i and P_i for the dual Krylov subspaces $\mathcal{K}_i(G, x_0)$ and $\mathcal{K}_i(G_T, Cx_0)$ without using a full Gram–Schmidt orthonormalization process. The complexity of generating a basis for a Krylov subspace is therefore not quadratic. Moreover, the Golub–Kahan algorithm can generate the projection $Q_i Q_i^T A$ as it generates the bases Q_i and P_i. We propose using the block version [22] of the Golub–Kahan algorithm; block Krylov subspaces may present enough work to amortize parallel overhead compared to single-vector methods. This is important when parallel implementations of BLAS or LAPACK are available, such as ATLAS [23] or GotoBLAS [24].

Our matrix C is available only in factored form as ΦA, and we have noted that C may be dense even when A is sparse. The Golub–Kahan algorithm only interacts with C via matrix–vector products, so we may use the factors of C directly rather than forming C explicitly and incurring fill-in. The matrix Φ is typically formed matrix-free via 2 DFTs and point-wise multiplication. Thus, at most 4 DFTs and 2 sparse matrix–vector products are necessary per iteration of the Golub–Kahan

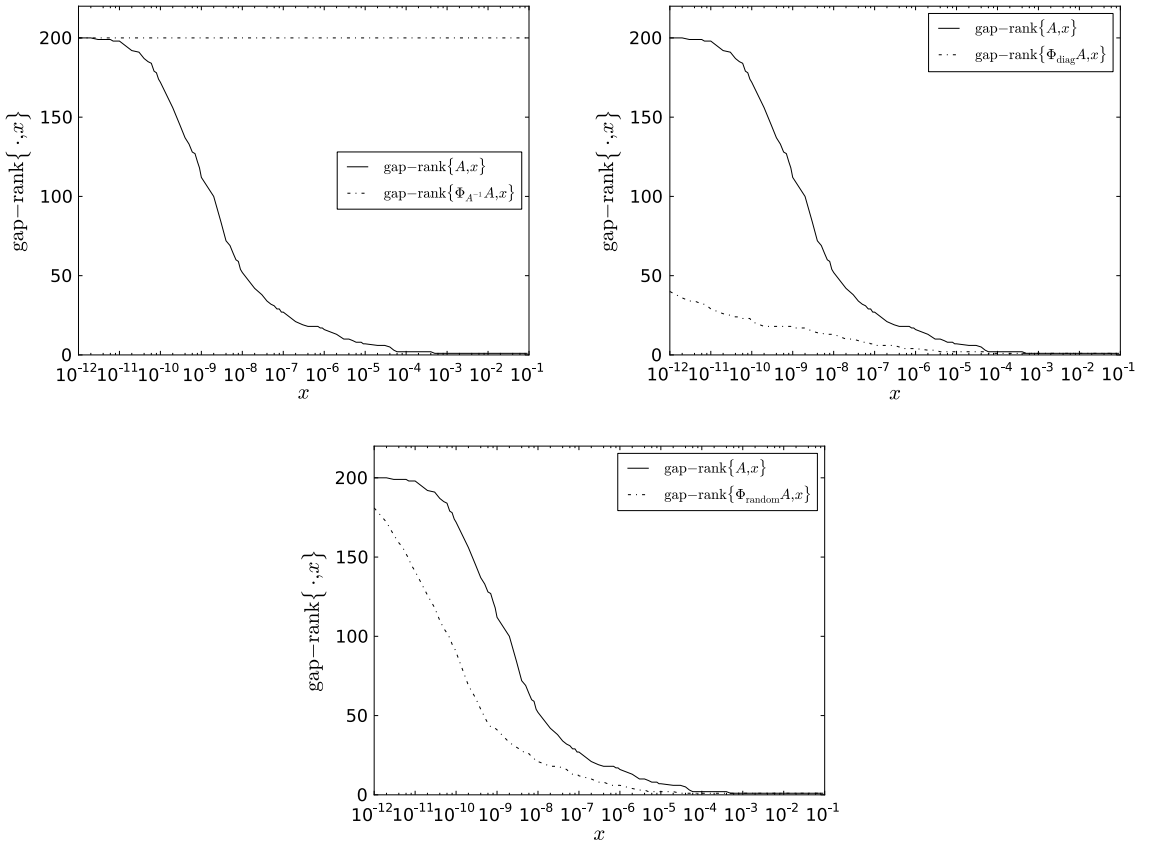

Fig. 2. The gap-rank for $\Phi_{A^{-1}}A$ (top left), $\Phi_{\text{diag}}A$ (top right) and $\Phi_{\text{random}}A$ (bottom). The X axes in these plots are the τ parameter from gap-rank and the Y axes are the values of the gap rank, so a smaller number indicates the more of the Frobenius norm is concentrated in a small number of singular values. Proposition 1 shows that $\Phi_{\text{diag}}A$ will have a spectrum of $\sigma(\Phi)\sigma(A)$, while the relatively flat spectrum of Φ_{random} allows p from Theorem 1 to be small, which guarantees that gap-rank$\{\Phi_{\text{random}}A, \tau\} \leq$ gap-rank$\{A, \tau\}$ when gap-rank$\{A, \tau\} \geq 2$. For $\Phi_{A^{-1}}A$, $\sigma_1(\Phi_{A^{-1}}A) = 1$ forces $p = 200$.

algorithm. We will see that one of these DFT operations is an identity, and we present a modified Golub–Kahan algorithm that eliminates it.

There is a close relationship between the Golub–Kahan algorithm and the Lanczos method [25–27] for square Hermitian matrices. It is important to notice that the Golub–Kahan algorithm, like the Lanczos algorithm, typically encounters some difficulties due to round-off error [26, p. 293] which require some extra work to maintain orthogonality of the basis vectors. Also, using block vectors also introduces the possibility that $\mathcal{K}_i(G, X_0)$ will have dimension less than si, where X_0 has s columns. However, these two problems have been extensively studied [28–30,10,31] and satisfactory solutions exist to both problems.

An important feature of Krylov subspace approximations to the truncated SVD is that the normalized Frobenius norm approximation error $\epsilon_C^{(k)}$ is determined exactly by singular value approximation error. This is a consequence of the relationship in (8) and Theorem 4 in [15]. Therefore, only singular value approximation error is important; and we notice that asymptotic bounds for singular value approximation error may shrink faster than singular vector approximation error.

3.1. The block Golub–Kahan algorithm

The Golub–Kahan algorithm in block form [22], presented in Algorithm 1, begins with a start block vector X_0 with s orthonormal columns and the input matrix C and computes orthonormal bases for $\mathcal{K}_i(G, X_0)$ and $\mathcal{K}_i(G_T, C^*X_0)$, as well as the projection of C into the intersection of $\mathcal{K}_i(G, X_0)$ and $\mathcal{K}_i(G_T, C^*X_0)$. When X_0 has only 1 column, Algorithm 1 reduces to the classic Golub–Kahan algorithm [21].

After k iterations of Algorithm 1, we have $CQ^{(k)} = P^{(k+1)}B^{(k+1,k)}$ and

$$P^{(k+1)*}CQ^{(k)} = B^{(k+1,k)}.$$

Then we can approximate C with

$$C \approx \hat{C}^{(k)} = P^{(k+1)}B^{(k+1,k)}Q^{(k)*}. \tag{8}$$

Algorithm 1 Golub–Kahan Bidiagonalization

Require: input matrix $C \in \mathbb{C}^{n\times m}$, start block vector $X_0 \in \mathbb{C}^{m\times s}$, number of iterations k.

1: $Q_1 \leftarrow X_0$
2: $B_0 \leftarrow 0$
3: **for** $j = 1, 2, \ldots, k$ **do**
4: $\quad R \leftarrow CQ_j - P_{j-1}B_{j-1}$
5: $\quad$ Let $P_jA_j = R$ be the QR factorization of R
6: $\quad S \leftarrow C^*P_j - Q_jA_j$
7: $\quad$ Let $Q_{j+1}B_j = S$ be the QR factorization of S
8: **end for**
9: **return** $Q^{(k)} \leftarrow [Q_1\ Q_2\ \ldots\ Q_k]$,
$P^{(k)} \leftarrow [P_1\ P_2\ \ldots\ P_k]$,

$$B^{(k,k)} \leftarrow \begin{bmatrix} A_1 & B_1 & & & \\ & A_2 & B_2 & & \\ & & \ddots & \ddots & \\ & & & A_{k-1} & B_{k-1} \\ & & & & A_k \end{bmatrix}.$$

Note that the only interactions with C are the matrix–vector products on lines 4 and 6 of Algorithm 1. Thus, the Golub–Kahan method may be used even when C is available only in factored form as ΦA, provided one can compute the products CQ_j and C^*P_j.

3.1.1. Matrix–vector products of the factored C in the Golub–Kahan algorithm

We are interested in how many DFT operations are needed inside the Golub–Kahan algorithm when Φ is evaluated matrix-free and C is never materialized explicitly. We have only introduced Φ for the sake of theoretical analysis, and it is almost always formed matrix-free with DFT operations.

To see how the products CQ_j and C^*P_j may be computed in the Golub–Kahan algorithm with DFT operations, we first recall the definition of C in terms of convolutions of a filter signal f with columns of A: $C := [(f * a_1)\ (f * a_2)\ \ldots\ (f * a_m)]$. We also notice that convolution in the Fourier domain is simply multiplication by a diagonal operator [18], $\Phi = \Psi \mathrm{diag}(\tilde{f})\Psi^*$, where Ψ is the DFT matrix [17], and $\tilde{f} = \Psi f$ is the Fourier transform of the filter signal. Then one can see that

$$C = \Psi^* \mathrm{diag}(\tilde{f})\Psi A.$$

It follows that

$$CQ_j = \Psi^* \mathrm{diag}(\tilde{f})\Psi (AQ_j)$$

and one requires $2s$ DFT operations to compute CQ_j, where Q_j has s columns. The case for C^*P_j is similar, except that

$$C^* = A^*\Psi^* \mathrm{diag}(\tilde{f}^*)\Psi$$

and then

$$C^*P_j = A^*(\Psi^* \mathrm{diag}(\tilde{f}^*)\Psi P_j).$$

This matrix–vector product also requires $2s$ DFT operations, and one iteration of Algorithm 1 will require $4s$ DFT operations.

3.1.2. Avoiding redundant DFT operations

Straightforward application of the previous section's observations can allow Algorithm 1 to form C using multiplications with A and 2 DFTs to form Φ. However, one of these DFT operations is an identity. This may be seen by noting the formulation of R on line 4 of Algorithm 1. If $C = \Phi A = \Psi^* \mathrm{diag}(\tilde{f})\Psi A$, then R becomes

$$R \leftarrow \Psi^* \mathrm{diag}(\tilde{f})\Psi AQ_j - P_{j-1}B_{j-1}.$$

Likewise $P_jA_i = R$ is a QR factorization of R, so

$$P_j \leftarrow (\Psi^* \mathrm{diag}(\tilde{f})\Psi AQ_j - P_{j-1}B_{j-1})A_j^{-1}.$$

Substituting this into line 6 of Algorithm 1 gives

$$S \leftarrow A^*\Psi^* \mathrm{diag}(\tilde{f})^*(\Psi\Psi^*)\mathrm{diag}(\tilde{f})\Psi AQ_jA_j^{-1} - P_{j-1}B_{j-1}A_j^{-1} - Q_jA_j \tag{9}$$

and the DFT to form R — indicated by parentheses in (9) — is an identity as Ψ is a unitary matrix. Therefore, we can eliminate one DFT per column by storing an intermediate value

$$\tilde{P}_j \leftarrow (\mathrm{diag}(\tilde{f})\Psi A Q_j - P_{j-1}B_{j-1})A_j^{-1}.$$

Thus, each Golub–Kahan iteration only requires $3s$ DFT operations. The modified algorithm is presented in Algorithm 2. Notice that Algorithm 2 never explicitly materializes either Φ or C; therefore, it is a matrix-free algorithm in some sense.

Algorithm 2 Block Golub–Kahan Bidiagonalization with Matrix-Free approximation of $C = \Phi A$

Require: input matrix $A \in \mathbb{C}^{n\times m}$, signal f with $\Phi = \Psi^* \mathrm{diag}(\tilde{f})\Psi$, start block vector $X_0 \in \mathbb{C}^{m\times s}$, number of iterations k.

1: $Q_1 \leftarrow X_0$
2: $B_0 \leftarrow 0$
3: **for** $j = 1, 2, \ldots$ **do**
4: $\quad \tilde{R} \leftarrow \mathrm{diag}(\tilde{f})\Psi(A^* Q_j) - \tilde{P}_{j-1}B_{j-1}$
5: $\quad R \leftarrow \Psi^*\{\tilde{P}_j\}$
6: $\quad$ Let $P_j A_j = S$ be the QR factorization of R
7: $\quad \tilde{P}_j \leftarrow \tilde{R}A_j^{-1}$
8: $\quad S \leftarrow \Psi^*(\mathrm{diag}(\tilde{f})^* \tilde{P}_j) - Q_j A_j$
9: $\quad$ Let QB_j be the QR factorization of S
10: **end for**
11: **return** $Q^{(k)} \leftarrow [Q_1\ Q_2\ \ldots\ Q_k]$,
$P^{(k)} \leftarrow [P_1\ P_2\ \ldots\ P_k]$,

$$B^{(k,k)} \leftarrow \begin{bmatrix} A_1 & B_1 & & & \\ & A_2 & B_2 & & \\ & & \ddots & \ddots & \\ & & & A_{k-1} & B_{k-1} \\ & & & & A_k \end{bmatrix}.$$

We remark that the addition of the matrix $\tilde{P}_j$ allows for computation of the DFT ΨA of A if $\Phi = I$ is the identity matrix. Then $\tilde{P}^{(k)} B^{(k,k)} Q^{(k)*} \approx A$, and can be used to approximate the truncated SVD of ΨA.

The asymptotic complexity for k iterations of Algorithm 2 using a block size of s for a $n \times m$ A, and a square Φ is $O(kmn + kn\log n + kns^2)$ when A is dense; the dense matrix–vector product requires $O(mn)$ FLOPS, the DFT operations require $O(n\log n)$, and the block vector operations require $O(ns^2)$ FLOPS. When A is sparse and has n_{nnz} nonzero elements, the complexity is $O(kn_{\mathrm{nnz}} + kn\log n + kns^2)$; the DFT operations are still $O(n\log n)$ and the block vector operations are still $O(ns^2)$, but the sparse matrix–vector products are $O(n_{\mathrm{nnz}})$. This may be compared against simply forming C explicitly, which would require $O(mn\log n)$ operations.

3.1.3. Similarities between the Golub–Kahan and Lanczos algorithms

The Golub–Kahan algorithm is closely related to the Lanczos algorithm [25], which is also available in block form [32,27]. The block Lanczos algorithm accepts a square matrix G and block vector X_0, and generates an orthonormal basis $Q^{(k)}$ for $\mathcal{K}_k(G, X_0)$ and a block-tridiagonal matrix $T^{(k,k)} = Q^{(k)*} G Q^{(k)}$. Closer inspection of the algorithm reveals that when $G := CC^*$, then the $Q^{(k)}$ produced by the block Golub–Kahan algorithm and those produced by the block Lanczos algorithm are the same in infinitely-precise arithmetic. Also, $B^{(k,k)} B^{(k,k)*} = T^{(k,k)}$ [10, p. 144].

When $n \gg m$, the block Lanczos method run on G_T may be preferred due to cost concerns. The cost of the QR factorizations and recovering from loss of orthogonality is reduced in the block Lanczos algorithm, and the basis $Q^{(k)}$ may simply be discarded if it is not needed. However, we notice that in the block Golub–Kahan algorithm, one may simply not store the previous Q_j, not return $Q^{(k)}$, and enforce orthogonality only on the P_j iterates. The coefficient B_j may be computed with a Cholesky factorization of $P_j^* CC^* P_j$ rather than the operation on line 9 of Algorithm 2. Then the block Golub–Kahan iteration is essentially the same as block Lanczos, in infinitely-precise arithmetic.

We remark that block Lanczos algorithm will be more prone to round-off error due to the effect of squaring singular values: the coefficient matrix $A_j^{(\mathrm{Lanczos})}$ used in the block Lanczos recurrence (written as A_j in [32, Eq. 4.1]) has $A_j^{(\mathrm{Lanczos})} = B_j B_j^*$ with B_j used in Algorithm 1. The condition number of $A_j^{(\mathrm{Lanczos})}$ is then the square of the condition number of B_j, and the problem is more badly conditioned.

3.2. Singular value error and normalized Frobenius norm error

There is a significant relationship between singular value approximation error and normalized Frobenius norm error $\epsilon_C^{(k)}$ when $\hat{C}^{(k)}$ is defined as in (8): $\epsilon_C^{(k)}$ may be defined exactly by approximation error of the squares of singular values

$\sigma_j^2(C) - \sigma_j^2(\hat{C}^{(k)})$ for $j = 1, 2, \ldots, m$. To see this, we note that $B^{(k+1,k)} = P^{(k+1)*} C Q^{(k+1)}$. Theorem 4 in [15] generalizes to complex matrices, and gives

$$\begin{aligned}\|C - \hat{C}^{(k)}\|_F &= \sqrt{\operatorname{tr}(CC^*) - \operatorname{tr}(B^{(k+1,k)} B^{(k+1,k)*})} \\ &= \sqrt{\sum_{i=1}^{sk} \sigma_i(C)^2 - \sigma_i(B^{(k+1,k)})^2 + \sum_{i=1}^{sk+1} \sigma_i(C)^2}\end{aligned}$$

as $B^{(k+1,k)}$ has at most sk nonzero singular values, and therefore

$$\epsilon_C^{(k)} = \frac{\sqrt{\sum_{i=1}^{sk} \sigma_i(C)^2 - \sigma_i(B^{(k+1,k)})^2 + \sum_{i=1}^{sk+1} \sigma_i(C)^2}}{\|C\|_F}.$$

Note that the asymptotic shrinkage of the error $\sigma_i(C)^2 - \sigma_i(B^{(k+1,k)})^2$ is characterized in Theorem 3.2 in [22]. Noting the similarities between the Golub–Kahan and Lanczos algorithms allows us to apply theorems in [9] to $\sigma(CC^*) = [\sigma_1(C)^2\, \sigma_2(C)^2 \ldots]$. An important implication of this observation is that only singular value approximation error is important for Krylov subspace approximations. Comparison of Theorems 5 and 6 in [9] shows that singular value approximation error may shrink faster than singular vector approximation error when the $\hat{\gamma}_i$ are large because of the squaring of the Chebyshev polynomial in the denominator of Equation 3.10.

3.3. Practical considerations

Though the Golub–Kahan algorithm produces orthonormal bases $Q^{(k)}$ and $P^{(k)}$ when one's arithmetic is infinitely-precise, practical implementations will encounter a loss of orthogonality between the columns of $Q^{(k)}$ and $P^{(k)}$. Loss of linear independence of the columns of Q_j or P_j is also a potential problem. However, both of these issues are well-known and likewise well-studied. Due to the similarity between the Golub–Kahan algorithm and the Lanczos algorithm, the solutions for the Lanczos algorithm may simply be applied to the Golub–Kahan algorithm.

Loss of orthogonality in the Golub–Kahan basis vectors is caused by the convergence of an approximate singular vector of A to within machine tolerance of a true singular vector [33]. This can be remedied by adding a full Gram–Schmidt orthonormalization step to explicitly set $Q_i \perp Q_j$ and $P_i \perp P_j$ for $j < i$; the result is that the complexity of the Golub–Kahan algorithm becomes $O(k^2mn + k^2n \log n)$ when A is dense and $O(k^2 n_{\text{nnz}} + k^2 n \log n)$ when A is sparse. When k is small, this is acceptable, and is the approach used in [13,12]. Partial reorthogonalization [29] estimates loss of orthogonality as it iterates, and only orthogonalizes when it is necessary. This strategy is used in [11], which also contains other results regarding the effects of loss of orthogonality on low-rank approximation problems.

Loss of linear independence in the block vectors Q_j and P_j is also well-studied, for example, see the discussion in [10] for the closely-related band Lanczos algorithm. Loss of linear independence is not problematic, since one may either deflate [10] or inflate [31] the Krylov subspace. Additionally, loss of linear independence is linked to the existence of a vector $b \in \operatorname{colspan}\{Q_j\}$ such that the tangent of the angle between b and $\mathcal{K}_{i-1}(G, X_0)$ is tiny. An analogous case holds for P_j and $\mathcal{K}_{i-1}(G_T, C^*X_0)$. Note that this is equivalent to having a tiny residual when solving $G^{-1}x - b$ using the block Conjugate Gradient method [34] with a start guess of Q_j for some $b \in \operatorname{colspan}\{Q_j\}$, as $G^{-1}\mathcal{K}_{i-1}(G^{-1}, GQ_j) = \mathcal{K}_{i-1}(G, X_0)$. When i is small, it is unlikely that this residual is small enough to require deflation, and we did not observe any case that required deflation in our experiments.

4. Numerical examples

We recall the example matrix problem $C \coloneqq [(f * a_1)\ (f * a_2)\ \ldots\ (f * a_m)]$. Direct formation of C may be accomplished by convolving each column of A with f, requiring m DFTs. We seek to reduce the number of DFTs necessary by approximating C. The products Cx and C^*y can be formed with DFTs, and each dimension of the Krylov subspace therefore requires only 3 DFTs. When A is a good candidate for low-rank approximation, computational savings may be obtained, as a good approximation will be available with only a small number of Golub–Kahan iterations.

To show the performance of approximation of C with the Golub–Kahan algorithm, we present two examples from signal processing: frequency-domain integration of a series of 1-dimensional signals, and simulation of a series of cameras with a fixed correction filter on a target image. The integration example is only intended to show the convergence of C in Krylov subspaces. The camera simulation example also shows this behavior. In addition, the camera simulation example shows the scalability of block Golub–Kahan iteration. We compare that scalability to simply parallelizing all m convolutions to directly form C. Additionally, the camera example shows how increasing m leads to a sublinear increase in the number of dimensions k needed to obtain a fixed approximation error. Therefore, obtaining an approximation of C with a fixed relative error is asymptotically less expensive than directly forming C when the process that forms A produces vectors with a large degree of linear dependence.

For both examples, we compare our block Golub–Kahan approximation scheme against a baseline method, presented in Section 4.1. This baseline method is also designed to approximate C with a low-rank matrix while performing as few DFT operations as possible, but uses the SVD of A rather than approximating the SVD of C.

4.1. The baseline approximation method

We first present a baseline method against which to compare our block Golub–Kahan method. Though this method is not necessarily Frobenius-norm optimal, it does have the virtue of requiring only k DFT operations to product a rank-k approximation, compared to $3k$ DFTs for the block Golub–Kahan method.

When a matrix to be approximated is presented in factored form, as is the case with $C = \Phi A$, then singular vector projections based on the SVD of A may still be used, but will no longer be optimal with respect to the Frobenius norm. When the matrix A with SVD $A = U\Sigma V^*$ has rank r and therefore has only r nonzero singular values, then C may be recovered exactly, with $C = \Phi U^{(r)}\Sigma^{(r,r)}V^{(r)*}$. For $k \leq r$ we may still approximate C as

$$\hat{C}^{(k)} := \hat{U}^{(k)}\Sigma^{(k,k)}V^{(k)*} \tag{10}$$

with $\hat{U} := [(f * u_1)\ (f * u_2)\ \ldots\ (f * u_k)] = \Phi U^{(k)}$; bounds on the error are developed in [15]. This approach only requires k convolutions to produce a rank-k approximation.

When $k < r$, the Frobenius norm optimality of this approach is no longer guaranteed. Error depends on the intersection between the right and left singular vector spaces of Φ and A. That is, we note that $AV^{(k)}V^{(k)*} = U^{(k)}\Sigma^{(k,k)}V^{(k)*}$ and substituting into (10) gives

$$\hat{C}^{(k)} = \Phi U^{(k)}\Sigma^{(k,k)}V^{(k)*} = \Phi AV^{(k)}V^{(k)*}.$$

Let $QR = \Phi U$ be a QR factorization of ΦU and $R\Sigma = X\Upsilon Z^*$ be the SVD of $R\Sigma$. The structure of Z^* induces rotations that transform the singular vectors of A into singular vectors of ΦA. When the upper-left corner Z^* is nearly block-diagonal with small off-block diagonal entries, then there will exist a small k such that $\|Z^*V^*V^{(k)}\|_F^2$ is nearly k and the baseline method will be near-optimal.

4.2. Signal integration example

In this problem, a $40{,}000 \times 487$ matrix A is composed of 1-dimensional acceleration time histories that must be integrated. This may be performed by simply convolving each column of A with the Heaviside step function defined as

$$f[t] = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Both the matrix A and the diagonal matrix $\Phi = \Psi^*\mathrm{diag}(\tilde{f})\Psi$ have rapidly-decaying singular values, where Ψ is the DFT operator. The singular values of A, Φ and C are shown in Fig. 3. These plots indicate that the matrix C is a good candidate for low-rank approximation due to its rapidly-decaying singular values. In fact, C has more rapidly-decaying singular values than A.

The matrix A has gap-rank$\{A, 10^{-15}\} = 201$. Using Remark 1, we look for the smallest p such that $\sigma_1(\Phi A) \geq \sigma_1(A)\sigma_p(\Phi)$, which is $p = 12$. Then any τ which gives gap-rank$\{\Phi, \tau\} \geq 23$ will also give gap-rank$\{\Phi, \tau\} \geq$ gap-rank$\{\Phi A, \tau\}$. For example, while Φ has gap-rank$\{\Phi, 10^{-2}\} = 66$, ΦA has gap-rank$\{\Phi A, 10^{-2}\} = 13$.

One may notice that rank$\{C\} = 200$ because rank$\{A\} = 200$. This implies that at most 200 Golub–Kahan iterations are necessary, or even possible. The Krylov subspace $\mathcal{K}_k(G, X_0)$ will be exhausted when its dimension is greater than 200 provided the start block vector X_0 is chosen so that it is not orthogonal to any left singular vector of C.

4.2.1. Approximation errors

In this section, we consider Krylov subspaces generated with a block size $s = 1$, using a full Gram–Schmidt reorthogonalization as in [13].

Because the singular values of C decay rapidly, approximations of singular values from Krylov subspaces will have low normalized Frobenius norm error in only a few iterations [9,22]. Indeed, after only 5 iterations, the worst-case error for the largest singular value predicted by Theorem 6 in [9] applied to $G = CC^*$ is less than 1. Considering that this singular value is responsible for more than 93% of the overall Frobenius norm of C, its rapid convergence implies rapid reduction in error for low-rank approximations generated by Algorithm 1. Errors for approximation of C using both Algorithm 2 and the baseline method from Section 4.1 are shown in Fig. 4. These plots show error both in terms of subspace dimension k and in terms of the number of DFT operations required to generate the low-rank approximation. The Frobenius-norm optimal approximation error based on the truncated SVD of C is also shown.

Fig. 4 shows that the Golub–Kahan iteration produces smaller approximation error than the baseline approach both in terms of DFT operations and in terms of subspace dimension. In terms of subspace dimension, the Golub–Kahan method always produces smaller errors. In terms of DFT operations, the Golub–Kahan approach produces lower error

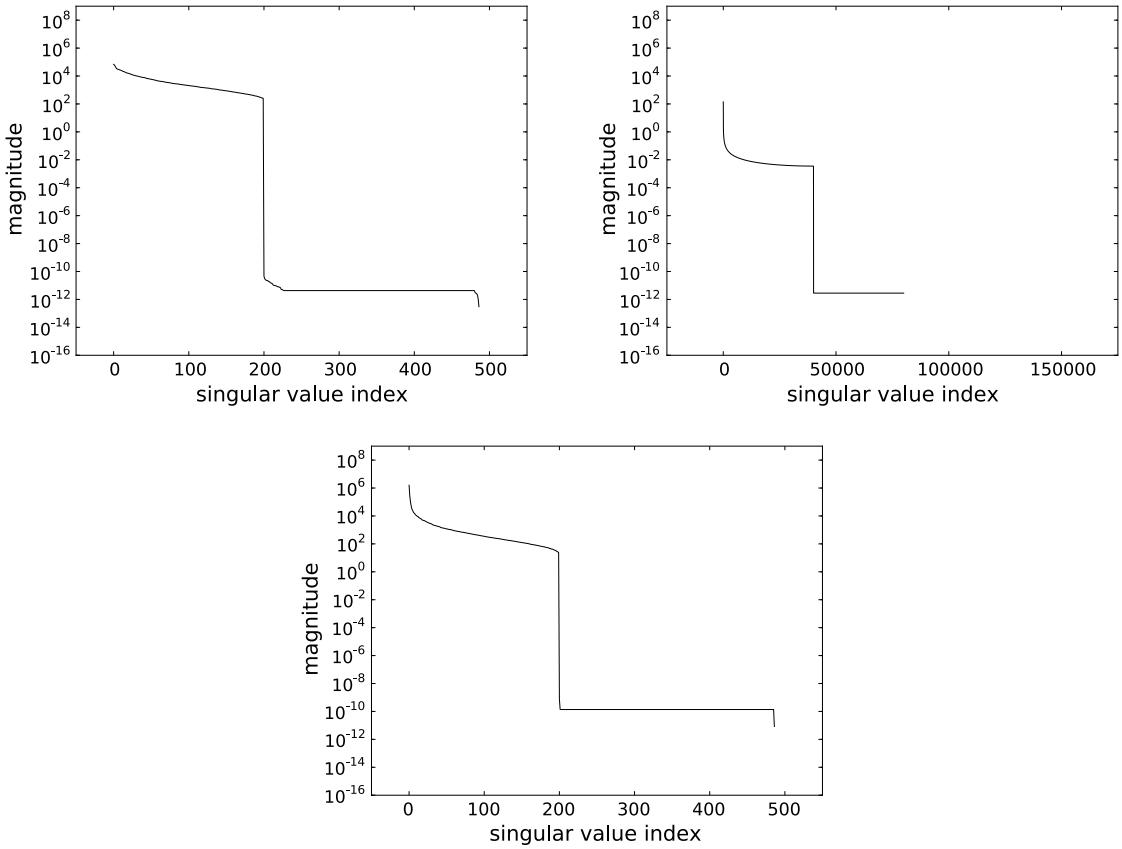

Fig. 3. Signal integration: singular values of A (top left), diagonal matrix for $\Phi = \Psi^* \mathrm{diag}\{\tilde{f}\}\Psi$ (top right) and $C = \Phi A$ (bottom).

approximations up to 200 DFTs, at which point the baseline method is guaranteed to achieve zero approximation error. Despite requiring 3 times as many DFT operations per dimension, the Golub–Kahan approach produces approximation errors nearly an order of magnitude smaller than the baseline approach for small subspace dimensions, even when compared in terms of DFT operations rather than subspace dimension. Moreover, the Krylov subspace approximations also nearly equal to the error of the Frobenius norm-optimal, truncated SVD approximation; this was also witnessed in [11].

4.3. Camera simulation example

In this example, we consider the problem of simulating a series of camera point spread functions (PSFs) on a camera test target; PSFs are the impulse response functions of the camera system in the spatial domain. When an imaging system is shift invariant, only one PSF describes the whole imaging system. However, simulating an imaging system typically requires evaluation of many different PSFs because many imaging systems are not actually shift invariant; rather, they are treated as being locally shift invariant. Many different PSFs must be evaluated. The larger the number of PSFs that can be evaluated, the closer the simulated image is to the actual result of the shift variant imaging system.

In this case, Φ represents a target image, and A is composed of a series of PSFs. One particular advantage of PSFs is that they have compact spatial support. This implies that significant savings may be realized with a sparse representation, both in terms of storage in memory and compute costs for factored evaluation of $C = \Phi A$. Each matrix–vector product with A uses $O(A_{\mathrm{nnz}} + n)$ FLOPS, where A_{nnz} is the number of nonzero elements of A. The Golub–Kahan approach is asymptotically superior to the baseline approach in this case.

In this example, we consider the simulation of a set of camera systems coupled with a sharpening filter on a checkerboard test image. The test image, a sample PSF and the simulated image are shown in Fig. 5.

We approximated the matrix C with $k \leq 50$. In this example, we also show the effect of block size s in the Golub–Kahan algorithm on exposing parallelism through both linear algebra operations and the embarrassingly parallel DFT operations on the columns of P_j and Q_j.

Our implementation used Python, Numpy [35] and its FFTPACK [36] bindings. Although Python does have a global interpreter lock that inhibits parallel code, we note that all of the Python-level operations are simply acquisition of work buffers for FFTPACK, which would be synchronized in any implementation. The DFT operations are outside of the Python

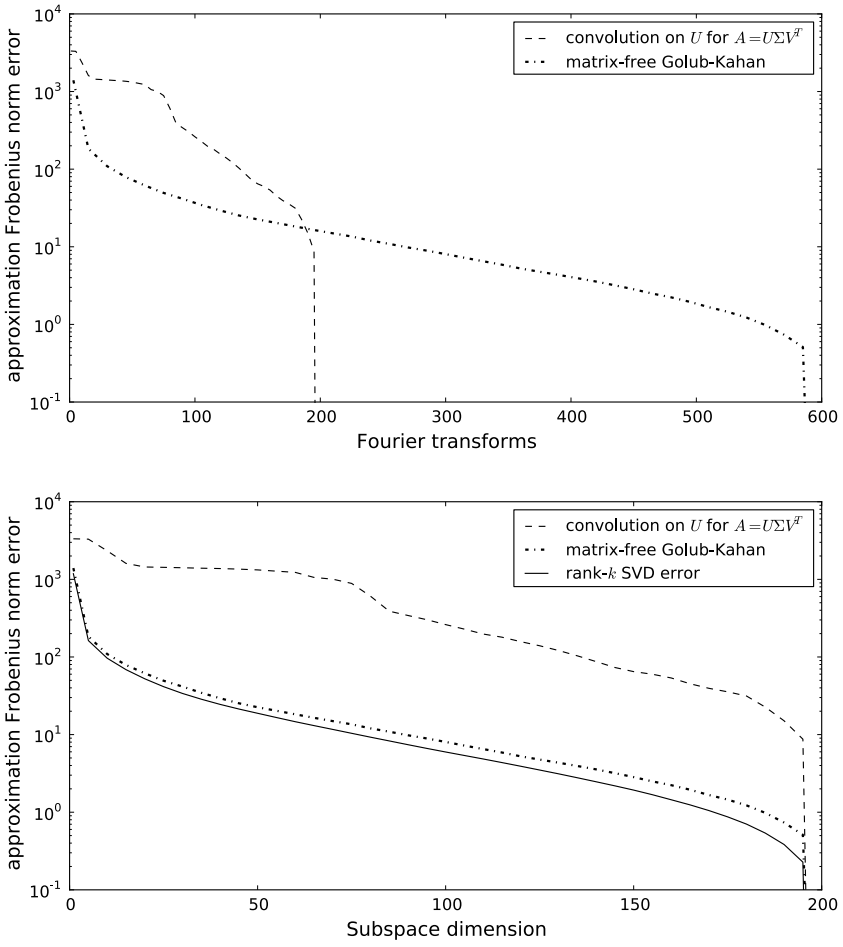

Fig. 4. Signal integration: approximation error of baseline and matrix-free Golub–Kahan approximations of C with $s = 1$; the X axis is Fourier transforms on top and subspace dimension on bottom. The approximation error from the truncated SVD of C — that is the Frobenius norm-optimal rank-k approximation — is shown in the bottom diagram.

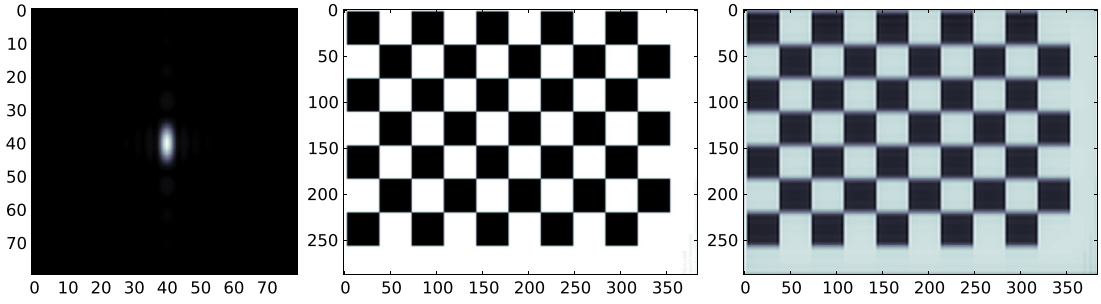

Fig. 5. Camera simulation: Sample PSF (left), test image (center), and test image convolved with the PSF (right).

global lock, and our results witness parallel speedup of DFT operations. Numpy was further linked against ATLAS [23] to allow for parallel numerical linear algebra operations.

All PSFs and the target image are 288 × 384 pixels. PSFs were derived from Fraunhofer diffraction on a rectangular-aperture [37]; aperture heights, widths, depths to focal plane, and light wavelengths were varied to generate the PSFs. The resulting matrix derived from zero-padding and flattening the PSFs is 442,368 × 625. The matrix Φ is simply $\Psi^* F \Psi$, where Ψ is the 2-dimensional DFT matrix, and F is a diagonal matrix assembled from the DFT of the test image. The spectra of A, Φ and C are shown in Fig. 6.

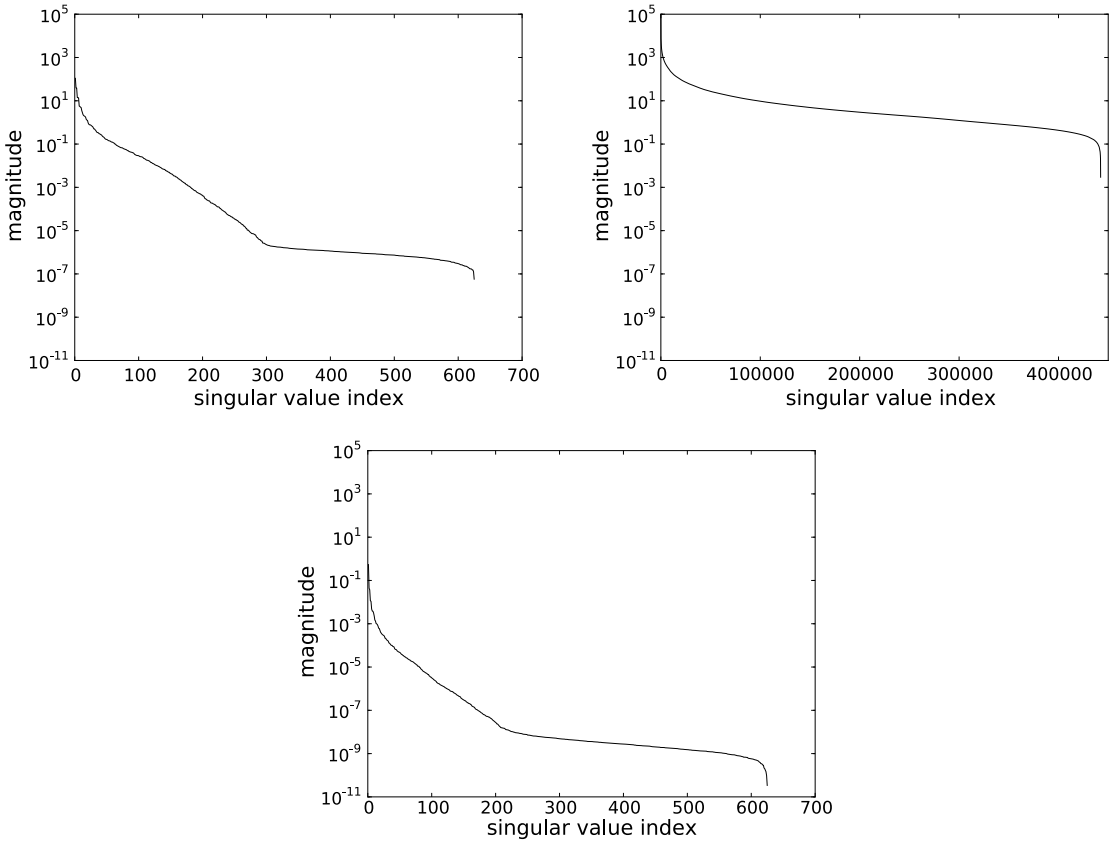

Fig. 6. Camera simulation: spectra of A (top left), Φ (top right) and C (bottom).

4.3.1. Approximation errors

In this section, we only consider bounds and errors for a block size $s = 1$; we also use full reorthogonalization.

The matrix A has gap-rank$\{A, 10^{-3}\} = 62$. The gap-rank bounds from Theorem 1 are pessimistic, as the smallest p that satisfies $\sigma_1(\Phi A) \geq \sigma_1(\Phi)\sigma_p(A)$ is $p = 286$, but the calculated gap-rank of $C = \Phi A$is gap-rank$\{C, 10^{-3}\} = 19$.

The leading singular values of C are again well-separated, so we may expect rapid convergence of singular value approximations in the Krylov subspaces $\mathcal{K}_k(G, x_0)$. In this example, the baseline method exhibits poor approximation of C, as high-frequency, local variations in the PSFs are amplified by the sharpening filter. As predicted by the theoretical bounds, the matrix-free approach always produces approximations with smaller error than the baseline method for $k \leq 50$.

At 30 dimensions, the actual error satisfied $\epsilon_C^{(30)} < 1.2 \times 10^{-3}$. The normalized errors for approximations of dimension k with $1 \leq k \leq 50$ are shown in Fig. 7.

4.3.2. Asymptotic complexity

An advantage of approximating C is noted in this example since A is sparse. When A is sparse, the complexity of the matrix–vector products in Algorithm 2 only depends on the number of nonzero entries, which is typically $O(n)$. As we have mentioned, the complexity of finding a rank-k approximation of C is only $O(kn \log n)$, compared to $O(mn \log n)$ for directly computing C, assuming that no explicit construction of C is required. When $k \in o(m)$; that is, when k is asymptotically dominated by m, then approximating C will be asymptotically better than directly forming C. Then, given a fixed degree of parallelism $p_\parallel$, there will always be a problem size for which approximating C is less computationally expensive than directly forming C, even with $p_\parallel$ workers. Theorems 1 and 2 may be used to place an upper bound on k.

These PSFs exhibit a large degree of linear dependence; i.e. small variations in aperture size, depth to focal plane or light wavelength result in only small variations in the resulting PSF. The gap-rank of the matrix A is asymptotically dominated by the number of PSFs, and Theorem 1 shows that the gap-rank of C has a linear relationship to the ranks of A. Then the gap-rank grows sublinearly compared to the number of columns of A, and we expect an asymptotic advantage in approximating C rather than generating it directly when the number of columns are varied.

To show this asymptotic difference, we conducted an experiment in which we varied the number of PSFs. In the preceding experiments, the number of PSFs was fixed at 625. Here, we vary the number of PSFs from 81 to 6561. We approximated C using only as many dimensions k as were required to produce an approximation with normalized Frobenius norm error

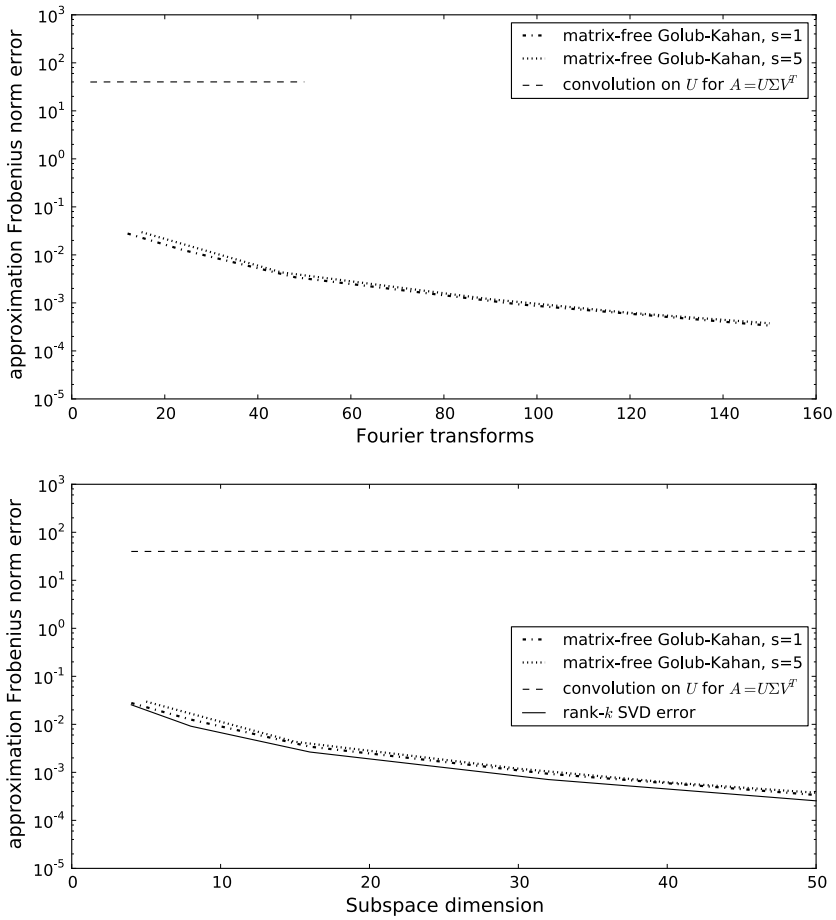

Fig. 7. Camera simulation: Approximation error of baseline and matrix-free Golub–Kahan approximations of C with $s = 1$; the X axis is Fourier transforms on top and subspace dimension on bottom. Again, approximation error from the truncated SVD of C is shown for comparison in the bottom diagram.

of less than 0.001. We computed p from Theorems 1 and 2; these are shown in Fig. 8. For all matrices, the quantity $\hat{\tau} := \sigma_1(\Phi)\sigma_p(A)/\sigma_p(\Phi)\sigma_1(A) < 10^{-5}$. Thus, we can expect $k \leq p$ for all matrices. The number of dimensions needed to approximate $C := \Phi A$ to less than 10^{-3} normalized Frobenius norm error is shown in Fig. 8.

We used a block size $s = 1$ in the Golub–Kahan algorithm. We used explicit reorthogonalization for the basis vectors. The compute times to approximate C and to directly form it with no parallelism are also shown in Fig. 8. Note that the compute times to form C directly grow faster than the compute times to approximate C to less than 0.001 normalized Frobenius norm error. This is because the gap-rank$\{C, \tau\} \in o(m)$ for a fixed τ when the number of columns m of C is varied.

4.3.3. Compute costs and parallel scalability

Approximating C is intended to reduce computational costs compared to simply computing the convolution of all columns of A with m DFTs. Coupling the convolution with approximation in a Krylov subspace produces significant savings; the block Golub–Kahan algorithm also allows one to leverage parallel computational hardware.

We measured wall clock time for convolution of all m PSFs with the test image and sharpening filter, and the wall clock time for computation of a low-rank approximation of C for various k for the matrix-free approach. We parallelized convolution of all m PSFs by simply allowing up to $p_\parallel$ convolutions to operate independently, where $p_\parallel$ is the parallelism factor. We also allowed up to $\min\{p_\parallel, s\}$ DFTs to operate independently for the block Golub–Kahan algorithm.

We present two sets of timings for scaling: one for calculating the $Q^{(k)}$, $B^{(k,k)}$ and $P^{(k)}$ factors of the Golub–Kahan bidiagonalization $C \approx P^{(k)}B^{(k,k)}Q^{(k)*}$, and one which also explicitly constructs $\hat{C} \leftarrow P^{(k)}B^{(k,k)}Q^{(k)*}$. An explicit construction of $\hat{C}$ is not always necessary. Some operations were performed in the Krylov subspace; for example, one can compute Euclidean distance between data in the Krylov subspace. For this example, one could find the convolved image that was closest to its unconvolved self in the 2-norm. Other tasks, such as 1-norm calculation, require explicit construction of the approximation $\hat{C}$.

The compute times to find an approximation in a 30-dimensional Krylov subspace are shown in Fig. 9. These compute times are the averages of 10 runs, with the error bars showing 3 standard deviations from the means. We note that

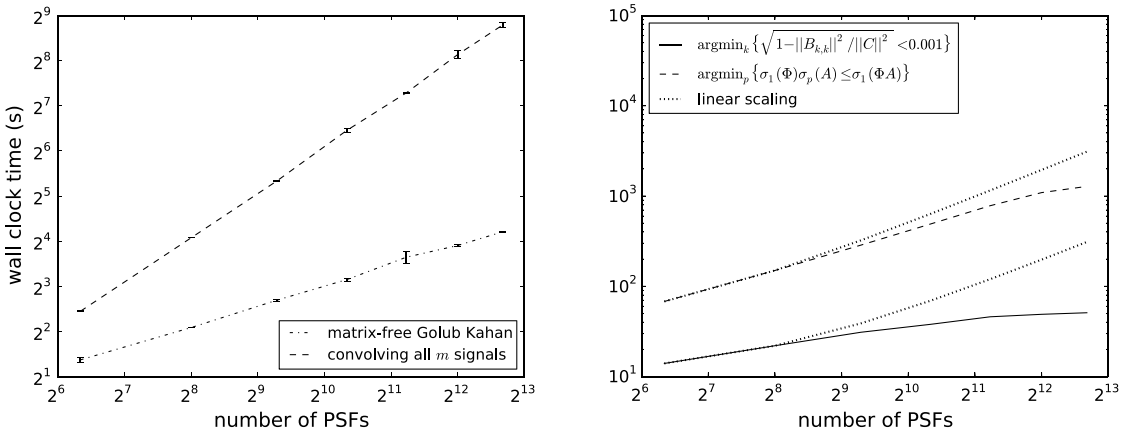

Fig. 8. Camera simulation: wall clock times to compute a low-rank approximation to C as a function of m (the number of PSFs in A). The compute time to convolve all PSFs with the image and filter are shown on the left. Note there is no parallelization used in this experiment. For each different number of PSFs, we computed C to have normalized error $\|C - \hat{C}\|/\|C\| \leq 10^{-3}$; the number of dimensions k needed to achieve that error is shown on the right; these k may be compared with the p from Theorem 2. The value of $\hat{\tau}$ from Theorem 2 is less than 10^{-5} for all matrices. Note that both p and k grow sublinearly as a function of the number of PSFs.

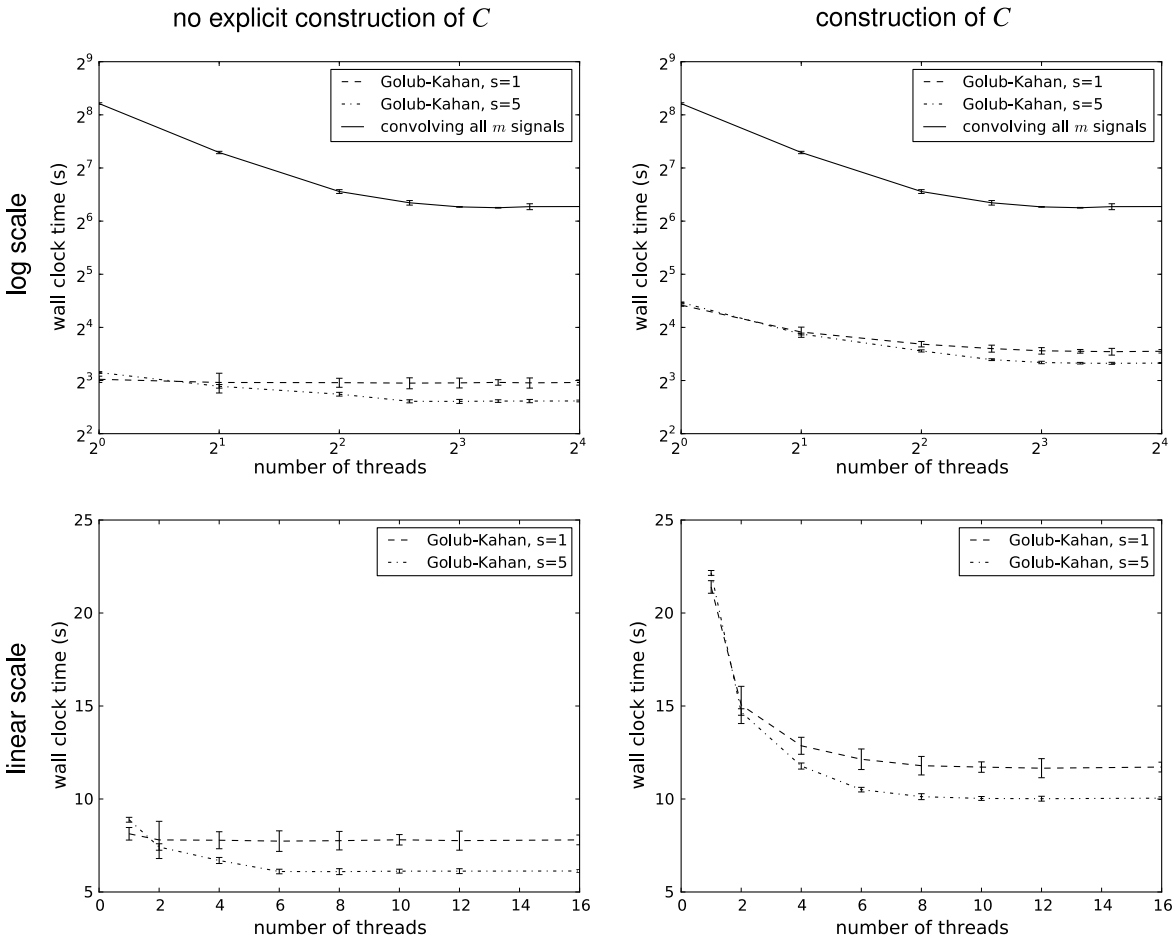

Fig. 9. Camera simulation: wall clock times versus parallelization. Top plots: Comparison of wall clock time to directly form C to wall clock time to approximation of C using matrix free Golub–Kahan iteration, without (left plot) and with (right plot) explicitly constructing $\hat{C}$ (logarithmic scale). Bottom plots: Comparison of wall clock times for block size $s = 1$ and $s = 5$ without (left plot) and with (right plot) explicitly constructing $\hat{C}$ (linear scale). The times are averaged over 10 runs; error bars are 3 standard deviations.

parallelism was exposed both in the linear algebra operations of the Golub–Kahan algorithm, and the matrix-free DFTs. Also, as the left plot in Fig. 9 shows, some parallelism is exposed in the explicit construction of $\hat{C}$.

5. Conclusion

We have presented a matrix-free approach to exploit implicit application of an LSI system to a set of m data points $a_i[t]$ with a significant degree of linear dependence. When the $a_i[t]$ have such linear dependence, then the LSI-transformed $c_i[t] = (a_i * f)[t]$ may maintain enough linear dependence that one may approximate all m $c_i[t]$ without performing m DFTs. We present a theorem that provides upper bounds on singular value decay for a factored matrix $C = \Phi A$; this theorem shows that singular values of $C = \Phi A$ decay at least as rapidly as those of A, subject to the smallest p that satisfies $\sigma_1(\Phi A) \geq \sigma_1(A)\sigma_p(\Phi)$ (cf. Theorem 1). When A and Φ have rapidly-decaying singular values and p is small, then singular values of $C = \Phi A$ will decay at least as fast as those of A. When the singular values of $C = \Phi A$ decay rapidly, a low-rank approximation to $C = \Phi A$ with negligible normalized Frobenius norm error may be produced with only a few DFTs.

We have shown how the truncated SVD of $C = \Phi A$ can be used to approximate $C = \Phi A$ without performing m DFTs. We have also shown how a Krylov subspace projection can be used to approximate the truncated SVD. The block Golub–Kahan algorithm can generate the necessary Krylov projections, and $C = \Phi A$ need not be formed explicitly. We have proposed a variant of the block Golub–Kahan algorithm to form C in terms of its factors and with Φ matrix-free that reduces the number of DFT operations per iteration from 4 s to 3 s.

Our approach can reduce the number of DFT operations necessary if some approximation error can be tolerated. The block Golub–Kahan algorithm also exposes parallelism in its linear algebra operations and in the matrix-free formation of the products CQ_j and C^*P_j. Often, approximations with low error may be realized with only a few Golub–Kahan iterations and DFT operations. We showed examples performing time–history integration and camera simulation. We note that Krylov subspace methods that exploit the 2-dimensional structure of image or multi-dimensional array data [38] can also be modified in an identical way to our modified block Golub–Kahan algorithm to produce approximations to array data without any flattening.

Acknowledgment

The authors would like to acknowledge Martina Barnas for her helpful suggestions in the preparation of this manuscript.

References

[1] A.-J. Van Der Veen, E.F. Deprettere, A.L. Swindlehurst, Subspace-based signal analysis using singular value decomposition, Proc. IEEE 81 (9) (1993) 1277–1308.
[2] F. Castells, P. Laguna, L. Sörnmo, A. Bollmann, J.M. Roig, Principal component analysis in ECG signal processing, EURASIP J. Appl. Signal Process. 2007 (1) (2007) 98–98.
[3] Y. Liang, H. Lee, S. Lim, W. Lin, K. Lee, C. Wu, Proper orthogonal decomposition and its applications—part I: Theory, J. Sound Vib. 252 (3) (2002) 527–544.
[4] J. Mendel, Lessons in Estimation Theory for Signal Processing, Communications, and Control, Pearson Education, 1995, URL: https://books.google.com/books?id=8Ks0-nLhi84C.
[5] I. Jolliffe, Principal Component Analysis, in: Springer Series in Statistics, Springer, 2002, URL: https://books.google.com/books?id=_olByCrhjwIC.
[6] D. Skillicorn, Understanding Complex Datasets: Data Mining with Matrix Decompositions, in: Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, CRC Press, 2007, URL: https://books.google.com/books?id=9SzLDi8jUnAC.
[7] M. Turk, A. Pentland, Eigenfaces for recognition, J. Cogn. Neurosci. 3 (1) (1991) 71–86.
[8] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, R. Harshman, Indexing by latent semantic analysis, J. Am. Soc. Inf. Sci. 41 (6) (1990) 391–407.
[9] Y. Saad, On the rates of convergence of the Lanczos and the block-Lanczos methods, SIAM J. Numer. Anal. (1980) 687–706.
[10] J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.
[11] H. Simon, H. Zha, Low-rank matrix approximation using the Lanczos bidiagonalization process with applications, SIAM J. Sci. Comput. 21 (6) (2000) 2257–2274.
[12] K. Blom, A. Ruhe, A Krylov subspace method for information retrieval, SIAM J. Matrix Anal. Appl. 26 (2005) 566–582. http://dx.doi.org/10.1137/S0895479803392261.
[13] J. Chen, Y. Saad, Lanczos vectors versus singular vectors for effective dimension reduction, IEEE Trans. Knowl. Data Eng. (2009) 1091–1103.
[14] J. Chen, H. Fang, Y. Saad, Fast approximate k NN graph construction for high dimensional data via recursive lanczos bisection, J. Mach. Learn. Res. 10 (2009) 1989–2012.
[15] A. Breuer, Minimal Krylov subspaces for dimension reduction, Ph.D. thesis, Bloomington, IN USA, ARL-RP-0480; Reprint of ProQuest AAI7525622, 2013. URL: http://www.arl.army.mil/arlreports/2014/ARL-RP-0480.pdf.
[16] V. Madisetti, The Digital Signal Processing Handbook, in: Electrical Engineering Handbook, Taylor & Francis, 1997, URL: https://books.google.com/books?id=Zhc36gyobk0C.
[17] J. Demmel, Applied Numerical Linear Algebra, Society for Industrial and Applied Mathematics, 1997.
[18] R. Bracewell, The Fourier Transform and Its Applications, in: McGraw-Hill international editions, McGraw-Hill Education, 2000, URL: http://books.google.com/books?id=ecH2KgAACAAJ.
[19] F. Zhang, Matrix Theory: Basic Results and Techniques, Springer, 2011.
[20] P. Zhu, A. Knyazev, Angles between subspaces and their tangents, J. Numer. Math. 21 (4) (2013) 325–340. http://dx.doi.org/10.1515/jnum-2013-0013.
[21] G. Golub, W. Kahan, Calculating the singular values and pseudo-inverse of a matrix, J. Soc. Ind. Appl. Math.: Ser. B, Numer. Anal. 2 (2) (1965) 205–224.
[22] G. Golub, F. Luk, M. Overton, A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix, ACM Trans. Math. Software (TOMS) 7 (2) (1981) 149–169.
[23] R. Whaley, A. Petitet, Minimizing development and maintenance costs in supporting persistently optimized BLAS, Softw. - Pract. Exp. 35 (2) (2005) 101–121. http://www.cs.utsa.edu/~whaley/papers/spercw04.ps.

[24] Q. Wang, X. Zhang, Y. Zhang, Q. Yi, Augem: Automatically generate high performance dense linear algebra kernels on x86 CPUs, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '13, ACM, New York, NY, USA, 2013, pp. 25:1–25:12. http://dx.doi.org/10.1145/2503210.2503219, URL: http://doi.acm.org/10.1145/2503210.2503219.
[25] C. Lanczos, Iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Natl. Bur. Stand. 45 (1950) 255–282.
[26] B. Parlett, The Symmetric Eigenvalue Problem, in: Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, 1998, URL: https://books.google.com/books?id=uaYXlkHVPpEC.
[27] R. Underwood, An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems., Ph.D. thesis, Stanford, CA, USA, AAI7525622 (1975).
[28] B. Parley, D. Scott, The Lanczos algorithm with selective orthogonalization, Math. Comp. 33 (145) (1979) 217–238.
[29] H. Simon, The Lanczos algorithm with partial reorthogonalization, Math. Comp. 42 (165) (1984) 115–142.
[30] J. Cullum, R. Willoughby, Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. 1, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.
[31] Q. Ye, An adaptive block Lanczos algorithm, Numer. Algorithms 12 (1) (1996) 97–110.
[32] J. Cullum, W. Donath, A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, real symmetric matrices, in: 1974 IEEE Conference on Decision and Control Including the 13th Symposium on Adaptive Processes, Vol. 13, IEEE, 1974, pp. 505–509.
[33] H. Simon, Analysis of the symmetric Lanczos algorithm with reorthogonalization methods, Linear Algebra Appl. 61 (1984) 101–131.
[34] D. O'Leary, The block conjugate gradient algorithm and related methods, Linear Algebra Appl. 29 (1980) 293–322. http://dx.doi.org/10.1016/0024-3795(80)90247-5, URL: http://www.sciencedirect.com/science/article/pii/0024379580902475.
[35] T. Oliphant, A Guide to NumPy, Vol. 1, Trelgol Publishing, USA, 2006.
[36] P. Swarztrauber, FFTPACK: A package of Fortran subprograms for the fast Fourier transform of periodic and other symmetric sequences, 1985. http://netlib.org/fftpack.
[37] R. Driggers, Encyclopedia of Optical Engineering, in: Dekker Encyclopedias Series, vol. 3, Marcel Dekker, 2003, no. URL: http://books.google.com/books?id=rcrGlrguj1YC.
[38] C. Ren, D. Dai, Bilinear Lanczos components for fast dimensionality reduction and feature extraction, Pattern Recognit. 43 (11) (2010) 3742–3752.