Accelerating data uncertainty quantification by solving linear systems with multiple right-hand sides

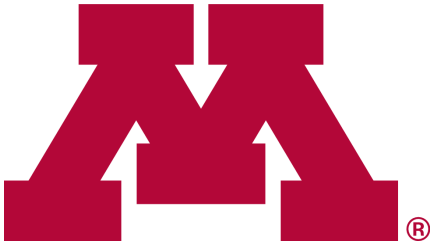
Vassilis Kalantzis, Costas Bekas, Alessandro Curioni, and Efstratios Gallopoulos

January 2013

EPrint ID: 2013.1

Department of Computer Science and Engineering
University of Minnesota

Preprints available from: `http://www-users.cs.umn.edu/kalantzi`



UNIVERSITY OF MINNESOTA

Supercomputing Institute

ACCELERATING DATA UNCERTAINTY QUANTIFICATION BY SOLVING LINEAR SYSTEMS WITH MULTIPLE RIGHT-HAND SIDES

V. KALANTZIS*, C. BEKAS†, A. CURIONI‡, AND E. GALLOPOULOS‡

Abstract. The subject of this work is accelerating data uncertainty quantification. In particular, we are interested in expediting the stochastic estimation of the diagonal of the inverse covariance (precision) matrix that holds a wealth of information concerning the quality of data collections, especially when the matrices are symmetric positive definite and dense. Schemes built on direct methods incur a prohibitive cubic cost. Recently proposed iterative methods can remedy this but the overall cost is raised again as the convergence of stochastic estimators can be slow. The motivation behind our approach stems from the fact that the computational bottleneck in stochastic estimation is the application of the precision matrix on a set of appropriately selected vectors. The proposed method combines block conjugate gradient with a block-seed approach for multiple right-hand sides, taking advantage of the nature of the right-hand sides and the fact that the diagonal is not sought to high accuracy. Our method is applicable if the matrix is only known implicitly and also produces a matrix-free diagonal preconditioner that can be applied to further accelerate the method. Numerical experiments confirm that the approach is promising and helps contain the overall cost of diagonal estimation as the number of samples grows.

1. Introduction. It is a common realization that the 21st century marks a new era in science, engineering, business as well as everyday life: the era of data. It is also understood that before we can get any useful knowledge out of the "tsunami of data", we need to understand its quality and the degree that it can be trusted. Therefore, there is a need for techniques that measure the uncertainty in data collections. Much information can be derived from the elements of the inverse of the sample covariance matrix for the data collection, especially those lying on the diagonal. We build upon previous work and show that the combination of stochastic estimation and new solvers for linear systems with multiple right-hand sides (mrhs) can drastically reduce the prohibitive cubic cost (with respect to the size of the data collection) of conventional methods for diagonal estimation for the general case where the underlying matrix is dense, symmetric positive definite (spd) without any additional structure. We note from the start that in this paper we do not address the important problem of data assimilation.

Adopting the basic premise of the vector space model in data analysis, in which complex, multivariate data are represented as long tuples of numbers, let $\mathcal{D}^{n\times q}$ be the data matrix where columns correspond to samples and rows to features modeled as random variables following certain distributions. The sample covariance matrix is

$$A = \frac{1}{q}\mathcal{D}\mathcal{D}^{\top} \tag{1.1}$$

and collectively holds relations between features. The matrix is spd possibly after preprocessing (see e.g. [13, 24]). In this context, element (i, j) of the inverse A^{-1}, also known as the precision matrix, can be interpreted as the partial variance of feature i with respect to feature j. Thus, its diagonal essentially holds the degree of confidence one can have in the data collection [21, 28, 35, 37]. If only few elements of the diagonal or if only the trace of the precision matrix are sought, an efficient

*Computer Eng. & Informatics Dept., University of Patras, Greece. {kalantzis@ceid.upatras.gr}
†IBM Research - Zurich, Switzerland. {bek@zurich.ibm.com,curioni@zurich.ibm.com}
‡Computer Eng. & Informatics Dept., University of Patras, Greece. {stratis@ceid.upatras.gr}

approach is to use methods in the spirit of the work started by Bai, Fahey and Golub [5] and then in [6, 26] and more recently by Brezinski et al. [12] that utilize the natural correspondence between the Lanczos algorithm and Gaussian quadrature via matrix moments. See also the seminal work of Golub and Meurant [18] for extensive discussions on these methods. Such methods can lead to estimates for the trace using Monte Carlo techniques and provide bounds for individual diagonal elements at quadratic cost for dense matrices.

In this paper, instead, we are concerned with estimating the entire diagonal of the precision matrix. Calculating it by means of standard numerical linear algebra techniques can be accomplished by first computing the Cholesky factorization $A = R^\top R$, then solving and retrieving the diagonal elements $(R^\top R)y_i = e_i, d_i = e_i^\top y_i$ for $i = 1, \ldots, n$, where e_i is the i-th column of the identity matrix. Covariance matrices are in general dense, so the overall cost of this procedure as well as the aforementioned quadrature-based methods scale as $O(n^3)$. As n can range from thousands to several millions, it is crucial to develop algorithms that demonstrate high performance and run efficiently on the types of computational platforms available to analysts, ranging from new commodity-level multicore systems for small scale problems to massively parallel supercomputers and beyond for large collections.

In previous work ([7, 8]) some of the present authors adopted a diagonal stochastic estimator introduced in [9] and showed that it is feasible to reduce the cost of estimating the diagonal of the inverse for dense matrices from cubic to quadratic levels by combining stochastic estimation and an efficient linear solver based on conjugate gradient (CG) with iterative refinement. The solver is also applicable when the matrix is implicitly known via matrix-vector multiplications (hereafter referred to as MVs). Consider a matrix function $\mathcal{P}(A)$ and a procedure $P(A, z)$ that implements (or approximates) $P(A, z) \approx \mathcal{P}(A)z$. Then, the following formula is an estimator for the diagonal of the matrix function $\mathcal{P}(A)$:

$$\mathcal{D}_s(\mathcal{P}(A)) := [\sum_{k=1}^{s}(z^{(k)} \odot P(A, z^{(k)}))] \oslash [\sum_{k=1}^{s} z^{(k)} \odot z^{(k)}], \tag{1.2}$$

for carefully selected vectors $z^{(k)}$. The symbols $\odot, \oslash$ denote Hadamard (element-wise) multiplication and division respectively (see [9]). In the present case, the aforementioned function is the matrix inverse, $\mathcal{P}(A) = A^{-1}$, and thus $P(A, z)$ can be any method that approximates $A^{-1}z$. Hence, for a limited number, say s, of sampling operations combined with a low (quadratic) cost solver $P(A, z)$, the overall cost becomes $O(sn^2)$. It is well known, however, that stochastic estimation converges slowly when the estimated quantity exhibits much variance and the number of samples, s, may need to be very large (see e.g. [4]). Once this becomes a significant fraction of n, the overall cost is elevated again to cubic levels. We demonstrate here that it is possible to perform the full number of required sampling steps (s) but only pay a fraction of the corresponding cost. The simple but crucial observation which motivates our proposal is that the diagonal estimator (1.2) requires the solution of a linear system with multiple (s) right-hand sides. Therefore, to this effect we develop a special matrix-free linear solver for systems with spd matrices and mrhs.

We note in passing that other approaches to reduce cost can be envisioned, such as taking advantage of (sparse) structure, specially designed vectors $z^{(k)}$; cf. [9, 25, 36]. However, in this work we are interested in covariance matrices that do not exhibit a matrix structure appropriate for these approaches.

2. The case for multiple right-hand sides in stochastic estimation . Formula (1.2) for the estimator suggests that if one assumes that the cost of applying $P(A, z^{(k)})$ is independent of the vector $z^{(k)}$, the complexity of the computation depends linearly on the number of samples, e.g. it is s times the cost of applying $P(A, z^{(k)})$. Indeed, a natural first step in attempting to accelerate the estimator is to implement the fastest method possible to compute $P(A, z^{(k)})$; see e.g. [7, 8]. On the other hand, the linear dependence on s is not quite true. To see this, first note that for the diagonal of the inverse, formula (1.2) requires the evaluation of all vectors $\{A^{-1}z^{(k)}\}_{k=1:s}$. If direct methods were applicable, then the cost of solving for s right-hand sides would require only one, rather than s Cholesky factorizations, thus the dominant cubic cost does not scale linearly with s. Direct methods are not applicable, however, first because we target very large problems and second because we would like the method to be applicable if the matrix is only available implicitly, via MVs. We thus explore what means are available in the context of matrix-free iterative methods to speed-up the stochastic diagonal estimator. We consider projection-type iterative methods that promise to solve $AX = Z$ for $X \in \mathbb{R}^{n\times s}$ at an average cost that is smaller than the average cost of solving each of the s linear systems independently. Two important issues are how many and what right-hand sides to use. We already mentioned earlier that s can be large. Regarding the second issue, based on the fact that no special structure is assumed for A and its inverse, the columns $z^{(k)}$ of Z are selected to be Rademacher vectors, that is random vectors with independent entries that take the values ± 1 with equal probability. Our choices regarding these issues are consistent with previous studies in stochastic estimation; cf. [4, 7, 22].

The main question, therefore, is how to design an efficient iterative method under these conditions. Seed methods inspired by early work in computational electromagnetics [33] as well as earlier ideas of Lanczos (cf. [32]), Parlett [30] and Saad [31], and block methods, such as block CG (BCG) introduced by O'Leary [27] are promising candidates. Many methods are available and analyzed in the literature for symmetric and nonsymmetric systems, see e.g. [11, 16, 20, 29, 34] for some recent contributions. Some of these methods, e.g. [29], are designed to handle sequences of nonsymmetric systems and mrhs. A notable characteristic of the present application is that the right-hand sides are readily available on demand, the matrix is spd and does not change, therefore some of the techniques developed in these papers are not necessary.

Recall that in seed methods, one or more right-hand sides are selected first and the corresponding system(s) (called "seed system(s)" or "seed" for short) are solved using a (possibly block) Krylov method. In the course of the solution of the seed, the remaining systems are projected and approximated on the generated Krylov subspace. After a sufficient number of iterations for the seed system(s), the effective condition number of the matrix for the remaining systems is smaller; cf. [31]. It is thus suggested to select another seed and repeat the same process until all systems have been adequately approximated. This "multi-seeding" approach was used in [33], pursued for nonsymmetric systems in [32], and then studied in detail for the spd case in [14]. Comparing the performance of methods analyzed in [14, p. 1718], it was observed that "the single seed method depends heavily on the closeness of the right-hand sides while the block method depends less" e.g. if the right-hand sides are statistically independent or orthogonal. This finding supports the use of the block rather than the single seed approach when estimating the diagonal of the inverse with formula (1.2) with Rademacher vectors as right-hand sides as is the case here. Our starting point, therefore, is a hybrid form of the seed and block approaches for spd matrices, which

we call "block-seed CG", first proposed by Chan and Wan ([14]) and extended later by Kilmer et al. to nonsymmetric systems [23]. Further extending the seed approach, several methods in the literature use information obtained in the early seeds to deflate spectral information in order to improve the effective condition of subsequent systems; see the work of Giraud et al. [17] for a systematic evaluation of several methods based on this approach as well as [1, 10, 19, 34].

The idea of deploying iterative methods that take advantage of the mrhs has also been used by Anitescu et al. in the context of data analysis [3]. They proposed using BCG for a maximum likelihood estimation problem that required the trace of a matrix function depending on the inverse of a covariance matrix; Chen in [15] further refined the method by combining it with deflation. The literature, however, is still sparse in iterative methods for estimating the diagonal of the inverse; to the best of our knowledge this is the first time iterative mrhs solvers are used to this effect.

3. A block-seed CG approach. In studies targeting the simple (non-block) seed approach Abdoul-Rehim et al. noted that it is more effective to seed only once and then apply an iterative method individually on each of the remaining systems using as starting vector the approximation obtained from the first step; e.g. see [2]. We cast the choices of [2] in the context of the proposed block-seed approach. Specifically, we construct a Krylov subspace that is larger and as such enables improved approximation of more eigenvectors corresponding to a wider range of extremal eigenvalues. To achieve this, we choose a relatively small tolerance for stopping criterion. This prevents the method from stopping because of an early convergence of the linear systems before these eigenvectors have been approximated.

Thus, the first step is to use block-seed CG but to seed only once, so that at the end of the seed phase a larger and richer (block) Krylov subspace is created that also approximates well the solutions of a block of seed systems. We call this algorithm INIT-BCG and list it as Algorithm 1. INIT-BCG will serve as the main module of the method presented in this paper. Note that it coincides with the special case $k = 0$ of the block-seed method in [14, p. 1712] and is a block generalization of methods found in [17]. Following INIT-BCG, one could apply BCG to solve the remaining systems. A more effective approach is proposed in Section 3.1.

INIT-BCG will not breakdown as long as blocks $P_i, R_i^{(1)}$ retain full rank. Each factor ζ_i in the iteration is the inverse of the upper-triangular factor of the thin QR decomposition of $R_i^{(1)} + P_{i-1}\beta$ and is also used to detect loss of rank. Inversion of small (order-p) blocks is done explicitly using the singular value decomposition. These inversions and multiplications between blocks to compute the terms α and β are performed using efficient codes for dense linear algebra. Recall that the stability of block methods is delicate because of the possible linear dependence among residual vectors for different right-hand sides (cf. [27] as well as [20] and references therein). Linear dependence could also be due to variability in the convergence of systems within a block. On the other hand, our experience here as well as reports from [15] suggest that when solving systems with Rademacher vectors as right-hand sides, BCG is less prone to loss of rank. It can be shown that the following relation holds:

$$R_{m+1}^{(j)} = \prod_{i=0}^{m} (I - AP_i(P_i^\top AP_i)^{-1}P_i^\top) R_0^{(j)}. \tag{3.1}$$

Matrix $I - AP_i(P_i^\top AP_i)^{-1}P_i^\top$ is a projector that removes from $R_0^{(j)}$ components along

Algorithm 1 The INIT-BCG algorithm for $AX = Z$ where $Z \in \mathbb{R}^{n\times s}$.

input : $A, Z = [z^{(1)}, \ldots, z^{(s)}], X_0, \text{tol}, \nu$
output : $[X_{m+1}^{(1)}, X_{m+1}^{(2)}, \ldots, X_{m+1}^{(\nu)}]$ {for convenience assume that $s = p\nu$}
$R_0 = Z - AX_0$, partition $R_0 := [R_0^{(1)}, \ldots, R_0^{(\nu)}]$, where $R_0^{(j)}$ is $n \times p$ and $s = p\nu$
compute $P_0 = R_0^{(1)}\zeta_0$ and $R_0^{(1)\top} R_0^{(1)}$ and set $i = 0$
repeat
  {BCG phase}
  $\alpha = (P_i^\top A P_i)^{-1}\zeta_i^\top (R_i^{(1)\top} R_i^{(1)})$
  $X_{i+1}^{(1)} = X_{i+1}^{(1)} + P_i\alpha$
  $R_{i+1}^{(1)} = R_i^{(1)} - AP_i\alpha$
  $\beta = \zeta_i^{-1}(R_i^{(1)\top} R_i^{(1)})^{-1}(R_{i+1}^{(1)\top} R_{i+1}^{(1)})$
  $P_{i+1} = (R_{i+1}^{(1)} + P_i\beta)\zeta_{i+1}$
  for $j = 2, \ldots, \nu$ **do** {projection phase}
    $\eta_i^{(j)} = (P_i^\top A P_i)^{-1}(P_i^\top R_i^{(j)})$
    $X_{i+1}^{(j)} = X_i^{(j)} + P_i\eta_i^{(j)}$
    $R_{i+1}^{(j)} = R_i^{(j)} - AP_i\eta_i^{(j)}$
  end for
  $i = i + 1$
until $\|R_{i+1}^{(1)}\|_F \le \text{tol}$

the column space of P_i. In the sequel we also make use of the orthogonal projector

$$\Pi_i = I - P_i P_i^\top. \tag{3.2}$$

Denoting by

$$\text{bspan}\{R_0, AR_0, \ldots, A^{m-1}R_0\} := \{\sum_{i=0}^{m-1} A^i R_0 \gamma_i; \gamma_i \in \mathbb{R}^{p\times p}\}$$

the block span of the elements defining the block Krylov subspace, in exact arithmetic it holds that

$$\text{bspan}\{P_0, \ldots, P_{m-1}\} = \text{bspan}\{R_0, AR_0, \ldots, A^{m-1}R_0\}. \tag{3.3}$$

3.1. The MOD-INIT-BCG approach . Algorithm INIT-BCG applies block seeding once, possibly for more iterations than the minimum needed for obtaining the desired accuracy from the solution of the current block. We next propose an additional iteration layer with INIT-BCG, motivated by our evaluation of the effects of finite precision in the seeding that leads to a more effective algorithm for the problem under study. Specifically, assume that the seed block has already performed m iterations and that relation (3.1) holds. If one were to continue past iteration m it would hold that

$$R_{m+1}^{(j)} \in R_m^{(j)} + A\langle P_m\rangle, \quad R_{m+1}^{(j)} \perp \langle P_m\rangle,$$

where $\langle P_m\rangle$ denotes the column space of P_m. In exact arithmetic, blocks $P_0, \ldots, P_m$ are mutually A-orthogonal, hence block $R_{m+1}^{(j)}$ is orthogonal to $\text{bspan}\{P_0, \ldots, P_{m-1}\}$

(cf. Eq. (3.3)). Thus, $R_{m+1}^{(j)}$ is nearly orthogonal to any eigenvector that is well approximated in $\langle P_0, \ldots, P_m \rangle$. In finite precision, however, assuming that the block-seed is in iteration m, P_m is not exactly A-orthogonal to $P_0, \ldots, P_{m-1}$. Because of the loss of A-orthogonality, previously deflated components from more distant blocks $P_0, P_1, \ldots$ might become active again in non-seed residual blocks and components from eigenvectors associated with extremal eigenvalues could reappear as a consequence of the fact that extremal eigenvectors are the first to be approximated.

One remedy is reorthogonalization but this requires additional storage to hold a basis for the Krylov subspace and $O(nm^2p^2)$ computational cost that is prohibitive if m, p are large. Instead, we choose to deflate again, from non-seed residuals, direction blocks $P_0, \ldots, P_m$ in order to suppress components that re-emerge as described above. It appears that this would require preserving the blocks $P_0, AP_0, \ldots$ in memory in order to repeat the Galerkin projection as in INIT-BCG. These blocks are not needed during INIT-BCG and therefore could be stored in secondary storage. Either way there would be an overhead, either on memory or due to slow reads/writes necessary to access secondary storage. We propose to trade these undesirable storage related penalties for extra MVs. Specifically, we call INIT-BCG again in order to generate and then deflate again these direction blocks from the non-seeds. We name the method MOD-INIT-BCG and list it, together with the follow-up steps needed in order to accomplish the statistical estimation of the diagonal of the inverse, as Algorithm 2.

Algorithm 2 Estimating $\mathrm{diag}(A^{-1})$ with MOD-INIT-BCG.

input: $A, Z = [z^{(1)}, \ldots, z^{(s)}], \nu, \mathrm{tol}_1, \mathrm{tol}_2, \mathrm{tol}$
output: $\mathcal{D}_s(A^{-1})$
{MOD-INIT-BCG phase}
partition $Z = [Z^{(1)}, \ldots, Z^{(\nu)}]$ {for convenience assume that $s = p\nu$}
$[X^{(1)}, \hat{X}_0^{(2)} \ldots, \hat{X}_0^{(\nu)}] = \text{INIT-BCG}(A, Z, 0_{n\times s}, \mathrm{tol}_1, \nu)$
$\hat{X}_0 = [0_{n\times p}, \hat{X}_0^{(2)} \ldots, \hat{X}_0^{(\nu)}]$
$[\tilde{X}^{(1)}, X_0^{(2)} \ldots, X_0^{(\nu)}] = \text{INIT-BCG}(A, Z, \hat{X}_0, \mathrm{tol}_2, \nu)$
{use BCG to solve remaining systems}
for $k = 2, \ldots, \nu$ **do**
  $[\tilde{X}^{(k)}] = \text{BCG}(A, Z^{(k)}, X_0^{(k)}, \mathrm{tol})$
end for
{stochastic estimation phase}
$t = \sum_{j=1}^{s} \tilde{x}^{(j)} \odot z^{(j)}$ {Vectors $\tilde{x}^{(j)}$ are columns of $[\tilde{X}^{(1)}, \tilde{X}^{(2)}, \ldots, \tilde{X}^{(\nu)}]$}
$q = \sum_{j=1}^{s} z^{(j)} \odot z^{(j)}$
$\mathcal{D}_s(A^{-1}) = t \oslash q$

Note also that even though in the loop of Algorithm 2 that calls BCG to solve for the remaining right-hand sides a specific partitioning appears ($\nu - 1$ groups of p right-hand sides each), one is free to organize the solution of the right-hand sides differently: for example, to call BCG only once with block size $s - p$.

3.2. Practical considerations. The backbone of Algorithm 2 is INIT-BCG that consists of a BCG phase and a projection phase. In Algorithm 2, INIT-BCG is called twice to construct MOD-INIT-BCG followed by the stochastic estimation phase. So, we first consider the cost of INIT-BCG, ignoring terms not involving n. The memory

required in each iteration are $4p$ vectors when solving the block of seed systems and $2(s-p)$ in the projection phase, for a total of $2(s+p)$ n-vectors. Clearly, the memory requirements of MOD-INIT-BCG are the same. It is worth noting that in case n, s, p are very large, we could partition the right-hand sides and apply INIT-BCG and MOD-INIT-BCG repeatedly, a subset at a time. The computational cost per iteration of INIT-BCG is p MVs with matrix A, p^2+p dot products (DOTs) and 3 rank-p updates for the BCG step and another $(s-p)p$ DOTs and $2(s-p)$ rank-p updates. In these counts, DOTs and updates are operations on vectors or blocks whose leading dimension is n. To these, we must add the cost of thin-QR, that is approximately $2np^2$ operations per iteration. The dominant cost of the stochastic estimation phase is $4s$ operations on n-vectors.

Regarding the tolerances used in MOD-INIT-BCG, tol is the accuracy sought for the solution vectors passed to the stochastic estimator. Tolerance tol_1 is chosen to be small so that the first call to INIT-BCG generates a large Krylov subspace. This implies more iterations when seeding and possibly re-emergence of eigenvectors as noted earlier. Tolerance tol_2 is larger than tol_1 and is selected to deflate again such components. Therefore the values of these tolerances are related. We suggest to store a few direction blocks P_i with subscripts at some distance apart and once the block-seed system is solved, e.g. after m iterations, to compute the ratio $\frac{\|\Pi_i R_{m+1}^{(j)}\|}{\|R_{m+1}^{(j)}\|}$ for these blocks. Subsequently, to choose tol_2 so as to ensure that the second call to INIT-BCG will deflate again all non-seed residuals whose aforementioned ratio is above some chosen tolerance. Note that in the stochastic estimation phase we could use $X^{(1)}$ instead of $\tilde{X}^{(1)}$ as a solution of the block-seed since the former is computed with smaller tolerance. Also, there is the possibility that some extremal eigenvalues are well separated and their eigenvectors rapidly approximated, in which case MOD-INIT-BCG will not be as effective at removing them from the non-seed residuals. In that case, it might be helpful to first explicitly approximate and deflate these eigenvectors, e.g. by preprocessing using a Lanczos-based algorithm, so that the subsequent call to MOD-INIT-BCG will work on the orthogonal complement of these eigenvectors.

We next compare our approach with related methods that use approximate invariant subspaces [1, 34]. These algorithms first solve for one or more right-hand sides, generating approximate eigenvectors in the process. If only one right-hand side is used to approximate the invariant subspace, as in [1], it might be necessary to perform many iterations so as to capture a sufficient number of eigenvectors. If, as in [34], many right-hand sides are used to generate approximate eigenvectors, the memory needs also increase. In contrast, our approach takes advantage of the fact that all right-hand sides are readily available (an assumption not made in the aforementioned works) and obtains the initial guesses by solving for a block-seed. Since we are not interested in eigenvectors and deflation takes place only implicitly, our method is able to reach its aims with only limited demands on memory. Fig. 3.1 illustrates some of the aforementioned issues when solving $AX = B$ with matrix A of order $n = 10000$ constructed (using an orthogonal similarity tranformation built from a random matrix) to have condition number equal to n. The right-hand side B consisted of $s = 2$ Rademacher vectors the first one of which was selected as seed. Observe that after 900 iterations, the first 300 direction vectors are no longer deflated well. The plot confirms that the removal of direction vectors produced in later stages of the algorithm is more effective because of less round-off.

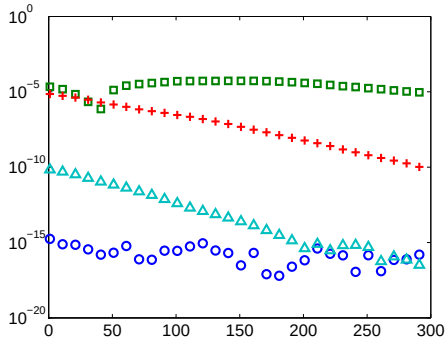
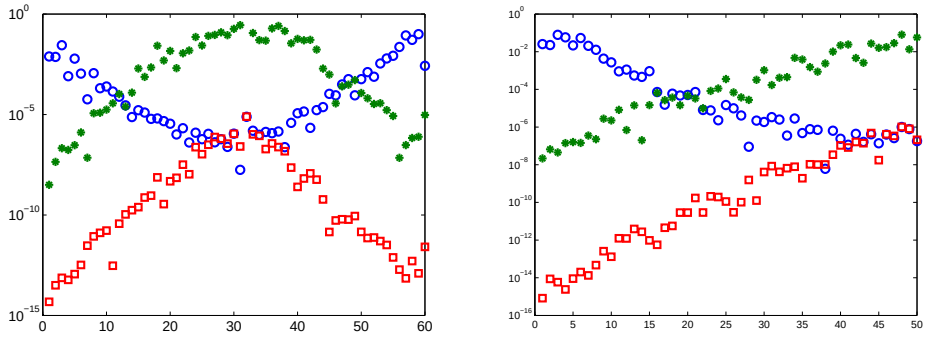


Fig. 3.1: Solving $AX = B$ for an spd matrix of size $n = 10000$ constructed to have spectral condition number equal to n and $s = 2$. Plots of :$\|\Pi_i R_{301}^{(2)}\|/\|R_{301}^{(2)}\|$ ("$\circ$") and $\|\Pi_i R_{901}^{(2)}\|/\|R_{901}^{(2)}\|$ ("$\square$") for $i = 1 : 10 : 300$; $\|\Pi_i R_{901}^{(2)}\|/\|R_{901}^{(2)}\|$ ("+") for $i = 301 : 10 : 600$; $\|\Pi_i R_{901}^{(2)}\|/\|R_{901}^{(2)}\|$ ("$\triangle$") for $i = 601 : 10 : 900$. Π_i is as defined in Eq. (3.2). Here each $R^{(j)}$ consists of only one column.

3.3. Matrix-free Jacobi preconditioning . For dense matrices without any specific structure, preconditioning is more of an art. If the diagonal of the underlying matrix was available, then one could try simple Jacobi preconditioning. On the other hand, if methods for stochastic estimation of the diagonal of the inverse, like Algorithm 2, or the diagonal of the matrix, as in [9], are successful, it becomes of interest to use them to generate (matrix-free) diagonal preconditioners. To the best of our knowledge, this idea has not been considered elsewhere. From these two possibilities, we sketch and test the effect of using the estimate for the diagonal of the inverse as preconditioner. One version, that can be termed "static", is to obtain an estimate for the diagonal of the inverse after solving p systems (the seed block) and then apply it as preconditioner for the remaining $s - p$ systems. One could also envision a fully dynamic preconditioner, which is updated after every (block) solve to incorporate the latest diagonal estimate. In the next section we show results with the static preconditioner.

4. Numerical experiments . We present numerical experiments that illustrate our findings. Since our main focus is in data uncertainty quantification, we experiment primarily with parameterized (model) covariance matrices. For completeness, we also include some experiments that indicate the applicability of our techniques for general (sparse) matrices. For economy of presentation, we enclose in parentheses after the name of the method the selected tolerances, e.g. INIT-BCG(tol) and MOD-INIT-BCG($\text{tol}_1, \text{tol}_2$). In all cases, n, s, p denote the order of the matrix, the number of right-hand sides, and the block size used. Throughout our experiments, the right-hand sides used were Rademacher vectors and the tolerance tol was set equal to 10^{-5} for all methods. The mean squared relative error and absolute mean relative error for an estimation, say $\mathcal{D}_s(A^{-1})$ of the diagonal of the inverse are the values $\frac{1}{N}\sum_i^N((d_i - \hat{d}_i)/d_i)^2$ and $\frac{1}{N}\sum_i^N |(d_i - \hat{d}_i)/d_i|$ respectively, where $d_i = [A^{-1}]_{i,i}$ and $\hat{d}_i = [\mathcal{D}_s(A^{-1})]_i$. We first report on experiments written in MATLAB (64-bit version 7.10 R2010a) that ran on a machine with two 4-core Intel Xeon E5330 processors set



(a) (b)

Fig. 4.1: Contributions to residual of extreme eigenvectors using model covariance matrices. Curves marked with "$\circ$" and "$*$" are for INIT-BCG with tolerances tol_1 and tol_2 respectively and the curve marked with "$\square$" is for MOD-INIT-BCG$(\text{tol}_1, \text{tol}_2)$. *a)* Eigenvector contributions corresponding to the smallest 30 (indexed from 1 to 30 in the x-axis) and 30 largest (indexed from 31 to 60) eigenvalues; $n = 5000$, $\theta = 1, \kappa = 1$, $p = 4$, $\text{tol}_1 = 10^{-10}$, $\text{tol}_2 = 10^{-5}$ *b)* Contributions corresponding to the 50 smallest eigenvalues; $n = 10000$, $\theta = 3/4, \kappa = 2$, $p = 10$, $\text{tol}_1 = 10^{-10}$, $\text{tol}_2 = 10^{-6}$.

at 1.6 GHz. The model covariance matrices are defined to have the general form (see e.g. [8]):

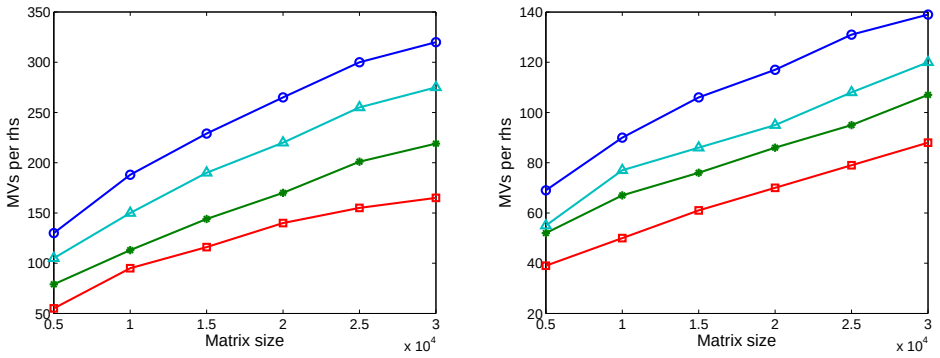
$$[A]_{i,i} = 1 + i^{\theta}, \qquad [A]_{i,j} = \frac{1}{|i-j|^{\kappa}} \quad (\text{when } i \neq j), \text{ for } i, j = 1, ..., n, \tag{4.1}$$

where θ, κ are real parameters, with $\kappa \geq 1$. Such matrices exhibit a decaying behavior away from the main diagonal modeling the fact that some features are far less correlated than others (cf. [36] and references therein). Decay is controlled by the exponent κ while θ controls the diagonal, with higher values increasing the condition number. Please note that since we assume that the matrix is only implicitly known, we cannot take advantage of the special structure of these matrices.

We first examine the effectiveness of INIT-BCG and MOD-INIT-BCG in suppressing eigenvectors using two test matrices. The plots in Fig. 4.1 depict for each matrix the contribution of selected eigenvectors to a non-seed residual. We observe that MOD-INIT-BCG is more effective at removing eigenvectors from non-seed residuals because direction blocks that are reproduced by the repeated call to INIT-BCG are better deflated (even compared to applying INIT-BCG only for tol_2).

Fig. 4.2 compares methods BCG, the hybrid block-seed method of [14] (abbreviated as BSM), INIT-BCG and MOD-INIT-BCG for two model covariance matrices. We set $s = 300$ and vary the matrix size. We see that MOD-INIT-BCG takes significantly fewer MVs per right-hand side than all other methods. Note also that even though BSM requires a smaller number of MVs per right-hand side than INIT-BCG, its actual runtime is greater.

Table 4.1 shows results with MOD-INIT-BCG as s increases. We tabulate the mean squared relative error achieved with the stochastic estimator as well as the relative error for the trace of the inverse. Regarding the latter, we remind the reader that



(a) $p = 10$, $\theta = 1$, $\kappa = 1$ (b) $p = 10$, $\theta = 3/4$, $\kappa = 2$

Fig. 4.2: Comparative performance of methods described in this paper for $n = 5000 : 5000 : 30000$ and $s = 300$. BCG is depicted with "$\circ$", INIT-BCG with "$\triangle$", BSM with "$*$" and MOD-INIT-BCG with "$\square$".

Table 4.1: Performance results using a model covariance matrix of order $n = 4000$ using $\kappa = 2$, $\theta = 1/2$ and $p = s/10$. (Col. 2): Mean relative error of estimator with MOD-INIT-BCG$(10^{-10}, 10^{-4})$. (Col. 3): Relative error for trace(A^{-1}). (Col. 4): Ratio of MVs using CG s times over using MOD-INIT-BCG$(10^{-10}, 10^{-4})$.

s	mre	rel. error for trace	MV speedup
20	$1.10 \cdot 10^{-4}$	0.0072	1.56
30	$1.01 \cdot 10^{-4}$	0.0059	1.73
40	$9.90 \cdot 10^{-5}$	0.0051	1.90
50	$9.86 \cdot 10^{-5}$	0.0046	2.11
60	$9.81 \cdot 10^{-5}$	0.0042	2.31

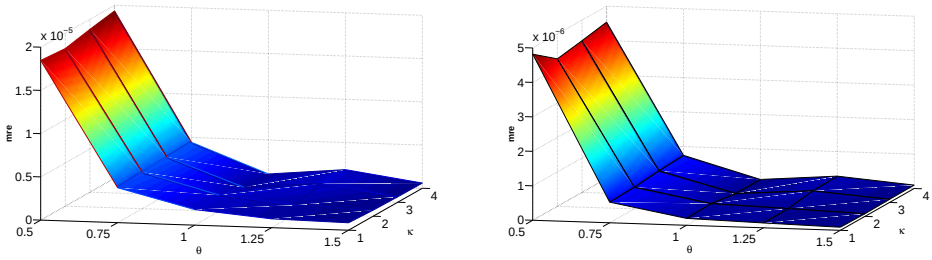
MOD-INIT-BCG is not currently optimized for the trace for which other methods might be more preferable. We use MOD-INIT-BCG$(10^{-10}, 10^{-4})$ with $p = s/10$ for a model covariance matrix of order $n = 4000$ with $\kappa = 2$, $\theta = 1/2$. The last column shows the ratio of the total number of MVs for CG over the total number of MVs using the proposed method. The ratios reveal the sublinear increase of the MVs relative to the independent runs of CG which in turn permits using more sample vectors to achieve a better statistical estimation for the error.

Table 4.2 shows the ratio of MVs required by CG compared to MOD-INIT-BCG for $s = 100,\ 200,\ 300$ using model covariance matrices of size $n = 10000,\ 20000,\ 30000$ for fixed $\kappa = 2$ and varying $\theta = 3/4,\ 1,\ 5/4$. We see that MOD-INIT-BCG significantly reduces the average MV count as s increases. Moreover, the overhead incurred by larger blocksizes because of the two phases of MOD-INIT-BCG is amortized.

We next consider the quality of results obtained with Algorithm 2 as the condition number of the model covariance matrices grows for various degrees of decaying behavior away from the diagonal. The plots in Fig. 4.3 illustrate the mean squared relative error obtained for matrices of sizes $n = 2000, 10000$, $s = 300$ sampling vectors and parameter values in the range $(\kappa, \theta) \in [1 : 4] \times [0.5 : 0.25 : 1.5]$. For example,

Table 4.2: Ratio of MVs of CG over MOD-INIT-BCG for $s = 100$, $s = 200$ and $s = 300$, and various values of n and p for MOD-INIT-BCG. The model covariance matrices were constructed with $\kappa = 2$ and $\theta = 3/4,\ 1,\ 5/4$.

	$\theta = 3/4$			$\theta = 1$			$\theta = 5/4$		
s	100	200	300	100	200	300	100	200	300
$n = 10000$									
p=5	2.53	2.93	3.01	3.62	4.00	4.16	5.45	6.00	6.35
p=10	2.54	3.20	3.60	3.96	4.74	5.10	6.75	8.40	9.10
p=15	2.73	3.58	4.10	4.18	5.51	6.15	6.66	9.30	9.81
$n = 20000$									
p=5	2.35	2.68	2.82	3.64	4.15	4.21	5.77	6.49	6.77
p=10	2.46	3.02	3.27	3.48	4.20	4.55	5.65	6.90	7.34
p=15	2.48	3.34	3.77	3.44	4.45	5.74	6.15	7.43	7.99
$n = 30000$									
p=5	2.22	2.35	2.20	3.70	3.99	4.11	5.80	6.33	6.75
p=10	2.31	2.51	2.97	3.72	4.06	4.32	5.81	6.39	6.91
p=15	2.41	2.93	3.33	3.81	4.19	4.69	6.07	6.65	7.24



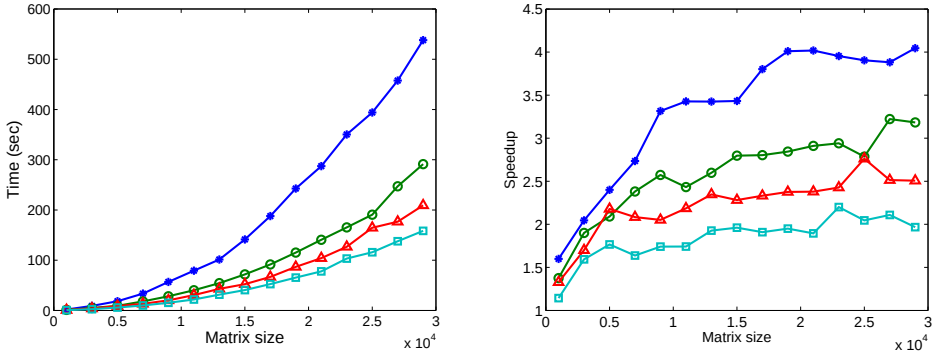
(a) $n = 2000$ (b) $n = 10000$

Fig. 4.3: Mean squared relative error of stochastic estimation Algorithm 2 for model covariance matrices with $(\kappa, \theta) \in [1 : 4] \times [0.5 : 0.25 : 1.5]$ and $s = 300$.

the condition number of the covariance matrix with $(\kappa, \theta) = (4, 1.5)$ and $n = 10000$ is at least 10^6; it is worth noting that this is much higher than what is encountered in practice in data uncertainty quantification. The block size is set to $p = 50$, and convergence tolerances are set to $\text{tol}_1 = 10^{-10}, \text{tol}_2 = 10^{-4}, \text{tol} = 10^{-5}$. The method achieves mean relative errors of the order 10^{-5}, which is again far more than what is required in practice. It is worth noting also that in separate experiments we observe errors of order 10^{-3} using fewer than $s = 100$ vectors.

Fig. 4.4 shows results with the preconditioner described in Section 3.3 for a model covariance matrix with parameters $\theta = 1/2$ and $\kappa = 2$, matrix size ranging from $n = 1000$ to 29000 and $s = 300$. The code for this experiment is written in C with calls to the multithreaded ESSL library, to run on an IBM Power7 based PS702 blade system with 16 cores and 64GB of RAM[1]. The left plot shows the baseline

[1]IBM, the IBM logo, Power, Power7, and ibm.com are trademarks or registered trademarks of



(a) (b)

Fig. 4.4: Performance of MOD-INIT-BCG for model covariance matrices of sizes ranging from $n = 1000$ to 29000 and $s = 300$ for values $\kappa = 2, \theta = \frac{1}{2}$ and block sizes $p = 10$ (marked "$*$"), 20 (marked "$\circ$"), 30 (marked "$\triangle$") and 50 (marked "$\square$"). *a*) Without preconditioning. *b*) Using matrix-free diagonal preconditioning as described in Section 3.3.

performance, without preconditioning, while the right plot shows the speedups over these values with the static diagonal preconditioner. The preconditioner significantly improves the performance of MOD-INIT-BCG, especially for larger matrices.

The purpose of the next experiment is to provide an illustration of the performance of the methods under consideration for matrices that are no longer of the model covariance variety but are spd and sparse. These are matrices KUU, PRES_POISSON and TREFETHEN[2] from the University of Florida Sparse Matrix collection[3]. Table 4.3 shows the average number of MVs per right-hand side for $s = 80,\ 160$ for BCG, BSM, INIT-BCG and MOD-INIT-BCG using block sizes $p = 2,\ 4$ and 8. For comparison, we include in the second column the number of MVs required by simple CG when applied to only one such right-hand side. We consider that a method has converged when the residual (Frobenius) norm becomes less than 10^{-5}. On the other hand, the mean squared relative errors for the three matrices under consideration are 0.69, 3.39 and 1.64×10^{-4} respectively for $s = 80$, and 0.49, 2.25 and 1.17×10^{-4} respectively for $s = 160$. Evidently, MOD-INIT-BCG converges in fewer MVs than all other methods. Moreover, the improvement is greater as the number of right-hand sides increases. The lowest performance of MOD-INIT-BCG is for matrix PRES_POISSON, whose leading eigenvalues are isolated while the trailing ones are clustered. In this case, the method is unable to effectively suppress the corresponding eigenvectors in the second call to INIT-BCG.

International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information website.

[2]This matrix is from Problem 7 of the "SIAM 100-digit challenge" that sought a very good approximation of element $[A^{-1}]_{11}$; cf. `http://www-m3.ma.tum.de/m3old/bornemann/challengebook/Chapter7.`

[3]`http://www.cise.ufl.edu/research/sparse/matrices`

Table 4.3: Average number of MVs per right-hand side for sparse spd matrices. For comparison we list next to the "$p = 2$" markups the MVs for CG for a single right-hand side.

			BCG	INIT-BCG	BSM		MOD-INIT-BCG	
	n	$\kappa(A)$	$s = 80$	$s = 80$	$s = 80$	$s = 160$	$s = 80$	$s = 160$
KUU	7102	$O(10^4)$						
p=2, (642)			528	392	408	360	273	261
p=4			416	256	325	310	205	190
p=8			292	182	243	236	164	141
PRES_POISSON	14822	$O(10^6)$						
p=2, (2641)			2269	1602	2001	1950	1460	1430
p=4			1907	1401	1750	1704	1350	1285
p=8			1560	1067	1500	1462	930	862
TREFETHEN	20000	$O(10^5)$						
p=2, (1631)			1110	457	813	665	380	355
p=4			725	365	596	546	268	235
p=8			503	251	427	415	224	184

5. Concluding remarks. We presented a framework for a central problem in data uncertainty quantification, in particular the stochastic estimation of the diagonal of the precision matrix. This is based on a method consisting of repeated application of a special block-seed CG method followed by BCG to solve the systems that arise in the course of the stochastic estimation. We showed that this new framework causes a significant reduction in the overall cost if the underlying covariance matrix is dense, spd, and a large number of stochastic samplings is required. The method is matrix-free and also naturally leads to the construction of a diagonal preconditioner that appears to be very effective for model covariance matrices with decaying off-diagonal entries. Limited experimental data suggests that the underlying iterative method could also be useful for more general sparse matrices and other application areas, however more work is required in that direction.

Acknowledgments. The last author is grateful to Ahmed Sameh for his hospitality and support at Purdue University during his sabbatical leave where part of this work was accomplished; and to Michela Redivo-Zaglia for her hospitality in Padova and for encouraging our participation to SC'2011. We thank Andreas Stathopoulos, Giorgos Kollias, Marilena Mitrouli and Paraskevi Fika for useful discussions. We are especially grateful to Gérard Meurant, editor in charge for his patient and fair handling of the paper. We would also like to thank one of the anonymous reviewers whose constructive criticism in matters of content and form helped us turn this into a better paper. In our opinion, her/his gracefully written reports were exemplars of peer reviews.

REFERENCES

[1] A. Abdel-Rehim, R. Morgan, D. Nicely, and W. Wilcox, *Deflated and restarted symmetric Lanczos methods for eigenvalues and linear equations with multiple right-hand sides*, SIAM J. Sci. Comput., 32 (2010), pp. 129–149.

[2] A. Abdel-Rehim, R. Morgan, and W. Wilcox, *Improved seed methods for symmetric positive definite linear equations with multiple right-hand sides.* arXiv:0810.0330v1 [math-ph], 2008.

[3] M. Anitescu, J. Chen, and L. Wang, *A matrix-free approach for solving the Gaussian process maximum likelihood problem*, SIAM J. Sci. Comput., (To appear).
[4] H. Avron and S. Toledo, *Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix*, J. ACM, 58 (2011), p. 8.
[5] Z. Bai, M. Fahey, and G. Golub, *Some large-scale matrix computation problems*, J. Comput. Appl. Math., 74 (1996), pp. 71–89.
[6] Z. Bai and G. Golub, *Bounds for the trace of the inverse and the determinant of symmetric positive definite matrices*, Ann. Num. Math., 4 (1997), pp. 29–38.
[7] C. Bekas, A. Curioni, and I. Fedulova, *Low cost high perf. uncertainty quantification*, in Worskhop on High Performance Computational Finance, Supercomputing'09, Portland, Portland, Oregon, 2009.
[8] C. Bekas, A. Curioni, and I. Fedulova, *Low-cost data uncertainty quantification*, Concurrency and Computation: Practice and Experience, (2011).
[9] C. Bekas, E. Kokiopoulou, and Y. Saad, *An estimator for the diagonal of a matrix*, Appl. Numer. Math., 57 (2007), pp. 1214–1229.
[10] C. Bekas, E. Kokiopoulou, and Y.Saad, *Computation of large invariant subspaces using polynomial filtered Lanczos iterations with applications in density functional theory*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 397–418.
[11] R. Bouyouli, K. Jbilou, R. Sadaka, and H. Sadok, *Convergence properties of some block Krylov subspace methods for multiple linear systems*, J. Comp. Appl.Math., 196 (2006), pp. 498–511.
[12] C. Brezinski, P. Fika, and M. Mitrouli, *Moments of a linear operator, with applications to the trace of the inverse of matrices and the solution of equations*, Numer. Lin. Alg. Appl., (2011).
[13] G. Cao, L. Bachega, and C. Bouman, *The sparse matrix transform for covariance estimation and analysis of high dimensional signals*, IEEE Trans. Image Proc., 20 (2011), pp. 625–640.
[14] T. Chan and W. Wan, *Analysis of projection methods for solving linear systems with multiple right-hand sides*, SIAM J. Sci. Stat. Comput., 18 (1997), pp. 1698–1721.
[15] J. Chen, *A deflated version of the block conjugate gradient algorithm with an application to Gaussian process maximum likelihood estimation*, Preprint ANL/MCS-P1927-0811, Argonne Nat'l. Lab., 2011.
[16] L. Du, T. Sogabe, B. Yu, Y. Yamamoto, and S. L. Zhang, *A block IDR(s) method for nonsymmetric linear systems with multiple right-hand sides*, J. Comput. Appl. Math., 235 (2011), pp. 4095–4106.
[17] L. Giraud, D. Ruiz, and A.Touhami, *A comparative study of iterative solvers exploiting spectral information for spd systems*, SIAM J. Sci. Comput., 27 (2006), pp. 1760–1786.
[18] G. Golub and G. Meurant, *Matrices, Moments and Quadrature with Applications*, Princeton Univ. Press, 2010.
[19] G. Golub, D. Ruiz, and A. Touhami, *A hybrid approach combining Chebyshev filter and conjugate gradient for solving linear systems with multiple right-hand sides*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 774–795.
[20] M. Gutknecht, *Block Krylov space methods for linear systems with multiple right-hand sides: An introduction*, in Modern Mathematical Models, Methods and Algorithms for Real World Systems, A. Siddiqi, I. Duff, and O. Christensen, eds., Anamaya Publishers, New Delhi, India, 2007, pp. 420–447.
[21] J. Hartlap, P. Simon, and P. Schneider, *Why your model parameter confidences might be too optimistic – unbiased estimation of the inverse covariance matrix*, Astronomy & Astrophysics, 464 (2007), pp. 399–404.
[22] M. Hutchinson, *A stochastic estimator for the trace of the influence matrix for Laplacian smoothing splines*, Comm. Stat. - Simul. and Comput., (1989).
[23] M. Kilmer, E. Miller, and C. Rappaport, *QMR-based projection techniques for the solution of non-Hermitian systems with multiple right-hand sides*, SIAM J. Sci. Comput., 23 (2001).
[24] O. Ledoit and M. Wolf, *A well-conditioned estimator for large-dimensional covariance matrices*, J. Multiv. Anal., 88 (2004), pp. 365–411.
[25] L. Lin, C. Yang, J. Meza, J. Lu, L. Ying, and W. E, *SelInv – an algorithm for selected inversion of a sparse symmetric matrix*, ACM Trans. Math. Softw., 37 (2011), pp. 40:1–40:19.
[26] G. Meurant, *Estimates of the trace of the inverse of a symmetric matrix using the modified Chebyshev algorithm*, Num. Alg., 51 (2009), pp. 309–318.
[27] D. O'Leary, *The block conjugate gradient algorithm and related methods*, Lin. Alg. Appl., 29 (1980), pp. 293–322.
[28] D. S. Oliver, *Calculation of the inverse of the covariance*, Math. Geol., 30 (1998), pp. 911–933.

[29] M. Parks, E. de Sturler, G. Mackey, D. Johnson, and S. Maiti, *Recycling Krylov subspaces for sequences of linear systems*, SIAM J. Sci. Comput., 28 (2006), pp. 1651–1674.
[30] B. N. Parlett, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations*, Lin. Alg. Appl., 29 (1980), pp. 323–346.
[31] Y. Saad, *On the Lanczos method for solving symmetric systems with several right hand sides*, Math. Comp., 48 (1987), pp. 651–662.
[32] V. Simoncini and E. Gallopoulos, *An iterative method for nonsymmetric systems with multiple right-hand sides*, SIAM J. Sci. Comput., 16 (1995), pp. 917–933.
[33] C. F. Smith, A. F. Peterson, and R. Mittra, *A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields*, IEEE Trans. Ant. Prop., 37 (1989), pp. 1490–1493.
[34] A. Stathopoulos and K. Orginos, *Computing and deflating eigenvalues while solving multiple right-hand hide linear systems with an application to quantum chromodynamics*, SIAM J. Sci. Comput., 32 (2010), pp. 439–462.
[35] G. Stevens, *On the inverse of the covariance matrix in portfolio analysis*, J. Finance, 53 (1998), pp. 1821–1827.
[36] J. Tang and Y. Saad, *A probing method for computing the diagonal of a matrix inverse*, Num. Lin. Alg. Appl., 19 (2012), pp. 485–501.
[37] K. Visweswariah, P. Olsen, R. Gopinath, and S. Axelrod, *Maximum likelihood training of subspaces for inverse covariance modeling*, in Proc. ICASSP, vol. 1, 2003, pp. 848–851.