SIAM J. MATRIX ANAL. APPL.
Vol. 31, No. 5, pp. 2376–2399

A FAST ALGORITHM FOR UPDATING AND DOWNSIZING THE DOMINANT KERNEL PRINCIPAL COMPONENTS*

NICOLA MASTRONARDI†, EUGENE E. TYRTYSHNIKOV‡, AND PAUL VAN DOOREN§

Abstract. Many important kernel methods in the machine learning area, such as kernel principal component analysis, feature approximation, denoising, compression, and prediction require the computation of the dominant set of eigenvectors of the symmetric kernel Gram matrix. Recently, an efficient incremental approach was presented for the fast calculation of the dominant kernel eigenbasis. In this paper we propose faster algorithms for incrementally updating and downsizing the dominant kernel eigenbasis. These methods are well-suited for large scale problems since they are efficient in terms of both complexity and data management.

Key words. dominant eigenvalues, updating, kernel gram matrix, principal components, large scale data

AMS subject classifications. 15A06, 15A18, 65A15

DOI. 10.1137/090774422

1. Introduction.

We consider in this paper the problem of finding a low rank approximation of so-called Gram matrices, which play an important role in kernel based learning methods [28]. These arise in transforming learning algorithms into kernel representations [15, 27] in the support vector machines (SVM) framework [27, 28]. Given N data points $\mathbf{x}_i \in \mathbb{R}^p,\ i = 1, \ldots, N$, one constructs a square symmetric $N \times N$ kernel Gram matrix K with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, where the kernel $k(\cdot, \cdot)$ provides a similarity measure between pairs of data points:

$$k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R} : (\mathbf{x}_i, \mathbf{x}_j) \mapsto k(\mathbf{x}_i, \mathbf{x}_j).$$

Typical examples of such functions are

$k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$	(linear SVM),
$k(\mathbf{x}, \mathbf{y}) = (\tau + \mathbf{x}^T\mathbf{y})^d,\ \tau \in \mathbb{R}, d \in \mathbb{N}$	(polynomial SVM of degree d),
$k(\mathbf{x}, \mathbf{y}) = \exp(-\Vert\mathbf{y} - \mathbf{x}\Vert_2^2/\sigma^2),\ \sigma \in \mathbb{R}$	(radial basis function kernel),
$k(\mathbf{x}, \mathbf{y}) = \tanh(\kappa_1\mathbf{x}^T\mathbf{y} + \kappa_2),\ \kappa_1, \kappa_2 \in \mathbb{R}$	(multilayer perceptron kernel).

Since only a scalar product of two vectors is involved in the computation of each

*Received by the authors October 21, 2009; accepted for publication (in revised form) by D. A. Bini April 28, 2010; published electronically July 22, 2010. The scientific responsibility of this work rests with its authors.

http://www.siam.org/journals/simax/31-5/77442.html

†Istituto per le Applicazioni del Calcolo "M. Picone," CNR, I-70126 Sede di Bari, Italy (n.mastronardi@ba.iac.cnr.it). The work of this author was partly supported by PRIN 2008 N. 20083KLJEZ, partly by the National Research Council (CNR) of Italy under the Short Term Mobility Program, and partly by the Russian-Italian collaboration agreement between the Russian Academy of Sciences (RAS) and CNR.

‡Institute of Numerical Mathematics, Russian Academy of Sciences, 119991 Moscow, Russia, and Institute for Computational Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong (tee@inm.ras.ru). This author's work was supported by the Russian Foundation for Basic Research (RFBR 08-01-00115, RFBR/DFG 09-01-91332) and the Priority Research grant of the Mathematical Sciences Department of RAS, Russian-Italian collaboration agreement between RAS and CNR.

§Department of Mathematical Engineering, Catholic University of Louvain, B-1348 Louvain-la-Neuve, Belgium (paul.vandooren@uclouvain.be). The work of this author was partly supported by the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, and supported by CNR under the Short Term Mobility Program.

element of the matrix with the latter functions, the overall complexity of computing the Gram matrix is of the order $O(pN^2)$, where, typically, $p \ll N$. Moreover, the Gram matrix is supposed to be symmetric positive definite (SPD).

Let us denote by $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{N-1} \geq \lambda_N > 0$ the eigenvalue of K, and let us denote by $\lambda_1^{(n)} \geq \lambda_2^{(n)} \geq \cdots \geq \lambda_{n-1}^{(n)} \geq \lambda_n^{(n)} > 0$ the eigenvalues of the submatrices $K_n := K(1:n, 1:n)$, $n = 1, \ldots, N$.

Many methods in this area [11, 17, 19, 20, 21, 22, 24, 27] rely on the knowledge of the m dominant eigenvectors of K_N, with $m \ll N$. Computing this eigenspace with usual tools such as the eigenvalue decomposition [10] requires $O(N^3)$ operations. For large data sets (as, for example, in image processing, computer vision, or object recognition) an eigendecomposition of K_N can simply become too time-consuming. But also memory is an issue since $O(N^2)$ memory units are required. In addition, computations may have to be repeated several times if the problem is nonstationary.

Some applications require adaptive algorithms, and during the last three decades considerable effort was spent in dealing with such issues. The typical approach is called singular value decomposition (SVD) updating, in which the previous eigenspace is incremented when new data are incorporated in the scheme [1, 3, 4, 5, 6, 16, 15, 18]. In most of these methods the eigenspace is incrementally updated when additional rows and/or columns are added to the data set. These updating algorithms usually increase the size of the updating eigenspace by the size of added rows and columns.

In the particular context of kernel algorithms, since the kernel matrix increases both in row and column dimension when a point $\mathbf{x}_i$ is added, those updating schemes can be applied efficiently only when using the technique described in subsection 2.1 of this paper, adapted to the particular situation.

In [14], an online updating and downdating algorithm is presented for the dominant eigenspace case of a growing square symmetric matrix; it has a computational complexity of $O(nm^2 + m^3)$ and memory requirement of $O(nm)$ at the nth iteration. In [9], a modified algorithm for solving a similar problem has been proposed with essentially the same computational complexity. A so-called *"downsizing"* technique is also proposed in these two papers to maintain the number of rows of the subspace matrix fixed, say, equal to $\tilde{n}$, with an $O(\tilde{n}m^2 + m^3)$ complexity per iteration.

In this paper, we propose a new algorithm for updating with $O(nm + m^2)$ complexity and $O(nm)$ memory requirements per iteration. The idea is to compute only (at each step of the updating) the subspace associated with the eigenvalues to be discarded, which requires less work than computing the dominant subspace. We also derive an improved algorithm for downsizing the approximating subspace with a complexity of $O(\tilde{n}m+m^2)$ per iteration. The downdating procedure proposed in [14] could potentially destroy the semidefiniteness of the approximation. The proposed downdating procedure often preserves the semidefiniteness of the approximation. Moreover, it could also be applied to Laplacians of very large undirected graphs or other similar problems.

The algorithm described in this paper can be applied to handle large problems involving any SPD matrix. Moreover, modified in an appropriate way, it can be efficiently used to handle nonsymmetric problems as well, arising, e.g., in latent semantic indexing [3, 29] and subspace tracking [26]. These include updating a rank k approximation of a matrix A, when padding it with rows and/or columns, or when adding a low rank correction to it.

We also present an error tracking mechanism that provides a reasonably accurate estimate of the error norm of the computed low rank approximation. This estimate

is based on the eigenvalues that are neglected at each step of the iteration; therefore the computed estimate is obtained at a marginal additional cost. Numerical examples illustrate the performance of the estimate.

The paper is organized as follows. In section 2 we propose an updating scheme for an existing eigenspace. In section 3 we describe a downsizing algorithm inspired from [14] and show how it can be implemented in a more efficient way. In section 4 we derive some bounds for the accuracy that can be obtained with our method. In section 5 we present experiments on some typical benchmark data sets to demonstrate that accuracy and stability of the eigenvalues and eigenvectors are preserved during and after iteration. Section 6 summarizes the key results and compares the overall complexity of the different algorithms.

2. Updating procedure. In this section we describe in detail the updating for one step of the iterative procedure and show that it compares favorably to the complexity of the algorithm proposed in [14].

2.1. Basic idea. For the update we assume that we have an orthonormal basis $U_n \in \mathbb{R}^{n\times m}$ of the dominant eigenspace of the square symmetric matrix $A_n \in \mathbb{R}^{n\times n}$, which is a symmetric positive semidefinite rank m approximation of the Gram submatrix K_n, based on the n first data points.

The goal of updating is to obtain the $(n+1)\times m$ eigenspace matrix U_{n+1} of $A_{n+1} \in \mathbb{R}^{(n+1)\times(n+1)}$, a rank m approximation of the Gram submatrix K_{n+1}, based on the n first data points and one extra data point $\mathbf{x}_{n+1}$. To this end, first, a matrix $\hat{A}_{n+1}$ is constructed, appending one more row and column to A_n:

$$\hat{A}_{n+1} = \begin{bmatrix} A_n & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix}, \tag{2.1}$$

where $\mathbf{a} \in \mathbb{R}^n$ is the vector of kernel entries $a_i = k(\mathbf{x}_i, \mathbf{x}_{n+1})$, $i = 1, \ldots, n$, and b is the scalar $k(\mathbf{x}_{n+1}, \mathbf{x}_{n+1})$. The matrix $\hat{A}_{n+1}$ has rank $m+2$, at most. Then, A_{n+1} is computed as the optimal approximation of $\hat{A}_{n+1}$ in the class of symmetric positive semidefinite matrices of rank m, and that is *not* necessarily the best approximation in the (larger) class of symmetric matrices of rank m.

In [14], $\hat{A}_{n+1}$ is written as a rank 2 modification of A_n bordered with zeros:

$$\hat{A}_{n+1} = \begin{bmatrix} A_n & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \mathbf{a} \\ \frac{b}{2}+1 \end{bmatrix}\begin{bmatrix} \mathbf{a} \\ \frac{b}{2}+1 \end{bmatrix}^T - \frac{1}{2}\begin{bmatrix} \mathbf{a} \\ \frac{b}{2}-1 \end{bmatrix}\begin{bmatrix} \mathbf{a} \\ \frac{b}{2}-1 \end{bmatrix}^T, \tag{2.2}$$

where $\mathbf{0}$ is the zero vector of length n. This is then exploited to derive an algorithm of $O(nm^2 + m^3)$ complexity to compute the eigendecomposition of (2.2) and derive a new rank m approximation A_{n+1}. In this paper we describe an algorithm computing the same eigenspace with $O(nm + m^2)$ complexity.

Let A_n be a rank m matrix, $m < n$,

$$A_n = U_n M_n U_n^T,$$

with $M_n \in \mathbb{R}^{m\times m}$ SPD and $U_n \in \mathbb{R}^{n\times m}$ with orthonormal columns. Let

$$M_n = Q_m D_m Q_m^T \tag{2.3}$$

be the eigenvalue decomposition of M_n, where $D_m = \operatorname{diag}(\mu_1^{(n)}, \mu_2^{(n)}, \ldots, \mu_m^{(n)})$, $\mu_1^{(n)} \geq \mu_2^{(n)} \geq \cdots \geq \mu_m^{(n)} > 0$ (we will drop the index and use μ_i instead of $\mu_i^{(n)}$

when it is obvious from the context that we are at step n) and $Q_m^T Q_m = \mathbb{I}_m$. Also let

$$\mathbf{r} = U_n^T \mathbf{a}, \tag{2.4}$$

$$\mathbf{q} = (\mathbb{I}_n - U_n U_n^T)\mathbf{a} = \mathbf{a} - U_n \mathbf{r}, \tag{2.5}$$

$$\rho = \|\mathbf{q}\|_2, \tag{2.6}$$

$$\mathbf{u}_\perp = \mathbf{q}/\rho. \tag{2.7}$$

These computations correspond to the Gram–Schmidt orthogonalization [25] of the matrix $[\, U_n \mid \mathbf{a} \,]$ and require $4nm$ operations:

$$[\, U_n \mid \mathbf{a} \,] = [\, U_n \mid \mathbf{u}_\perp \,] \left[\begin{array}{c|c} \mathbb{I}_m & \mathbf{r} \\ \hline & \rho \end{array}\right].$$

In order to avoid loss of accuracy, the Gram–Schmidt orthogonalization can be performed twice [8], doubling the cost. If we replace

$$\mathbf{a} = U_n U_n^T \mathbf{a} + (\mathbb{I}_n - U_n U_n^T)\mathbf{a} = U_n \mathbf{r} + \rho \mathbf{u}_\perp,$$

then by (2.4), (2.5), (2.6), (2.7) we can write (2.1) as

$$\hat{A}_{n+1} = \left[\begin{array}{c|c} A_n & \mathbf{a} \\ \hline \mathbf{a}^T & b \end{array}\right] = \hat{U}\hat{M}\hat{U}^T, \tag{2.8}$$

where

$$\hat{U} := \left[\begin{array}{c|c} U_n \,|\, \mathbf{u}_\perp & \\ \hline & 1 \end{array}\right] \quad \text{and} \quad \hat{M} := \left[\begin{array}{c|c|c} M_n & & \mathbf{r} \\ \hline & 0 & \rho \\ \hline \mathbf{r}^T & \rho & b \end{array}\right]. \tag{2.9}$$

Let

$$\hat{M} = \hat{Q}_{m+2}\hat{D}_{m+2}\hat{Q}_{m+2}^T \tag{2.10}$$

be the spectral decomposition of $\hat{M}$, where $\hat{Q}_{m+2} \in \mathbb{R}^{(m+2)\times(m+2)}$ is the orthogonal matrix of eigenvectors of $\hat{M}$ and

$$\hat{D}_{m+2} = \operatorname{diag}(\hat{\mu}_1, \hat{\mu}_2, \ldots, \hat{\mu}_{m+1}, \hat{\mu}_{m+2}).$$

It then follows from the interlacing property [10] that

$$\hat{\mu}_1 \geq \mu_1 \geq \hat{\mu}_2 \geq \mu_2 \geq \cdots \geq \mu_m \geq \hat{\mu}_{m+1} \geq 0 \geq \hat{\mu}_{m+2}. \tag{2.11}$$

Since the order of $\hat{M}$ is $m+2$, its spectral decomposition can be computed with $O(m^3)$ floating point operations. In what follows, we will consider a technique to reduce this cost.

Replacing (2.9) and (2.10) in (2.8), we obtain

$$\hat{A}_{n+1} = \hat{U}\hat{Q}_{m+2}\hat{D}_{m+2}\hat{Q}_{m+2}^T\hat{U}^T. \tag{2.12}$$

The new updated dominant subspace U_{n+1} is obtained in [14] by multiplying $\hat{U}$ by $\hat{Q}_{m+2}$ and neglecting the last two columns, corresponding to the two smallest eigenvalues $\hat{\mu}_{m+1}$ and $\hat{\mu}_{m+2}$ of $\hat{M}$ with a cost of $2nm^2$ operations.

We now show a different way to compute U_{n+1}. We observe that it is not necessary to compute the whole spectral decomposition of the matrix $\hat{M}$. The idea is to compute only the subspace of $\mathbb{R}^{m+2}$ spanning the last two columns of $\hat{Q}_{m+2}$, i.e., the subspace spanned by the eigenvectors associated with the smallest (i.e., the most negative) two eigenvalues $\hat{\mu}_{m+1}$ and $\hat{\mu}_{m+2}$, and two orthogonal transformations H_1 and H_2 such that

$$H_2 H_1 \hat{Q}_{m+2} \begin{bmatrix} 0 \\ \mathbb{I}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ E \end{bmatrix}, \quad \text{where } E := \operatorname{diag}(\mp 1, \mp 1); \tag{2.13}$$

i.e., the orthogonal matrices H_1 and H_2 transform the eigenvectors associated to the eigenvalues $\hat{\mu}_{m+1}$ and $\hat{\mu}_{m+2}$ to the last two vectors of the canonical basis of $\mathbb{R}^{(m+2)}$. Then,

$$H_2 H_1 \hat{M} H_1^T H_2^T = \left[\begin{array}{c|cc} M_{n+1} & & \\ \hline & \hat{\mu}_{m+1} & \\ & & \hat{\mu}_{m+2} \end{array}\right],$$

with $M_{n+1} \in \mathbb{R}^{m \times m}$. Hence, (2.12) becomes

$$\hat{A}_{n+1} = \hat{U} {H_1}^T {H_2}^T \left[\begin{array}{c|cc} M_{n+1} & & \\ \hline & \hat{\mu}_{m+1} & \\ & & \hat{\mu}_{m+2} \end{array}\right] H_2 H_1 \hat{U}^T.$$

Therefore, the new updated decomposition is given by

$$A_{n+1} = U_{n+1} M_{n+1} U_{n+1}^T,$$

with U_{n+1} given by the first m columns of the product $\hat{U} H_1^T H_2^T$.

Remark 1. Each eigenvalue $\mu_i^{(n)}$ of A_n is a nondecreasing function of n. This follows from $\mu_i^{(n+1)} = \hat{\mu}_i,\ i = 1, \ldots, m$, and the interlacing property (2.11).

Each matrix $H_i,\ i = 1, 2$, can be chosen as either a Householder matrix or a sequence of $m + 1$ Givens rotations such that (2.13) holds. With such choices, the updated matrix U_{n+1} is computed with $4mn$ and $6mn$ operations, respectively. Modified Givens rotations [12] can be used instead of Givens rotations in order to reduce the computational complexity to $4mn$.

In (2.13), only the knowledge of the subspace of $\mathbb{R}^{m+2}$ spanned by the eigenvectors corresponding to $\hat{\mu}_{m+1}$ and $\hat{\mu}_{m+2}$ is required.

In what follows, a method to compute this subspace is proposed with a lower computational complexity. At each updating step, the matrix M_n is kept in factorized form. In particular, we will use the Cholesky factorization of M_n.

2.2. Using the Cholesky factorization. Let

$$M_n = L_n L_n^T \tag{2.14}$$

be the Cholesky factorization of the SPD matrix M_n, where $L_n \in \mathbb{R}^{m \times m}$. Replacing (2.14) in (2.9), we have

$$\hat{M} = \left[\begin{array}{c|c|c} L_n L_n^T & & \mathbf{r} \\ \hline & 0 & \rho \\ \hline \mathbf{r}^T & \rho & b \end{array}\right] = \hat{L} \hat{D} \hat{L}^T, \tag{2.15}$$

with

(2.16)
$$\hat{L} := \left[\begin{array}{c|c} L_n & \\ \hline \mathbf{0}_m^T & \\ \hline \mathbf{t}^T & \mathbb{I}_2 \end{array}\right], \quad \hat{D} := \left[\begin{array}{c|c} \mathbb{I}_m & \\ \hline & S_c \end{array}\right], \quad S_c = \left[\begin{array}{c|c} 0 & \rho \\ \hline \rho & b - \mathbf{t}^T\mathbf{t} \end{array}\right], \quad \mathbf{t} = L_n^{-1}\mathbf{r}.$$

Due to the lower triangular structure of L_n, the linear system $L_n\mathbf{t} = \mathbf{r}$ is solved with m^2 floating point operations by forward substitution [10]. Each updating iteration by using the Cholesky factorization can be divided into the following three steps.

1. Computation of the matrix $\hat{M}$ by means of the procedure described in the previous section.
2. Computation of the subspace spanned by the eigenvectors corresponding to the smallest eigenvalues $\hat{\mu}_{m+1}$ and $\hat{\mu}_{m+2}$ of $\hat{M}$.
3. Reduction of the dimension of the updated subspace neglecting columns $m+1$ and $m+2$.

The subspace corresponding to the smallest eigenvalues $\hat{\mu}_{m+1}$ and $\hat{\mu}_{m+2}$ can be retrieved by applying a few steps of inverse iteration without shift to the matrix $\hat{M}$, factorized as in (2.15), starting from a couple of orthogonal vectors as initial guesses. It turns out that the most efficient choices for the initial vectors at the current iteration are the eigenvectors corresponding to the two eigenvalues $\hat{\mu}_{m+1}$ and $\hat{\mu}_{m+2}$ neglected at the previous iteration. At the ith step of inverse iteration, for each of the two vectors of the approximate invariant subspace $\mathbf{x}_0^{(i)}$ and $\mathbf{y}_0^{(i)}$, the following three structured linear systems must be solved:

$$\hat{L}\mathbf{x}_1^{(i)} = \mathbf{x}_0^{(i)}, \quad \hat{L}\mathbf{y}_1^{(i)} = \mathbf{y}_0^{(i)}, \tag{2.17}$$

$$\hat{D}\mathbf{x}_2^{(i)} = \mathbf{x}_1^{(i)}, \quad \hat{D}\mathbf{y}_2^{(i)} = \mathbf{y}_1^{(i)}, \tag{2.18}$$

$$\hat{L}^T\mathbf{x}_3^{(i)} = \mathbf{x}_2^{(i)}, \quad \hat{L}^T\mathbf{y}_3^{(i)} = \mathbf{y}_2^{(i)}. \tag{2.19}$$

Each system in (2.17) and (2.19) requires m^2 operations to be solved, due to the lower triangular structure of $\hat{L}$. Each system in (2.18) requires a constant number of operations, since the matrix $\hat{D}$ differs from the identity matrix only for the 2×2 block in the lower-right corner. One then also orthogonalizes the two computed vectors $\mathbf{x}_3^{(i)}$ and $\mathbf{y}_3^{(i)}$ against each other with a linear cost in m. Hence, each step of inverse iteration costs a total of $4m^2$ operations.

The last step is the updating of the matrix $\hat{U}$ by means of the two left multiplications by orthogonal matrices. In order to keep the triangular structure of the involved matrices, the two orthogonal matrices are chosen as the product of $m+1$ Givens rotations. We shortly describe how this process works.

Let $V \in \mathbb{R}^{(m+2)\times 2}$ be an orthogonal matrix spanning the invariant subspace generated by the eigenvectors of $\hat{M}$ corresponding to $\hat{\mu}_{m+1}$ and $\hat{\mu}_{m+2}$. In order to save some computation, we choose V with the entry $(1,1)$ equal to 0. This matrix can be always obtained by applying a Givens rotation G_0 to the right of V. Then, a Givens rotation $\hat{G}_1$ is applied to the left of V, acting on its second and third rows, to annihilate $V(2,1)$. Hence, $\hat{A}_{n+1}$ becomes

$$\hat{A}_{n+1} = \hat{U}\hat{L}\hat{D}\hat{L}\hat{U}^T = \left(\hat{U}\hat{G}_1^T\right)\left(\hat{G}_1\hat{L}\right)\hat{D}\left(\hat{L}\hat{G}_1^T\right)\left(\hat{G}_1\hat{U}^T\right). \tag{2.20}$$

The multiplication $\hat{G}_1\hat{L}$ modifies the second and third rows of $\hat{L}$, creating a nonzero entry in position $(2,3)$. One then applies a Givens rotation $\tilde{G}_1^T$ to the right of $\hat{G}_1\hat{L}$

in order to annihilate the nonzero entry, and (2.20) becomes

$$\hat{A}_{n+1} = \left(\hat{U}\hat{G}_1^T\right)\left(\hat{G}_1\hat{L}\tilde{G}_1^T\right)\left(\tilde{G}_1\hat{D}\tilde{G}_1^T\right)\left(\tilde{G}_1\hat{L}\hat{G}_1^T\right)\left(\hat{G}_1\hat{U}^T\right),$$

which completes the first step. We observe that $\tilde{G}_1\hat{D}\tilde{G}_1^T = \hat{D}$ for $m \geq 3$, since the first principal block of $\hat{D}$ is the identity matrix of order m.

All steps for $i = 1, \ldots, m$ are similar: a Givens rotation $\hat{G}_i$ is applied to the left of $\hat{L}$, creating a nonzero entry in position $(i+1, i+1)$, and its transpose $\hat{G}_i^T$ is applied to the right of $\hat{U}$. This nonzero entry is then annihilated by a Givens rotation $\tilde{G}_i$ applied to the right of $\hat{L}$, and its transpose $\tilde{G}_i^T$ is applied to the left of $\hat{D}$. After m steps, the first column of V becomes equal to $\mathbf{e}_{m+2}$ and, consequently, the last entry of the second column of V is 0.

The whole procedure for annihilating all the entries of the first column of V but the last one for lower triangular matrices of size 6 is depicted in Figure 2.1. Another m Givens rotation $\hat{G}_i$, $i = m+1, \ldots, 2m$, must then be applied to the left of the second column of V to annihilate its first m entries. If we denote the products of these Givens transformations by

$$\Gamma_1 := \hat{G}_{2m}\hat{G}_{2m-1}\cdots\hat{G}_2\hat{G}_1, \quad \Gamma_2 := \tilde{G}_{2m}\tilde{G}_{2m-1}\cdots\tilde{G}_2\tilde{G}_1, \tag{2.21}$$

then at the end of this procedure the matrix V is transformed into the last two columns of the identity matrix of order $m+2$. Moreover, $\hat{A}_{n+1}$ is given by

$$\hat{A}_{n+1} = \left(\hat{U}\Gamma_1^T\right)\left(\Gamma_1\hat{L}\Gamma_2^T\right)\left(\Gamma_2\hat{D}\Gamma_2^T\right)\left(\Gamma_2\hat{L}\Gamma_1^T\right)\left(\Gamma_1\hat{U}^T\right) \tag{2.22}$$

and the matrix $\Gamma_2\hat{D}\Gamma_2^T$ has the following structure:

$$\Gamma_2\hat{D}\Gamma_2^T = \left[\begin{array}{c|c|c} \mathbb{I}_{m-2} & & \\ \hline & T_{11} & T_{12} \\ \hline & T_{12} & T_{22} \end{array}\right], \tag{2.23}$$

with $T_{ij} \in \mathbb{R}^{2\times 2}$, $i, j = 1, 2$, and T_{11} an SPD matrix, since the first principal matrix of $\hat{M}$ is SPD. The structure of the initial $\hat{L}\hat{L}^T$ is retrieved by computing the Cholesky factorization of T_{11}:

$$T_{11} = L_2 L_2^T, \tag{2.24}$$

with $L_2 \in \mathbb{R}^{2\times 2}$. Since one must delete the last two rows and columns of $\hat{M}$, the new matrix L_{n+1} is then given by

$$L_{n+1} := \left[\begin{array}{c|c} \mathbb{I}_m & \mathbf{0} \end{array}\right]\Gamma_1\hat{L}\Gamma_2^T\left[\begin{array}{c|c} \mathbb{I}_{m-2} & \\ \hline & L_2 \end{array}\right], \tag{2.25}$$

and the new updated matrix U_{n+1} then consists of the first m columns of the orthogonal matrix

$$U_{n+1} := \hat{U}\Gamma_1^T\left[\begin{array}{c} \mathbb{I}_m \\ \hline \mathbf{0} \end{array}\right]. \tag{2.26}$$

This computation requires $12nm$ operations, but the complexity can again be reduced to $8nm$ if the Givens rotations are replaced by modified Givens rotations [12].

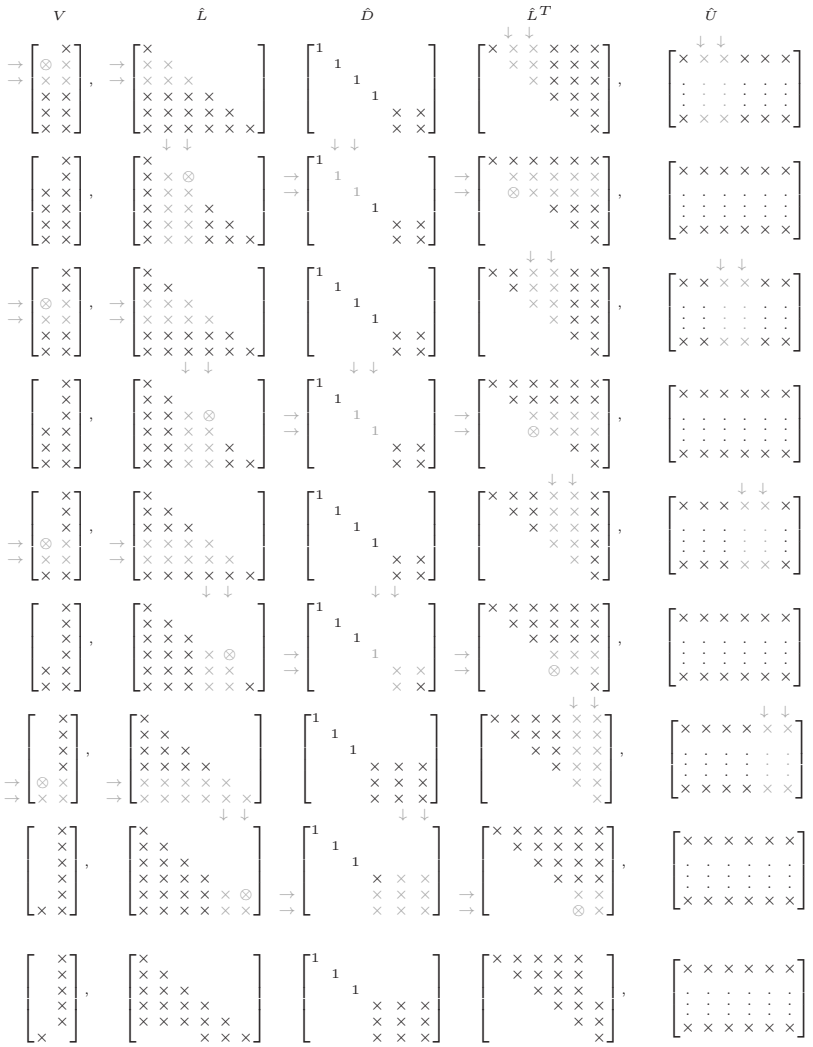

FIG. 2.1. *Effect of Givens rotations on V, $\hat{L}$ $\hat{D}$ $\hat{L}^T$, and $\hat{U}$ (part 1). A Givens rotation is applied to the matrix V spanning the subspace generated by the eigenvectors associated to the smallest eigenvalues (gray arrows pointing to the first matrix in each row of the figure) in order to annihilate one entry, denoted by "⊗." The same matrix acts to the left of $\hat{L}$ and to the right of $\hat{U}$. This creates a nonzero entry in the matrix $\hat{L}$ to be annihilated by another Givens rotation. $\hat{D}$ is updated accordingly.*

Remark 2. This reduction step can also be made with complexity $8nm$ without using modified Givens rotations by keeping the matrix $\hat{U}$ in factorized form at each iteration.[1] In particular, we consider the factorization $U_n = Y_n^{(1)} Y_n^{(2)}$, with $Y_n^{(1)} \in \mathbb{R}^{n\times m}$, $Y_n^{(2)} \in \mathbb{R}^{m\times n}$, $Y_n^{(1)}, Y_n^{(2)}$ orthogonal (at the first iteration we choose $Y_n^{(1)} = U_n$,

[1] A similar factorization was introduced independently in [3].

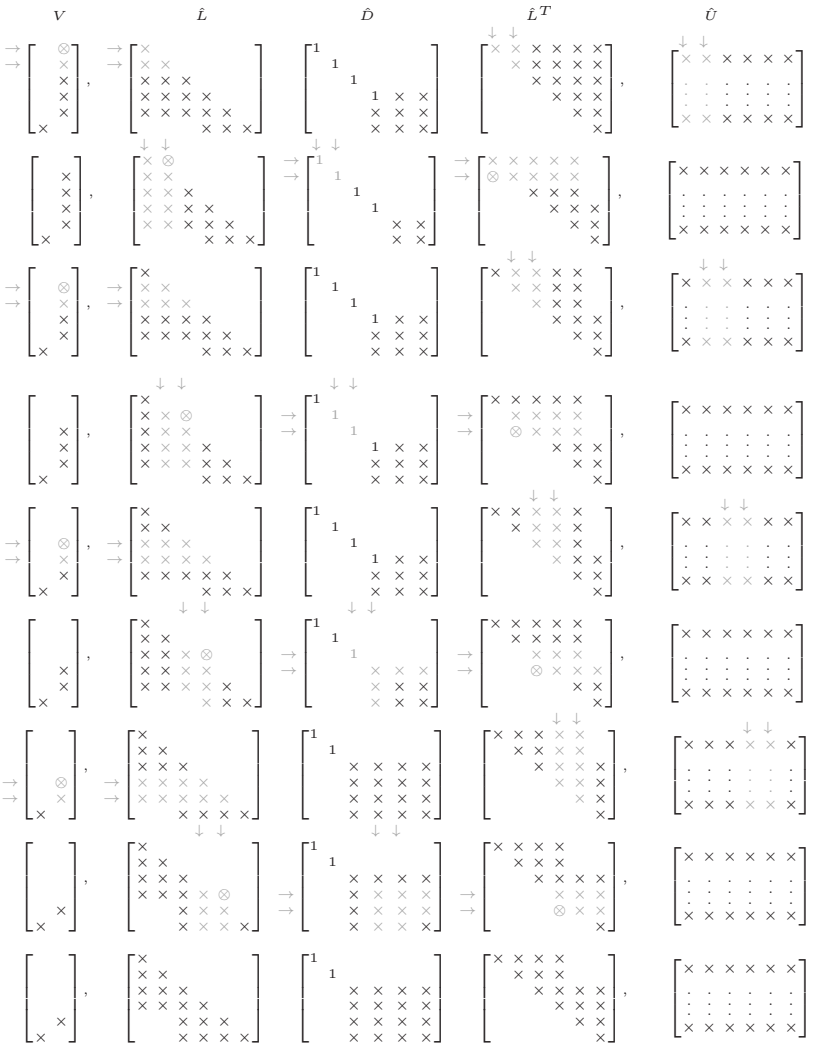

FIG. 2.2. *Effect of Givens rotations on V, $\hat{L}$ $\hat{D}$ $\hat{L}^T$, and $\hat{U}$ (part 2).*

$Y_n^{(2)} = \mathbb{I}_m$). Thus, $\hat{U}$ becomes

$$\hat{U} = \left[\begin{array}{c|c|c} Y_n^{(1)} & \mathbf{u}_\perp & \\ \hline & & 1 \end{array}\right] \left[\begin{array}{c|c|c} Y_n^{(2)} & & \\ \hline & 1 & \\ \hline & & 1 \end{array}\right] = \hat{Y}_n^{(1)} \hat{Y}_n^{(2)}.$$

Factoring $\hat{U}$ in this way, the sequence of the $2m$ Givens rotation Γ_1 (2.21) acts only on $\hat{Y}_n^{(2)}$, i.e.,

$$\hat{U}\Gamma_1^T = \hat{Y}_n^{(1)} \left(\hat{Y}_n^{(2)} \Gamma_1^T \right),$$

with cost $12m^2$. To retrieve the subspace associated with the largest m eigenvalues of $\hat{A}_m$, it is sufficient to consider two Householder matrices of order $m+2$, H_1, and H_2,

such that

$$H_2 H_1 \left(\hat{Y}_n^{(2)} \Gamma_1^T\right) \begin{bmatrix} 0 \\ \mathbb{I}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ E \end{bmatrix}, \quad \text{where} \quad E = \operatorname{diag}(\mp 1, \mp 1),$$

with cost $8m^2$. Then the left factor of (2.26) can be written as

$$\hat{U}\Gamma_1^T = \hat{Y}_n^{(1)} \hat{Y}_n^{(2)} \Gamma_1^T = \left(\hat{Y}_n^{(1)} H_1^T H_2^T\right) \left(H_2 H_1 \hat{Y}_n^{(2)} \Gamma_1^T\right).$$

The subspace associated with the largest m eigenvalues of $\hat{A}_m$ is given by the first m columns of the right multiplication of $\hat{Y}_n^{(1)}$ by the Householder matrices H_1 and H_2, with the higher order term of the complexity equal to $8mn$.

3. Downsizing. For each update of the matrix n, the number of rows of the eigenspace matrix U_n increases. This is not practical for on-line applications, where the dimension of the matrices should remain reasonably bounded. This means that we should keep the size $n = \tilde{n}$ and downsize the matrix after each update, while also preserving orthogonality. In [14], a scheme is proposed to reduce the rows of the updated matrix U_{n+1} by one and recover orthogonality of the eigenspace matrix, but its complexity is $O(\tilde{n}m^2 + m^3)$. We show that this computation can be reduced to $O(\tilde{n}m + m^2)$ by exploiting the properties of the involved matrices. Moreover, instead of removing the first row of the matrix U_{n+1} in the downsizing procedure, we choose to remove the row of minimum norm and show the advantages of this choice.

First we briefly describe the downsizing method described in [14].

In downsizing, the first row $\mathbf{v}$ of the $(n+1) \times m$ eigenspace matrix U_{n+1} must be deleted, such that we can continue with the remaining part V of size $n \times m$:

$$U_{n+1} = \begin{bmatrix} \mathbf{v}^T \\ V \end{bmatrix}.$$

The columns of V are in general not orthonormal. The following procedure has been proposed in [14] to compute a matrix U_n spanning the same space as V and having orthonormal columns. Let

$$\mathbb{I}_m = U_{n+1}^T U_{n+1} = \begin{bmatrix} \mathbf{v} & V^T \end{bmatrix} \begin{bmatrix} \mathbf{v}^T \\ V \end{bmatrix} = V^T V + \|\mathbf{v}\|_2^2 \tilde{\mathbf{v}} \tilde{\mathbf{v}}^T, \tag{3.1}$$

with $\tilde{\mathbf{v}} = \mathbf{v}/\|\mathbf{v}\|_2$. A column transformation N of order m is sought in order to orthogonalize V, i.e.,

$$U_n = VN \ \text{ with } \ U_n^T U_n = \mathbb{I}_m. \tag{3.2}$$

Replacing (3.2) in (3.1) yields

$$\mathbb{I}_m = N^T V^T V N = N^T (\mathbb{I}_m - \|\mathbf{v}\|_2^2 \tilde{\mathbf{v}} \tilde{\mathbf{v}}^T) N.$$

The matrix N is therefore obtained from the eigendecomposition of $\mathbb{I}_m - \|\mathbf{v}\|_2^2 \tilde{\mathbf{v}} \tilde{\mathbf{v}}^T = ZSZ^T$, which is given by

$$S = \operatorname{diag}\left(1 - \|\mathbf{v}\|_2^2, \underbrace{1, \ldots, 1}_{m-1}\right), \quad Z = \left[\begin{array}{c|c} \tilde{\mathbf{v}} & \tilde{\mathbf{V}}_\perp \end{array}\right],$$

where $\tilde{\mathbf{V}}_\perp \in \mathbb{R}^{m\times(m-1)}$ is the orthogonal complement of $\tilde{\mathbf{v}} \in \mathbb{R}^m$. In [14], the computation of $\tilde{\mathbf{V}}_\perp$ is made with $O(m^3)$ floating point operations. Furthermore, it turns out that

$$N = ZS^{-1/2}.$$

To complete the downsizing step, one replaces the matrix U_{n+1} by

$$U_n = VN. \tag{3.3}$$

In [14], forming this product has $O(nm^2)$ complexity.

In what follows, an equivalent algorithm is proposed with $O(nm)$ complexity. Moreover, we show that the proposed algorithm has numerical advantages. Let H_3 be a Householder matrix such that

$$H_3\mathbf{v} = [\, \upsilon\ 0\ \cdots\ 0\,], \qquad \upsilon = \mp\|\mathbf{v}\|_2. \tag{3.4}$$

Then

$$U_{n+1}H_3 = \left[\begin{array}{c|ccc} \upsilon & 0 & \cdots & 0 \\ \hline & & VH_3 & \end{array}\right]. \tag{3.5}$$

The columns of VH_3 are mutually orthogonal. To retrieve orthonormality, it is sufficient to divide the first column of VH_3 by $\sqrt{1-\upsilon^2}$ and, correspondingly, multiply the first column and row of M_n by the same quantity. If the matrix M_n is factored as $L_nL_n^T$ by the Cholesky decomposition, this operation reduces to multiplying the entry $(1,1)$ of L_n by $\sqrt{1-\upsilon^2}$.

We observe that any row of U_{n+1} can be chosen to be removed. In particular, by choosing the one with smallest 2-norm, we get rid of the row (and corresponding column) of smallest energy in $\hat{A}_{n+1}$. Notice that after the downsizing procedure, the positive definiteness of the matrix M_n could be lost. The procedure that we recommend, namely to remove the row $\mathbf{v}$ of smallest norm in U_{n+1}, guarantees that this will in fact not happen, as the eigenvalue matrix S indicates. Indeed, this matrix will be singular only if $\|\mathbf{v}\|_2 = 1$ but this cannot happen in an orthonormal matrix U_{n+1} with more rows than columns.

Remark 3. The proposed downsizing procedure works well if the relevant information of the matrix is contained in a restricted number of rows and columns, i.e., only some rows and columns are "dominant" and the rest of the matrix can be considered noise. In case the information is distributed over the whole matrix, the proposed downsizing procedure will give results similar to those delivered by the downsizing procedure in [14].

The process to get rid of a row $\mathbf{v}$ (say, the first row) of U_{n+1}, with M_{n+1} factored as $L_{n+1}L_{n+1}^T$, is graphically depicted in Figure 3.1. In each step, we eliminate one of the $m-1$ elements of the row $\mathbf{v}$ of U_{n+1} by a Givens rotation $\hat{G}_i$. When applying $\hat{G}_i^T$ to the rows of L_{n+1}, this creates a nonzero entry that is subsequently eliminated by a Givens rotation $\tilde{G}_i$, applied to the columns of L_{n+1}.

Notice that we used in this section the same notation for a different matrix since $U_nM_nU_n^T$ is an approximation of another $n\times n$ submatrix of K_N.

We now recapitulate the different steps discussed in the previous sections in a single algorithm, with input K_N, sizes $m, \tilde{n}$, and a Boolean *Down*, indicating whether or not we need to maintain a fixed dimension $\tilde{n}$.

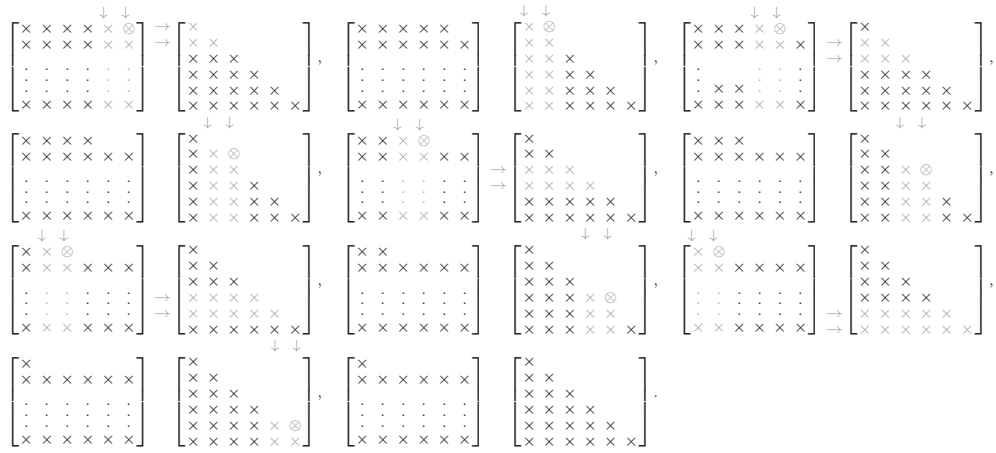

FIG. 3.1. *Effect of downsizing on the matrices U_{n+1} and L_{n+1}. A sequence of Givens rotations is applied to U_{n+1} annihilating the entries denoted by "$\otimes$." The elements modified by the Givens rotations in U_{n+1} and L_{n+1} are marked in gray.*

`function` $[U, L]$ =`Dominant`$(K, m, \tilde{n}, Down)$;
Compute the best rank m approximation UMU^T of $K_{\tilde{n}}$.
for $n = \tilde{n} + 1 : N$,
  Compute the Gram–Schmidt orthogonalization (2.4)–(2.7).
  Compute the eigenspace V for the smallest 2 eigenvalues (2.17)–(2.19).
  Compute the transformations H_1 and H_2 as in (2.21)–(2.24).
  Update U and L with H_1 and H_2 as in (2.25)–(2.26).
  if Down **then** Find row **v** of least norm in current U;
    Compute H_3 and υ as in (3.4)–(3.5).
    Update U and L with H_3 and υ as in Figure 3.1.
  end if
end for

4. Accuracy bounds. In this section we provide some bounds on the error of the low rank approximation of the kernel matrix. For this we study the local approximation errors and show that these can be used to provide estimates for the global error at the end of the algorithm. Let $K_N \in \mathbb{R}^{N\times N}$ be an SPD matrix with eigenvalue decomposition

$$K_N = V_N \Lambda_N V_N^T,$$

with $\Lambda_N = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N > 0$. Let K_n be any $n \times n$ principal submatrix of order n with eigenvalue decomposition

$$K_n = V_n \Lambda_n^{(n)} V_n^T,$$

with $\Lambda_n^{(n)} = \mathrm{diag}(\lambda_1^{(n)}, \ldots, \lambda_n^{(n)})$, $\lambda_1^{(n)} \geq \lambda_2^{(n)} \geq \cdots \geq \lambda_n^{(n)} \geq 0$.

It is well known, by the Cauchy interlacing theorem [10], that

$$\lambda_i^{(n+1)} \geq \lambda_i^{(n)} \quad \text{and} \quad \lambda_i^{(n)} \geq \lambda_{i+N-n}, \quad i = 1, \ldots, n. \tag{4.1}$$

Since $\lambda_N > 0$, we also have $\lambda_n^{(n)} > 0$. Moreover, each eigenvalue $\lambda_i^{(n)}$ is a nondecreasing function of n.

In the updating scheme, we approximated K_n by a rank m approximation

$$K_n \approx A_n = U_n M_n U_n^T, \quad U_n^T U_n = \mathbb{I}_m, \quad M_n = M_n^T > 0$$

which was obtained recursively for $n = \tilde{n}+1, \ldots, N$ (with $\tilde{n} > m$). The first approximation $A_{\tilde{n}}$ is assumed to be an optimal rank m approximation of $K_{\tilde{n}}$. All next approximations were obtained by solving a local minimization problem at each iteration step n using the bordered matrix problem

$$\min \left\| \left[\begin{array}{c|c} U_n M_n U_n^T & \mathbf{a}_n \\ \hline \mathbf{a}_n^T & b_n \end{array}\right] - U_{n+1} M_{n+1} U_{n+1}^T \right\|_F^2 \tag{4.2}$$

such that $M_{n+1} > 0$ and $U_{n+1}^T U_{n+1} = \mathbb{I}_m$, where $\mathbf{a}_n, b_n$ are the elements of the bordered matrix

$$K_{n+1} = \left[\begin{array}{c|c} K_n & \mathbf{a}_n \\ \hline \mathbf{a}_n^T & b_n \end{array}\right].$$

We indicate here that (4.2) always has a unique solution in the Frobenius norm and that it also minimizes the 2-norm of (4.2).

LEMMA 4.1. *Let*

$$\hat{A}_{n+1} := \left[\begin{array}{c|c} A_n & \mathbf{a}_n \\ \hline \mathbf{a}_n^T & b_n \end{array}\right]$$

be such that $A_n = A_n^T \geq 0$ has rank m. Then the optimal semidefinite rank m approximation in the Frobenius norm is unique and is given by

$$A_{n+1} := U_{n+1} M_{n+1} U_{n+1}^T, \quad U_{n+1}^T U_{n+1} = \mathbb{I}_m, \quad M_{n+1} = M_{n+1}^T > 0,$$

where the eigenvalues of M_{n+1} are the most positive eigenvalues of the bordered matrix. A_n is also an optimal semidefinite approximation in the 2-norm.

Proof. By the Cauchy interlacing theorem, we know that the eigenvalues of $\hat{A}_{n+1}$ and A_n interlace, but, since A_n has m strictly positive eigenvalues $\mu_1, \ldots, \mu_m$ (and $n-m$ zero ones), $\hat{A}_{n+1}$ must have at least m strictly positive eigenvalues $\hat{\mu_1}, \ldots, \hat{\mu}_m$. Since $n > m$ then there are $n-1-m$ roots equal to zero and two roots $\hat{\mu}_{m+1}$ and $\hat{\mu}_{n+1}$ that are possibly nonzero and satisfy

$$\hat{\mu}_1 \geq \mu_1 \geq \cdots \geq \hat{\mu}_m \geq \mu_m \geq \hat{\mu}_{m+1} \geq \hat{\mu}_{m+2} = \cdots = \hat{\mu}_n = 0 \geq \hat{\mu}_{n+1}.$$

It now follows from [13] that the closest rank m matrix in the set of semidefinite matrices is obtained by putting $\hat{\mu}_{m+1}$ and $\hat{\mu}_{n+1}$ equal to zero in the decomposition of $\hat{A}_{n+1}$. This will also be the closest rank m matrix without imposing a constraint of semidefiniteness, iff $\hat{\mu}_{n+1} \leq \hat{\mu}_m$. (Note that in subsection 2.1, $\hat{\mu}_{n+1}$ was called $\hat{\mu}_{m+2}$ since we dismissed the $n-m$ zero eigenvalues.) ∎

Remark 4. We can show that in the first iteration the approximation $A_{\tilde{n}+1}$ is also the best rank m approximation of $\hat{A}_{\tilde{n}+1}$, without imposing the constraint that $A_{\tilde{n}+1}$ is semidefinite. We assumed that $K_{\tilde{n}} = A_{\tilde{n}} + E_{\tilde{n}}$ with $E_{\tilde{n}} > 0$ and $\|E_{\tilde{n}}\|_2 = \lambda_{m+1}^{(\tilde{n})}$. Since $K_{\tilde{n}+1}$ is SPD and

$$\hat{A}_{\tilde{n}+1} = K_{\tilde{n}+1} - \left[\begin{array}{c|c} E_{\tilde{n}} & \mathbf{0} \\ \hline \mathbf{0}^T & 0 \end{array}\right],$$

its negative eigenvalue $\hat{\mu}_{m+2}$ is bounded in magnitude by $\|E_{\tilde{n}}\|_2 = \lambda_{m+1}^{(\tilde{n})}$. But this implies $|\hat{\mu}_{m+2}| \leq \lambda_m^{(\tilde{n})} = \mu_m^{(\tilde{n})} \leq \mu_m^{(\tilde{n}+1)} = \hat{\mu}_m$.

Such a result will not be obtained for the later steps, and hence it is not clear that we are obtaining the best rank m approximation in all steps. In our simulations this was always the case, but in general one cannot guarantee such a property.

We now try to bound the errors $\|A_n - K_n\|_2$ and $\|A_n - K_n\|_F$ at each step. For the initial step $\tilde{n}$, we clearly have

$$\|A_{\tilde{n}} - K_{\tilde{n}}\|_2 = \|E_{\tilde{n}}\|_2, \qquad \|A_{\tilde{n}} - K_{\tilde{n}}\|_F = \|E_{\tilde{n}}\|_F.$$

The next bounds are derived recursively. We have that A_{n+1} is a rank 2 correction to $\hat{A}_{n+1}$:

$$A_{n+1} = \left[\begin{array}{c|c} A_n & \mathbf{a}_n \\ \hline \mathbf{a}_n^T & b_n \end{array}\right] - \mathbf{v}_{n+1}\delta_{n+1}^{(+)}\mathbf{v}_{n+1}^T + \mathbf{w}_{n+1}\delta_{n+1}^{(-)}\mathbf{w}_{n+1}^T,$$

with

$$\delta_{n+1}^{(+)} := \hat{\mu}_{m+1} \geq 0, \quad \delta_{n+1}^{(-)} := -\hat{\mu}_{n+1} \geq 0, \tag{4.3}$$

$\|\mathbf{w}_{n+1}\|_2 = \|\mathbf{v}_{n+1}\|_2 = 1$, and $\mathbf{w}_{n+1}^T\mathbf{v}_{n+1} = 0$. Moreover, we have

$$K_{n+1} - A_{n+1} = \left[\begin{array}{c|c} K_n - A_n & \mathbf{0} \\ \hline \mathbf{0}^T & 0 \end{array}\right] + \mathbf{v}_{n+1}\delta_{n+1}^{(+)}\mathbf{v}_{n+1}^T - \mathbf{w}_{n+1}\delta_{n+1}^{(-)}\mathbf{w}_{n+1}^T.$$

Hence, it follows that

$$\|K_{n+1} - A_{n+1}\|_2 \leq \|K_n - A_n\|_2 + \max\left\{\delta_{n+1}^{(+)}, \delta_{n+1}^{(-)}\right\},$$

$$\|K_{n+1} - A_{n+1}\|_F^2 \leq \|K_n - A_n\|_F^2 + {\delta_{n+1}^{(+)}}^2 + {\delta_{n+1}^{(-)}}^2.$$

If we start with $K_{\tilde{n}} = A_{\tilde{n}} + E_{\tilde{n}}$, then by induction we obtain at step n

$$K_n - A_n = \hat{E}_{\tilde{n}} + \hat{V}\Delta^{(+)}\hat{V}^T - \hat{W}\Delta^{(-)}\hat{W}^T,$$

where $\Delta^{(+)}$ and $\Delta^{(-)}$ are diagonal matrices of order $n - \tilde{n}$, $\hat{E}_{\tilde{n}}$ is $E_{\tilde{n}}$ padded with zeros, and $\hat{V}$, $\hat{W}$ contain the successive columns $\mathbf{v}_i$ and $\mathbf{w}_i$, also padded with zeros in order to have matching dimensions. It immediately follows that

$$\|K_n - A_n\|_F^2 \leq \eta_n := \sum_{i=m+1}^{\tilde{n}} {\lambda_i^{(\tilde{n})}}^2 + \sum_{i=\tilde{n}+1}^{n} {\delta_i^{(+)}}^2 + \sum_{i=\tilde{n}+1}^{n} {\delta_i^{(-)}}^2, \tag{4.4}$$

$$\|K_n - A_n\|_2 \leq \zeta_n := \lambda_{m+1}^{(\tilde{n})} + \sum_{i=\tilde{n}+1}^{n} \max\left\{\delta_i^{(+)}, \delta_i^{(-)}\right\}. \tag{4.5}$$

Since $\hat{E}_{\tilde{n}} \succeq 0$, $\hat{V}\Delta^{(+)}\hat{V}^T \succeq 0$, and

$$K_{n+1} = \hat{A}_{n+1} + \left[\begin{array}{c|c} \hat{E}_{\tilde{n}} + \hat{V}\Delta^{(+)}\hat{V}^T - \hat{W}\Delta^{(-)}\hat{W}^T & \mathbf{0} \\ \hline \mathbf{0}^T & 0 \end{array}\right] \succeq 0,$$

one obtains the following bound for the negative eigenvalue of $\hat{A}_{n+1}$:

$$|\delta_{n+1}^{(-)}| \leq \|\hat{E}_{\tilde{n}} + \hat{V}\Delta^{(+)}\hat{V}^T\|_2.$$

So $\|\Delta^{(-)}\|$ stays of the order of $\|\Delta^{(+)}\|$ and the initial error $\|E_{\tilde{n}}\|$.

One often observes that each $|\delta_n^{(-)}| \approx \delta_n^{(+)} = \hat{\mu}_{m+1}$ and thus that all approximations A_{n+1} are optimal rank m approximations of $\hat{A}_{n+1}$, *without constraints*. Moreover, if the approximation of K_n by A_n is "good," then the nonzero eigenvalues of $K_n - A_n$ are of the order of $\lambda_{m+1}^{(n)} \leq \lambda_{m+1}$. One also often observes that both $\delta_{n+1}^{(+)}$ and $\delta_{n+1}^{(-)}$ remain bounded by λ_{m+1}, which eventually would imply

$$\sum_{i=m+1}^{N} \lambda_i^2 \leq \|K_N - A_N\|_F^2 \lessapprox (N-m)\, \lambda_{m+1}^2.$$

If the vectors in $\hat{V}$ and $\hat{W}$ are nearly orthogonal to each other, one would also have

$$\lambda_{m+1} \leq \|K_N - A_N\|_2 \lessapprox c\lambda_{m+1},$$

with $c \approx 1$. We will verify in the examples of the next section that this is nearly satisfied, but there is of course no guarantee that this last bound always holds.

Let us now try to give bounds for the downsized problems. We point out that downsizing is in fact a projection onto smaller dimensional matrices, which will produce of course smaller eigenvalues in the downsized approximation matrix, but also smaller errors since the errors matrices also get projected. Let J_n be an $n \times (n+1)$ submatrix of $\mathbb{I}_{n+1}$ with one row removed, corresponding to the deleted row in A_{n+1}. Then the downsized matrices B_n and $\tilde{K}_n$ are defined as

$$B_n := J_n A_{n+1} J_n^T, \quad \tilde{K}_n := J_n K_{n+1} J_n^T,$$

where $\tilde{K}_n$ is an $n \times n$ principal submatrix of K_N. Since the error matrices in the discussion above are also projected their the error bounds for $\|B_n - \tilde{K}_n\|$ will also decrease, but it is difficult to give a priori bounds other than the ones for $\|A_n - K_n\|$. But notice that since B_n is a submatrix of A_{n+1}, its eigenvalues β_i will interlace the eigenvalues $\mu_i^{(n+1)}$ of A_{n+1}:

$$\mu_1^{(n+1)} \geq \beta_1 \geq \cdots \geq \mu_m^{(n+1)} \geq \beta_m \geq \mu_{m+1}^{(n+1)}.$$

It is therefore important that we do the best possible approximation (by deleting the row of smallest norm) so that the β_i approximate of the m largest eigenvalues $\mu_i^{(n+1)}$ is as well as possible. At time step N, the selected submatrix $\tilde{K}_{\tilde{n}}$ should be the most "dominant" submatrix out of all $\tilde{n} \times \tilde{n}$ submatrices of K_N.

5. Numerical experiments. Some numerical experiments, showing the properties of the proposed algorithm, are reported in this section. In particular, it is shown that the numerical results agree with the empirical bound of the previous section. Moreover, in Example 5.2 it is shown that the downdating procedure with minimal norm performs better than the downdating procedure proposed in [14].

Example 5.1. The matrix considered in this example is constructed, as in [14], from the Abalone benchmark data set [2]. The underlying kernel function is the radial basis function kernel,

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|_2^2}{h^2}\right),$$

with $h = 10$. This data set has $N = 4177$ training instances of dimension $p = 7$. In Figure 5.1 the largest 100 eigenvalues of the dense SPD kernel matrix K_N are

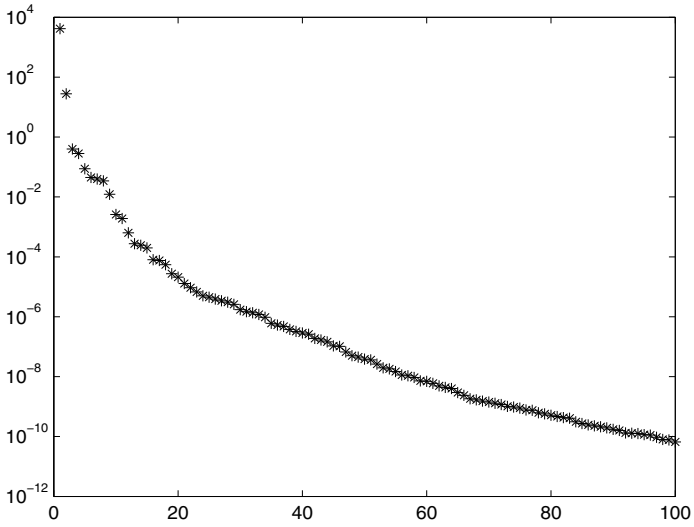

FIG. 5.1. *Distribution of the largest* 100 *eigenvalues of the Abalone matrix on a logarithmic scale.*

depicted on a logarithmic scale. We point out that there are gaps between the first 15 eigenvalues, which explains the good behavior of the updating algorithm and the bad behavior of the downsizing algorithm.

In Table 5.1 the largest nine eigenvalues of K_N, computed with the implicit restarted Lanczos algorithm (function `eigs` of MATLAB), are displayed in the second column, followed by the eigenvalues computed by the proposed algorithm, choosing $\tilde{n} = 500$, $m = 9$ (third column), and $\tilde{n} = 500$, $m = 20$ (fourth column). The correct digits are displayed in black.

TABLE 5.1

Largest nine eigenvalues of the Abalone matrix K_N (second column), of A_N obtained with updating with $\tilde{n} = 500$, $m = 9$ (third column), and with $\tilde{n} = 500$, $m = 20$ (fourth column), respectively.

i	λ_i	$\mu_i, \tilde{n} = 500, m = 9$	$\mu_i, \tilde{n} = 500, m = 20$
1	4.148381082558086e+3	4.14838108255812 7e+3	4.14838108255805 8e+3
2	2.771424671239261e+1	2.77142467123935 5e+1	2.77142467123908 1e+1
3	3.969464863546035e-1	3.96946485174339 6e-1	3.96946486354575 0e-1
4	2.828278386003848e-1	2.82827838240747 3e-1	2.82827838601794 9e-1
5	8.763549387295717e-2	8.76354893664714 5e-2	8.76354938730078 4e-2
6	4.481917665387177e-2	4.48191002296202 9e-2	4.48191766537462 7e-2
7	3.950058211492499e-2	3.95005033082028 5e-2	3.95005821145827 8e-2
8	3.449165942064433e-2	3.44915746496473 7e-2	3.44916594206963 3e-2
9	1.227519501234565e-2	1.22750932394003 8e-2	1.22751950116852 3e-2

The graphs of the sequences of $\delta_n^{(+)}$, $|\delta_n^{(-)}|$, η_n, choosing $\tilde{n} = 500$, $m = 9$, are depicted in Figure 5.2. We can observe that the behavior of the latter sequences are in agreement with the results of section 4.

In Table 5.2 we have the angles between the eigenvectors corresponding to the largest nine eigenvalues of K_N, computed by the function `eigs` of MATLAB and

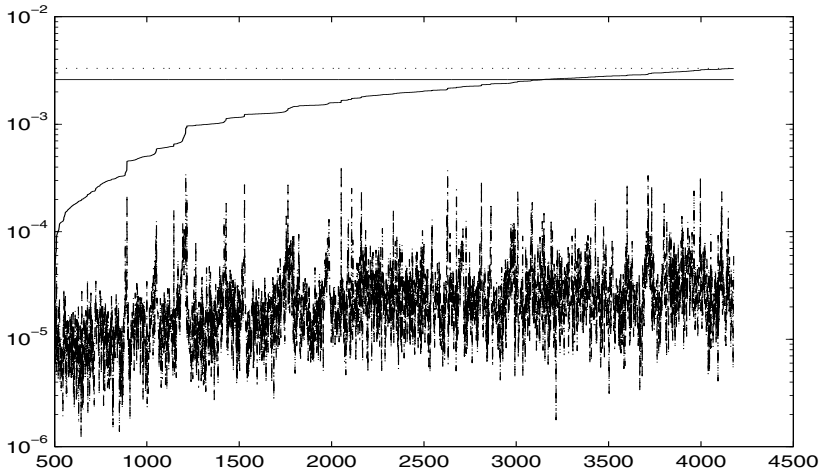

FIG. 5.2. *Plot of the sequences of* $\delta_n^{(+)}$ *(dashdot line),* $|\delta_k^{(-)}|$ *(dashed line),* η_n *(curved solid line),* λ_{m+1} *(horizontal solid line), and* $\|K_N - A_N\|_F$ *(horizontal dotted line).*

TABLE 5.2

Angles between the eigenvectors corresponding to the largest nine eigenvalues of K_N*, computed by the function* `eigs` *of* MATLAB *and those computed by the proposed algorithm for* $\tilde{n} = 500$, $m = 9$ *(second column), and* $\tilde{n} = 500$, $m = 20$ *(third column).*

i	$\angle(\mathbf{x}_i, \tilde{\mathbf{x}}_i)$	$\angle(\mathbf{x}_i, \hat{\mathbf{x}}_i)$
1	3.6500e-08	8.4294e-08
2	3.9425e-08	2.9802e-08
3	2.3774e-06	5.1619e-08
4	2.5086e-06	2.9802e-08
5	3.0084e-05	1.1151e-07
6	2.0446e-04	4.2147e-08
7	2.0213e-04	1.4901e-08
8	3.4670e-04	8.1617e-08
9	5.9886e-04	2.1073e-08

those computed by the proposed algorithm for $\tilde{n} = 500$, $m = 9$ (second column) and $\tilde{n} = 500$, $m = 20$ (third column).

In Table 5.3, we give the first nine eigenvalues of the K_N computed by the function `eigs` of MATLAB (second column), the first nine eigenvalues computed by the proposed downsizing algorithm with $\tilde{n} = 500$, $m = 9$ (third column), and the first nine eigenvalues of $K_N(I, I)$, where I is the vector of the $\tilde{n}$ indices selected by the downsizing algorithm with $\tilde{n} = 500$, $m = 9$ (fourth column), respectively. Moreover, the eigenvalues computed by the downsizing algorithm proposed in [14], with $\tilde{n} = 500$, $m = 9$, are displayed in the fifth column. Although the gaps between the eigenvalues are somehow revealed, the approximation of the eigenvalues computed by the downsizing algorithm is rather poor.

Example 5.2. In this example, we show that the proposed downdating technique, where the row to be neglected from the updated subspaces is the one with minimal norm, gives better results than the downdating procedure proposed in [14], where the first row of the updated subspace is neglected at each step.

TABLE 5.3

First nine eigenvalues of the K_N computed by the function `eigs` *of* MATLAB *(second column), by the proposed downdating algorithm with $\tilde{n} = 500$, $m = 9$ (third column), the first nine eigenvalues of $K_N(I, I)$, with $\tilde{n} = 500$, $m = 9$ (fourth column), and the first nine eigenvalues computed by the downsizing algorithm proposed in* [14]*, with $\tilde{n} = 500$, $m = 9$ (fifth column), respectively.*

i	λ_i	$\mu_i, \tilde{n} = 500$	$\lambda_i(K_N(I, I))$	$\mu_i, \tilde{n} = 500$
1	4.1484e+03	4.9263e+02	4.9263e+02	4.9662e+02
2	2.7714e+01	6.9920e+00	6.9921e+00	3.2793e+00
3	3.9695e-01	1.5378e-01	1.5385e-01	4.6656e-02
4	2.8283e-01	1.1061e-01	1.1064e-01	2.7912e-02
5	8.7635e-02	4.0973e-02	4.0997e-02	1.1534e-02
6	4.4819e-02	2.6118e-02	2.6126e-02	5.3982e-03
7	3.9501e-02	2.1848e-02	2.1858e-02	4.8140e-03
8	3.4492e-02	1.6124e-02	1.6135e-02	1.9859e-03
9	1.2275e-02	5.1022e-03	5.1035e-03	1.3614e-03

Let

$$F(i,j) = \sum_{k=1}^{3} \exp\left(-\frac{(i-\mu_k)^2 + (j-\mu_k)^2}{2\sigma_k}\right), \qquad i,j = 1,\ldots,100,$$

with

$$\mu = \begin{bmatrix} 4 & 18 & 76 \end{bmatrix}, \quad \sigma = \begin{bmatrix} 10 & 20 & 5 \end{bmatrix}.$$

Hence, F is a rank 3 matrix. Let $F = Q\Lambda Q^T$ be its spectral decomposition, and let $\tilde{\Delta} \in \mathbb{R}^{100\times 100}$ be a matrix of random numbers generated by the MATLAB function `randn`, and define $\Delta = \tilde{\Delta}/\|\tilde{\Delta}\|_2$. For this example, the considered SPD matrix is

$$K_N = F + \varepsilon\Delta\Delta^T,$$

with $\varepsilon = 1.0e-5$. The graphs of the size of the entries of K_N and of the distribution of its eigenvalues are depicted in Figure 5.3.

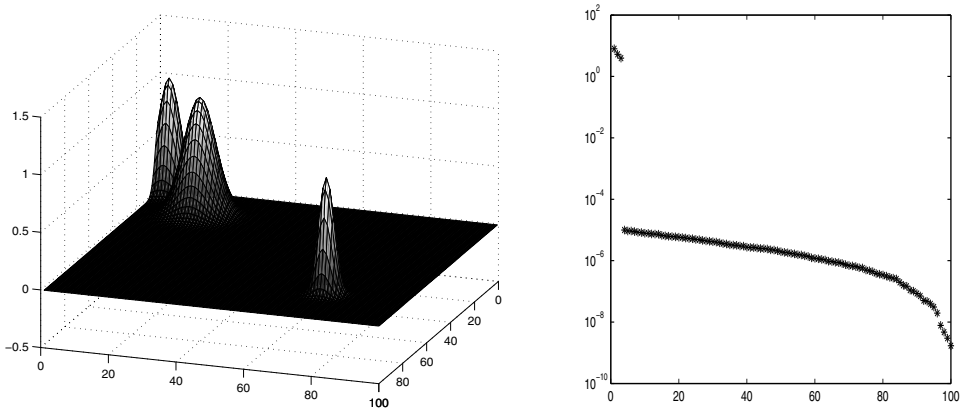

FIG. 5.3. *Graph of the size of the entries of the matrix K_N of Example* 5.2 *(left). Distribution of the eigenvalues of the matrix K_N in logarithmic scale (right).*

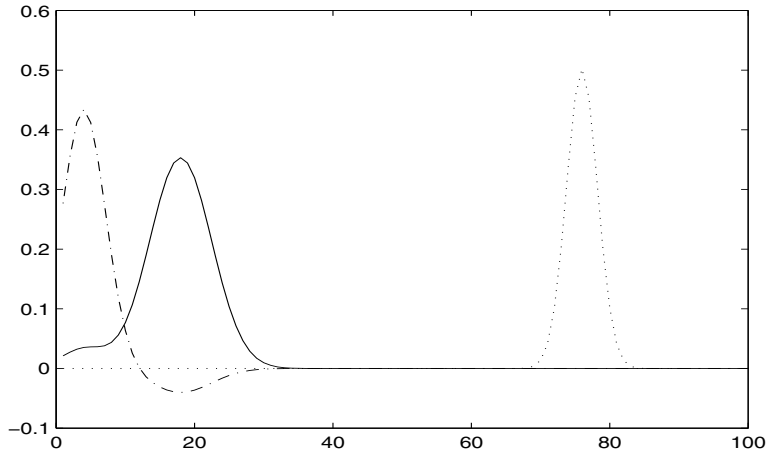

FIG. 5.4. *Plot of the eigenvectors of K_N corresponding to the three largest eigenvalues of K_N.*

The matrix K_N has three dominant eigenvalues, and their corresponding eigenvectors are depicted in Figure 5.4.

In Table 5.4 are reported the first three eigenvalues of the matrix K_N (second column), the eigenvalues obtained using the downdating procedure with minimum norm tracking with $\tilde{n} = 30, 40, 50$, $m = 3$ (third, fourth, and fifth column in the table, respectively), and the eigenvalues computed by the downsizing procedure, removing the first row of the updated matrix at each step as described in [14] (sixth column), with $m = 3$, $\tilde{n} = 50$. We observe that the latter procedure tracks only the third nonzero eigenvalue; i.e., it takes into account only the last $\tilde{n}$ rows and columns of K_N. In fact, the eigenvalues in column 7 are the three largest eigenvalues of the principal submatrix $K_N(51 : 100, 51 : 100)$. Furthermore, we observe that, choosing a smaller size of $\tilde{n}$, the updating algorithm with the removal of the first row breaks downs because the matrix A_n tends to lose the positive semidefiniteness property.

TABLE 5.4

Largest three eigenvalues of the matrix K_N of Example 5.2 *(second column). Largest three eigenvalues computed with downdating procedure with minimal norm, with $m = 3$ and $\tilde{n} = 30, 40, 50$ (third, fourth, and fifth column), respectively. Three largest eigenvalues of K_N computed with downdating procedure described in* [14] *(sixth column).*

i	λ_i	$\mu_i, \tilde{n} = 30$	$\mu_i, \tilde{n} = 40$	$\mu_i, \tilde{n} = 50$	$\mu_i, \tilde{n} = 50$
1	7.949478e+0	7.375113e+0	7.820407e+0	7.947127e+0	3.963329e+0
2	5.261405e+0	5.255163e+0	5.260243e+0	5.261384e+0	5.417202e-6
3	3.963329e+0	3.948244e+0	3.963213e+0	3.963329e+0	4.824060e-6

Moreover, the first three eigenvalues of the submatrix extracted from K_N, keeping the rows and columns with the same indexes tracked by the downsizing procedure with minimal norm for $m = 3$, $\tilde{n} = 30, 40, 50$, have the same significant digits of those displayed in columns 3, 4, 5 of Table 5.4, respectively. Therefore, this procedure can be seen as a way to extract the important features of an SPD matrix.

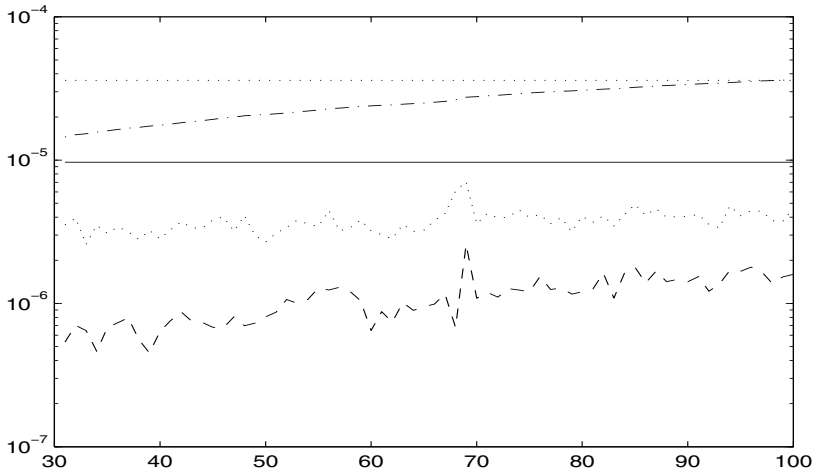

FIG. 5.5. *Plot of the sequences of* $\delta_n^{(+)}$ *(upper dotted line),* $|\delta_k^{(-)}|$ *(dashed line),* η_n *(dashdot line),* λ_{m+1} *(solid line), and* $\|K_N - A_N\|_F$ *(lower dotted line).*

The graphs of the sequences of $\delta_n^{(+)}$, $|\delta_n^{(-)}|$, η_n are depicted in Figure 5.5. We can observe that the behavior of these sequences is in agreement with the results of section 4.

Example 5.3. In this example, we consider the following matrix of order $N = 100$,

$$K_N = \begin{bmatrix} \tilde{K}_1 & \\ & \tilde{K}_2 \end{bmatrix}$$

with

$$\tilde{K}_1 = \begin{bmatrix} 1.4 & & & & & \\ & 1.3 & & & & \\ & & 1.2 & & & \\ & & & 1.1 & & \\ & & & & 1.0 & \end{bmatrix}, \quad \tilde{K}_2 = \alpha \mathbb{I}_k + \beta \mathbb{1}_k \mathbb{1}_k^T,$$

and $\alpha = 0.05$, $\beta = 0.02$, $k = 95$. Moreover, $\mathbb{1}_k$ is the vector of length k with all the entries equal to 1. The eigenvalues of $\tilde{K}_2$ are $\alpha + k\beta = 1.95$ and $\alpha = 0.05$ of multiplicity $k - 1 = 94$. Therefore, the eigenvalues of A are $[1.95, 1.4, 1.3, 1.2, 1.1, 1.0, 0.05, \ldots, 0.05]^T$.

The proposed updating algorithm is used to track the subspace associated to the largest $m = 5$ eigenvalues considering, as the initial step, the submatrix of K_N of order 5. The results are reported in Table 5.5.

We observe that the proposed algorithm fails to detect the eigenvalue 1.95 and the associated subspace since, at the ith step of the algorithm, $i = 6, \ldots, 100$, the

TABLE 5.5

Largest five eigenvalues of the matrix K_N of Example 5.3 *(second column). Largest five eigenvalues computed with the proposed updating procedure, with $m = 5$ and $\tilde{n} = 5$ (third column).*

i	λ_i	$\mu_i, \tilde{n} = 5$
1	1.950000000000000	1.400000000000000
2	1.400000000000000	1.300000000000000
3	1.300000000000000	1.200000000000000
4	1.200000000000000	1.100000000000000
5	1.100000000000000	1.000000000000000

processed matrix is

$$K^{(i)} = \left[\begin{array}{ccccc|cc} 1.4 & & & & & & \\ & 1.3 & & & & & \\ & & 1.2 & & & & \\ & & & 1.1 & & & \\ & & & & 1.0 & & \\ \hline & & & & & 0 & \beta\sqrt{i-5} \\ & & & & & \beta\sqrt{i-5} & \alpha+\beta \end{array}\right].$$

Hence the eigenvalues are $[1.4, 1.3, 1.2, 1.1, 1.0]$ and the eigenvalues of the 2×2 matrix

$$\left[\begin{array}{cc} 0 & \beta\sqrt{i-5} \\ \beta\sqrt{i-5} & \alpha+\beta \end{array}\right],$$

i.e., $\sigma_{m+1}^{(+)} = \frac{\alpha+\beta}{2} + \frac{\sqrt{(\alpha+\beta)^2+\beta^2(i-5)}}{2}$ and $\sigma_{m+1}^{(-)} = \frac{\alpha+\beta}{2} - \frac{\sqrt{(\alpha+\beta)^2+\beta^2(i-5)}}{2}$. Although $\sigma_{m+1}^{(+)} < |\sigma_{n+1}^{(-)}|$, for $i = m+1, \ldots, n$, it turns out that also $\sigma_{m+1}^{(+)} < 1.0$; that is, $\sigma_{m+1}^{(+)}$ is always smaller than the smallest eigenvalue of $\tilde{K}_1$. This means that the updating procedure proposed in this paper will fail to track the subspace associated to 1.95, the largest eigenvalue of $\tilde{K}_2$.

The graphs of the sequences of $\delta_n^{(+)}$, $|\delta_n^{(-)}|$, η_n are depicted in Figure 5.6.

Nevertheless, we observe that the theoretical bounds derived in section 4 still hold, since $\lambda_{m+1} = 1$, $\sqrt{\sum_{m+1}^{N} \lambda_i^2} = 1.1113$, $\|K_N - A_N\|_F = 2.0093$, $\sqrt{N-m}\lambda_{m+1} = 9.7467$.

TABLE 5.6

Largest five eigenvalues of the matrix PK_NP^T of Example 5.3 *computed by* `eig` *of* MATLAB *(second column). Largest five eigenvalues computed with the proposed updating procedure, with $m = 5$ and $\tilde{n} = 5$ (third column).*

i	λ_i	$\mu_i, \tilde{n} = 5$
1	1.949999999999999	1.949999999999999
2	1.400000000000000	1.400000000000000
3	1.300000000000000	1.299999999999998
4	1.199999999999999	1.199999999999997
5	1.100000000000000	1.099999999999998

However, to retrieve the subspace associated to the largest five eigenvalues, it is sufficient to apply the proposed procedure to the matrix PK_NP^T, with P a random permutation of order N. The results, in this case, are shown in Table 5.6 and depicted in Figure 5.7.

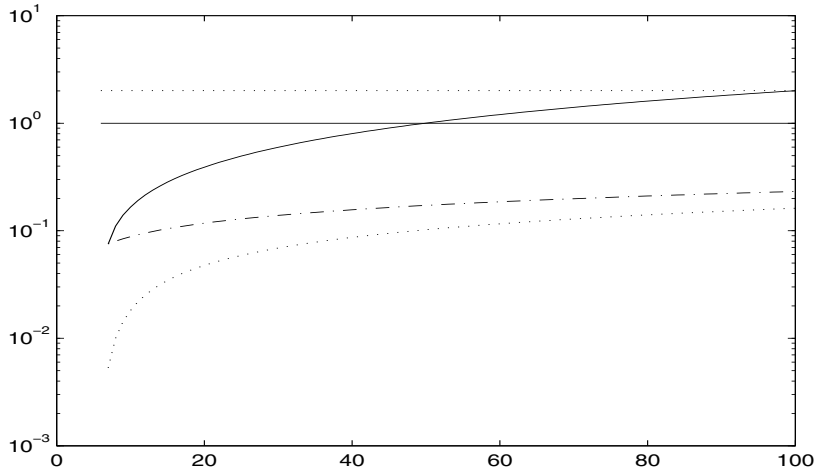

FIG. 5.6. *Plot of the sequences of* $\delta_n^{(+)}$ *(dotted line),* $|\delta_k^{(-)}|$ *(dashed line),* η_n *(dashdot line),* λ_{m+1} *(solid line), and* $\|K_N - A_N\|_F$ *(horizontal dotted line).*

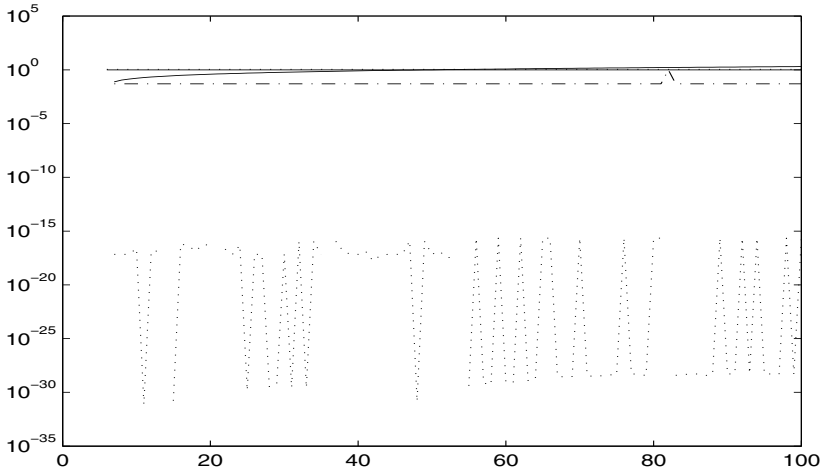

FIG. 5.7. *Plot of the sequences of* $\delta_n^{(+)}$ *(dotted line),* $|\delta_k^{(-)}|$ *(dashed line),* η_n *(dashdot line),* λ_{m+1} *(solid line), and* $\|K_N - A_N\|_F$ *(horizontal dotted line).*

6. Conclusions. In this paper, we presented fast algorithms to compute incrementally the dominant eigenspace of a positive definite matrix. This is applied to the specific problem of kernel based principal components. Our methods are strongly inspired by earlier work in this area [14, 9], but in this paper we improve significantly the complexity of these earlier papers and we do a detailed analysis of the accuracy that can be obtained with these methods. The overall complexity of the incremental updating technique to compute an $N \times m$ basis matrix U_N for the dominant eigenspace of K_N is reduced in this paper from $(m+4)N^2m + O(Nm^3)$ to $6N^2m + O(Nm^2)$.

When using both incremental updating and downsizing to compute an $\tilde{n} \times m$ basis matrix $\hat{U}_{\tilde{n}}$ of the dominant eigenspace of $\hat{K}_{\tilde{n}}$ (an $\tilde{n} \times \tilde{n}$ principal submatrix of K_N), the complexity is reduced in this paper from $(12m + 4)N\tilde{n}m + O(Nm^3)$ to $16N\tilde{n}m + O(Nm^2)$. This is in both cases essentially a reduction by a factor m.

Moreover, we proposed here a new downsizing technique that allows us to better keep track of the dominant components in the matrix K_N. Earlier methods were dismissing the first row/column of the process data (i.e., the "oldest" data point) whereas we propose to dismiss the row/column of least energy (i.e., the "weakest" data point). The previous approach had the disadvantage of focusing on the last $\tilde{n} \times \tilde{n}$ submatrix of K_N, which may not contain all the dominant features of that matrix. Our new method tries to keep the rows and columns with the most dominant features of the matrix and is in that sense almost like a dominant subset selection technique. If the columns and rows of the matrix K_N can be associated with time, one could say that downsizing the oldest data point keeps track of the most recent information, but this could be achieved as well by a simple exponential weighting of columns and rows of K_N, and this is easy to incorporate in our updating scheme as well.

Acknowledgment. We would like to thank the anonymous referees for their helpful comments and suggestions which helped to improve the presentation of the paper.

REFERENCES

[1] R. Badeau, G. Richard, and B. David, *Sliding window adaptive SVD algorithms*, IEEE Trans. Signal Process., 52 (2004), pp. 1–10.

[2] C. Blake and C. Merz, *UCI Repository of Machine Learning Databases*, available online at http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

[3] M. Brand, *Fast low-rank modifications of the thin singular value decomposition*, Linear Algebra Appl., 415 (2006), pp. 20–30.

[4] J. R. Bunch and C. P. Nielsen, *Updating the singular value decomposition*, Numer. Math., 31 (1978), pp. 111–129.

[5] P. Businger, *Updating a singular value decomposition*, BIT, 10 (1970), pp. 376–385.

[6] Y. Chahlaoui, K. Gallivan, and P. Van Dooren, *Recursive calculation of dominant singular subspaces*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 445–463.

[7] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang, *An eigenspace update algorithm for image analysis*, Graph. Model Image Process., 59 (1997), pp. 321–332.

[8] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart, *Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization*, Math. Comp., 30 (1976), pp. 772–795.

[9] G. Gins, I. Y. Smets, and J. F. Van Impe, *Efficient tracking of the dominant eigenspace of a normalized kernel matrix*, Neural. Comput., 20 (2008), pp. 523–554.

[10] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[11] A. M. Jade, B. Srikanth, V. Jayaraman, B. Kulkarni, J. Jog, and L. Priya, *Feature extraction and denoising using kernel PCA*, Chem. Eng. Sci., 58 (2003), pp. 4441–4448.

[12] Hammarling, *A note on modifications to the Givens plane rotation*, IMA J. Appl. Math., 13 (1974), pp. 215–218.

[13] N. Higham, *Computing a nearest symmetric positive semidefinite matrix*, Linear Algebra Appl., 103 (1988), pp. 103–118.

[14] L. Hoegaerts, L. De Lathauwer, I. Goethals, J. A. K. Suykens, J. Vandewalle, and B. De Moor, *Efficiently updating and tracking the dominant kernel principal components*, Neural Networks, 20 (2007), pp. 220–229.

[15] A. Levy and M. Lindenbaum, *Sequential Karhunen–Loeve basis extraction and its applications to images*, IEEE Trans. Image Process., 9 (2000), pp. 1371–1374.

[16] N. Mastronardi, M. Van Barel, and R. Vandebril, *A note on the recursive calculation of dominant singular subspaces*, Numer. Algebra, 38 (2005), pp. 237–242.

[17] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, *Kernel PCA and de-noising in feature spaces*, in Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems, Vol. 11, M. S. Kearns, S. A. Solla, and D. A. Cohn, eds., MIT Press, Cambridge, MA, 1999, pp. 536–542.
[18] M. Moonen, P. Van Dooren, and J. Vandewalle, *A singular value decomposition updating algorithm for subspace tracking*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1015–1038.
[19] R. Rosipal and M. Girolami, *An expectation-maximization approach to nonlinear component analysis*, Neural Comput., 13 (2001), pp. 505–510.
[20] R. Rosipal, M. Girolami, L. J. Trejo, and A. Cichocki, *Kernel PCA for feature extraction and de-noising in nonlinear regression*, Neural Comput. Appl., 10 (2001), pp. 231–243.
[21] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola, *Input space versus feature space in kernel-based methods*, IEEE Trans. Neural Networks, 10 (1999), pp. 1000–1017.
[22] B. Schölkopf, S. Mika, A. J. Smola, G. Rätsch, and K. R. Müller, *Kernel PCA pattern reconstruction via approximate pre-images*, in Proceedings of the 8th International Conference on Artificial Neural Networks, L. Niklasson, M. Bodén, and T. Ziemke, eds., Perspectives in Neural Computing, Springer-Verlag, Berlin, 1998, pp. 147–152.
[23] B. Schölkopf and A. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, 2002.
[24] B. Schölkopf, A. Smola, and K. R. Müller, *Nonlinear component analysis as a kernel eigenvalue problem*, Neural Comput., 10 (1998), pp. 1299–1319.
[25] G. W. Stewart, *Matrix Algorithms,* II: *Eigensystems,* SIAM, Philadelphia, 2001.
[26] G. W. Stewart, *An updating algorithm for subspace tracking*, IEEE Trans. Image Process., 40 (1992), pp. 1535–1541.
[27] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, B., and J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific, Singapore, 2002.
[28] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.
[29] H. Zha and H. D. Simon, *On updating problems in latent semantic indexing*, SIAM J. Sci. Comput., 21 (1999), pp. 782–791.