SIAM J. SCI. COMPUT.
Vol. 38, No. 5, pp. A2934–A2963

A MULTILEVEL APPROACH FOR COMPUTING THE LIMITED-MEMORY HESSIAN AND ITS INVERSE IN VARIATIONAL DATA ASSIMILATION*

KIRSTY L. BROWN†, IGOR GEJADZE‡, AND ALISON RAMAGE†

Abstract. Use of data assimilation techniques is becoming increasingly common across many application areas. The inverse Hessian (and its square root) plays an important role in several different aspects of these processes. In geophysical and engineering applications, the Hessian-vector product is typically defined by sequential solution of a tangent linear and adjoint problem; for the inverse Hessian, however, no such definition is possible. Frequently, the requirement to work in a matrix-free environment means that compact representation schemes are employed. In this paper, we propose an enhanced approach based on a new algorithm for constructing a multilevel eigenvalue decomposition of a given operator, which results in a much more efficient compact representation of the inverse Hessian (and its square root). After introducing these multilevel approximations, we investigate their accuracy and demonstrate their efficiency (in terms of reducing memory requirements and/or computational time) using the example of preconditioning a Gauss–Newton minimization procedure.

Key words. data assimilation, inverse Hessian, limited memory, preconditioning, multigrid

AMS subject classifications. 65K05, 65K10, 15A09, 15A29

DOI. 10.1137/15M1041407

1. Introduction and background material.

Methods of data assimilation (DA) have become an important tool for analysis of complex physical phenomena in various fields of science and technology. These methods allow us to combine mathematical models, data resulting from instrumental observations, and prior information. In particular, variational approaches have proven to be particularly useful for solving high-dimensional DA problems arising in geophysical and engineering applications involving complex fluid flow models. The problems of variational DA can be formulated as optimal control problems (see, for example, [12, 28]) to find unknown model parameters such as initial and/or boundary conditions, right-hand sides in the model equations (forcing terms), and distributed coefficients. Equivalently, variational DA can be considered as a special case of the maximum a posteriori probability estimator in a Bayesian framework [10]. Variational DA, implemented in the form of *incremental 4D-Var* [8, 37], is currently a preferred method for operational forecasting in meteorology and oceanography (more recently also in the form of *ensemble 4D-Var*; see, for example, [29]).

The importance of the Hessian matrix and its inverse in variational DA for geophysical applications is underlined in [39], although this has been a well established fact for decades in areas of statistics such as nonlinear regression (see, for example, [1]). Some relevant applications of the inverse Hessian are highlighted in section 1.2.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section September 25, 2015; accepted for publication (in revised form) July 20, 2016; published electronically September 22, 2016.

http://www.siam.org/journals/sisc/38-5/M104140.html

Funding: The second author's work was supported by the UK Natural Environment Research Council (NERC grant NE/J018201/1).

†Department of Mathematics and Statistics, University of Strathclyde, Glasgow G1 1XH, Scotland (kirsty.l.brown@strath.ac.uk, a.ramage@strath.ac.uk).

‡IRSTEA-Montpellier, 361 Rue Jean Francois Breton, BP 5095, France (igor.gejadze@irstea.fr).

A special feature of working with the Hessian for very high-dimensional problems is that neither the Hessian nor its inverse can be directly accessed in matrix form. While the Hessian-vector product can be computed by solving a sequence of the tangent linear and adjoint problems, no such option exists for defining the inverse Hessian-vector product (or the inverse square root Hessian-vector product, which is also relevant in many applications). One obvious approach is therefore to consider limited-memory schemes for computing and storing the inverse Hessian (or its square root). In this context, the idea of a multilevel framework becomes relevant. This is due to the fact that the inverse Hessian is, in essence, an approximation of the posterior covariance matrix and, if the initial flow field is considered as a spatially distributed control, then the correlations of different lengths between the flow field values can be described at different levels of spatial discretization.

Multigrid methods were initially developed for solving elliptic partial differential equations (PDEs) [3, 5] and have since been extended for solving PDEs of different types; see, for example, [41]. One key modern area of application of multigrid methods is in solving PDE-constrained optimization or inverse problems; see the review paper [4]. Here the multigrid solver is applied directly to the optimality system, which includes the original model, its adjoint, and the optimality condition. Some elements of the multigrid approach have been utilized previously in variational DA algorithms in meteorology and oceanography, but a complete multigrid algorithm has been considered only recently in [11]. Multigrid methods can also used for solving eigenvalue problems, which is most relevant to the work presented here. The usual multigrid approach in this context is to treat the eigenvalue problem as a nonlinear equation and apply a nonlinear multigrid solver [7, 21, 24]. Alternatively, an outer eigenvalue solver such as Rayleigh quotient iteration can be employed, which requires the solution of systems of linear equations with a shifted coefficient matrix using multigrid as an inner solver [38]. A third approach uses a standard eigenvalue solver (such as Lanczos [26] or Arnoldi [2]) with multigrid as a preconditioner. This type of method is reviewed in [25].

In this paper we develop a general multilevel eigenvalue decomposition of a given symmetric operator. Given its spectral decomposition, an operator A, say, can be approximated by a finite number of its eigenvalues and eigenvectors. To achieve a desired approximation quality (in terms of a specified distance between the exact and approximated operators) a certain number of eigenpairs must be used, dependent on the eigenvalue distribution. However, for high-dimensional problems, the computationally feasible number of eigenpairs (in terms of available storage, for example) may be too small to achieve any useful approximation quality. Thus, a single-level eigenvalue decomposition approach has its limitations. Our proposed *multilevel eigenvalue decomposition algorithm* (see section 2) involves an outer multilevel loop that provides an incomplete eigenvalue decomposition (using Lanczos) of the operator at each level, resulting in a final approximation involving eigenpairs associated with each discretization level. Note that if A is not symmetric, the same technique could be applied to $A^T A$.

The multilevel technique described in section 2 allows us to build limited-memory approximations to A^{-1} and $A^{-1/2}$ which, within a fixed memory framework, are much better than their single-level spectral counterparts. These approximations could be used in many situations, for example, as preconditioners for solving systems of linear equations, across multiple application areas; a specific example is given in section 2.3.4. In this paper, we use the technique as an efficient way of approximating the inverse Hessian in variational DA. For operational DA problems, we also introduce a

second idea, namely, decomposition of the Hessian into local sensor-based Hessians (see section 3). Although this is distinct from the multilevel eigenvalue decomposition, the latter provides a framework for its practical implementation. In section 4 we describe three different implementations of the algorithm, before exploring their accuracy and efficiency in section 5.

1.1. DA problem. We consider a nonlinear evolution model with an unknown initial state (the *analysis*), which must be retrieved using some prior guess (the *background*) and incomplete observation data of the state evolution. Let X be a Hilbert space of functions on a spatial domain Ω with associated inner product $\langle\cdot,\cdot\rangle_X$, and let $Y = L_2(0,T;X)$ for some time interval $[0,T]$ with scalar product

$$\langle f,g\rangle_Y = \int_0^T \langle f(t)g(t)\rangle_X dt, \qquad \forall f,g \in Y.$$

We will model a physical process on Ω that is described by a function $\varphi \in Y$ satisfying the evolution equation

$$\begin{cases} \dfrac{\partial\varphi(t)}{\partial t} = F(\varphi(t)) + f(t), & t \in [0,T], \\ \varphi(0) = u, \end{cases} \tag{1.1}$$

where $u \in X$, $f \in Y$, and F is a nonlinear operator mapping X into X. We assume that, for a given $u \in X$ and $f \in Y$ such that $\langle f,f\rangle_Y < \infty$, there exists a unique solution $\varphi \in Y$ to (1.1). For a particular u, the model (1.1) can also be represented in operator form via a control-to-state mapping R such that

$$\varphi = R(u).$$

The above framework is very general: as a concrete example of a specific evolutionary process, in the numerical experiments in section 5 we use Burgers' equation as a basic prototype equation of atmospheric dynamics.

Let u_{true} be the true initial state in (1.1). We define the input data as follows. First, we have a prior guess, or *background* function, $u_b \in X$ such that $u_b = u_{true} + \xi_b$, where $\xi_b \in X$ is the *background error*. Second, we have some observations represented by a function φ_{obs} in a finite-dimensional space $Y_{obs} = L_2(\mathbb{R}^M)$, called the *observation space*. Introducing the bounded linear observation operator $C : Y \to Y_{obs}$, we may write $\varphi_{obs} = C(R(u_{true})) + \xi_o$, where $\xi_o \in Y_{obs}$ is the *observation error*. Assuming that ξ_b and ξ_o are normally distributed, unbiased, and mutually uncorrelated, we define the covariance operators $V_b(\cdot) = E[\langle\cdot,\xi_b\rangle_X\ \xi_b]$ and $V_o(\cdot) = E[\langle\cdot,\xi_o\rangle_{Y_{obs}}\ \xi_o]$, where $E[\cdot]$ is the expectation. We further assume that the operators V_b and V_o are positive definite, hence invertible.

We now introduce the cost function

$$J(u) = \frac{1}{2}\langle V_b^{-1}(u-u_b), u-u_b\rangle_X + \frac{1}{2}\langle V_o^{-1}(CR(u)-\varphi_{obs}), CR(u)-\varphi_{obs}\rangle_{Y_{obs}}$$

and formulate the following DA problem with the aim of identifying the initial state in (1.1): for a given $f \in Y$, find $\bar{u} \in X$ such that the cost function (1.1) is minimized over all $u \in X$, that is,

$$\bar{u} = \arg\min_u J(u). \tag{1.2}$$

We denote the error in the solution of this optimal control problem (the so-called analysis error) by $\delta u = \bar{u} - u_{true}$ and assume δu is unbiased, that is, $E[\delta u] = 0$, with the covariance operator $V_{\delta u}$ defined by $V_{\delta u}(\cdot) = E[\langle\cdot,\delta u\rangle_X\ \delta u]$.

Assuming that the operator F is continuously Fréchet differentiable, for functions $v \in X$ and $\psi \in Y$, the tangent linear model associated with (1.1) is given by

$$\begin{cases} \dfrac{\partial \psi(t)}{\partial t} = F'(\varphi(t))\psi(t), \\ \psi(0) = v \end{cases} \tag{1.3}$$

(see, for example, [16]), where $\varphi(t) \equiv \varphi(t,u)$ depends on u. Recalling the control-to-state operator $R(u)$, and defining the associated tangent linear operator $R'(u)$ for a given $u \in X$ by

$$R'(u)v = \lim_{\tau \to 0} \frac{R(u+\tau v) - R(u)}{\tau}, \ \forall v \in X, \tag{1.4}$$

[30], we may write model (1.3) as $\psi = R'(u)v$.

A key role in variational DA is assigned to the Hessian of the following auxiliary DA problem (see [16]): find $v \in X$ such that the cost function

$$J_1(v) = \frac{1}{2}\langle V_b^{-1} v, v\rangle_X + \frac{1}{2}\langle V_o^{-1} CR(u)v, CR(u)v\rangle_{Y_{obs}} \tag{1.5}$$

is minimized over all $v \in X$. Defining the adjoint $R'^*(u)$ of the tangent linear operator $R'(u)$ by

$$\langle v, R'^*(u)v^*\rangle_X = \langle R'(u)v, v^*\rangle_Y, \ \forall v \in X, \ \forall v^* \in Y, \tag{1.6}$$

we may write the Hessian in operator form as

$$\mathcal{H}(u) = V_b^{-1} + R'^*(u)C^* V_o^{-1} CR'(u), \tag{1.7}$$

where C^* is adjoint to C. We denote the Hessian by $\mathcal{H}(u)$ to emphasize its dependence on u through φ in R' and R'^*. Note that for functions $v \in X$, $\psi \in Y$, and $\psi^* \in Y$, the adjoint model $v = R'^*(u)\psi$ can be written in PDE form as

$$\begin{cases} -\dfrac{\partial \psi^*(t)}{\partial t} - F'^*(\varphi(t))\psi^*(t) = \psi(t), \quad t \in (0,T), \\ \psi^*(T) = 0, \\ v = \psi^*(0), \end{cases} \tag{1.8}$$

where F'^* is adjoint to F'.

1.2. The role of the Hessian and its inverse. The Hessian (1.7) and its inverse play important roles in different aspects of DA. The first is as a coefficient matrix (and preconditioner) in incremental 4D-Var [8]. Here each step of an outer iterative Gauss–Newton process is of the form $u_{i+1} = u_i + \alpha_i \, \delta u_i$, with u_i a discrete approximation of the unknown initial state at iteration i, descent step α_i and update (descent direction) δu_i. As the update satisfies

$$\mathcal{H}(u_i)\delta u_i = -G(u_i), \tag{1.9}$$

where $G(u_i)$ is the gradient of the cost function, a system of linear equations involving $\mathcal{H}$ has to be solved at each step. Given a Hessian-vector product evaluation routine, the systems in (1.9) are usually solved iteratively using, for example, the conjugate gradient (CG) algorithm [23]. An approximation of $\mathcal{H}^{-1}$, if available at a reasonable

cost, can therefore be used to precondition equation (1.9) to accelerate convergence of this inner iteration. This is the application studied in our numerical experiments (see section 5.5).

In addition, $\mathcal{H}^{-1}$ is involved in several aspects of statistical postprocessing and characterization of the optimal solution. First, for linear and moderately nonlinear DA problems, $\mathcal{H}^{-1}$ can be used as an approximation of the analysis error covariance matrix [16, 36, 39]. For example, confidence intervals for the components of the analysis vector can be defined by the corresponding diagonal elements (variance) of $\mathcal{H}^{-1}(\bar{u})$. A column c_i of $\mathcal{H}^{-1}(\bar{u})$ which includes the ith diagonal element can be obtained by solving the equation $\mathcal{H}(\bar{u})c_i = e_i$ (where e_i is a Euclidean unit vector). If the number of requested diagonal elements is significant, it would be much less expensive to evaluate $\mathcal{H}^{-1}(\bar{u})$ once and keep it in some limited-memory form than to retrieve necessary diagonal elements using the Hessian-vector product rule. We note also that as the Hessian $\mathcal{H}(\bar{u})$ is equivalent to the Fisher information matrix (up to a constant multiplier), the diagonal elements of the inverse Hessian can also be used in the context of optimal experimental design involving such optimality criteria as l-optimality, for example. Second, the analysis probability density function (pdf) is defined by the analysis $\bar{u}$ and the analysis error covariance. Random functions from the Gaussian distribution $\mathcal{N}(\bar{u}, \mathcal{H}^{-1}(\bar{u}))$ can therefore be used as "particles" of the ensemble of initial states, which may be useful for ensemble forecasting [14, 40]. These functions can be generated using $u = \mathcal{H}^{-1/2}(\bar{u})\xi$, where $\xi \sim \mathcal{N}(0, I)$, or using the eigenvalues of $\mathcal{H}(\bar{u})$ [13]. However, in highly nonlinear cases the "particles" generated using $\mathcal{H}$ are unlikely to belong to the true posterior distribution, and thus one must solve perturbed DA problems. This approach is referred to as the *fully nonlinear ensemble method* [16] or *randomized maximum likelihood method* [6]. In these cases, an approximation of $\mathcal{H}^{-1/2}$ can be used for preconditioning the nonlinear minimization process to accelerate convergence, often with impressive results. Last, the analysis error δu and the data errors ξ_b and ξ_o are related via the approximate error equation

$$\mathcal{H}(\bar{u})\delta u = V_b^{-1}\xi_b + R'^{*}(\bar{u})C^{*}\xi_o$$

[18]. This equation can be considered as a meta-model for investigating the effects of non-Gaussian data errors on the analysis error pdf. Specifically, if the model depends on parameters $\theta \in \Theta$, where Θ is the parameter space, an important problem is to quantify the sensitivity of the analysis error to uncertainty ξ_θ in these parameters. This can be done using the relationship

$$\mathcal{H}(\bar{u})\delta u = R'^{*}(\bar{u})C^{*}V_o^{-1}CD(\bar{u})\xi_\theta,$$

where $D(\bar{u}) : \Theta \to Y$. Once again, $\mathcal{H}$ must be inverted to obtain δu.

The applications listed above all involve either solving multiple systems of linear equations involving $\mathcal{H}$ or having access to the inverse operator $\mathcal{H}^{-1}$. In practice, an explicit discrete representation of $\mathcal{H}$ is never required, since the Hessian-vector product can be obtained by successively applying operators in formula (1.7). The development of feasible methods for generation, storage, and subsequent use of $\mathcal{H}^{-1}$ or $\mathcal{H}^{-1/2}$ in this framework is not well understood: this is the prime motivation for our interest in the development of efficient algorithms for computing and managing the inverse Hessian such as those presented in this paper.

1.3. First-level preconditioning. In DA problems, it is common to transform the Hessian (1.7) to a new operator with a more favorable eigenvalue distribution

using so-called first-level preconditioning with $V_b^{1/2}$. This results in the preconditioned Hessian

$$H(u) = (V_b^{1/2})^* \mathcal{H}(u) V_b^{1/2} = I + (V_b^{1/2})^* R'^*(u) C^* V_o^{-1} C R'(u) V_b^{1/2}. \tag{1.10}$$

The action of applying $H(u)$ to a given function $v \in X$ is defined by the successive solutions of the following problems:

$$\begin{cases} \dfrac{\partial \psi(t)}{\partial t} = F'(\varphi(t,u))\psi(t), \\ \psi(0) = V_b^{1/2} v, \end{cases} \tag{1.11a}$$

$$\begin{cases} -\dfrac{\partial \psi^*(t)}{\partial t} - F'^*(\varphi(u,t))\psi^*(t) = -C^* V_o^{-1} C\psi(t), \quad t \in (0,T), \\ \psi^*(T) = 0, \end{cases} \tag{1.11b}$$

$$H(u)v = v - V_b^{1/2} \psi^*(0). \tag{1.11c}$$

It can be seen from (1.10) that all eigenvalues of $H(\bar{u})$ are greater than or equal to one (a detailed analysis of the conditioning of $H(\bar{u})$ can be found in [20]). Furthermore, it has been observed that for many practical DA problems, only a relatively small percentage of the eigenvalues are distinct enough from unity to contribute significantly to the Hessian. This suggests using *limited-memory* representations of the discrete Hessian, where this structure in the spectrum of H is exploited. Specifically, a few leading eigenvalue/eigenvector pairs $\{\lambda_i, w_i\}$ are computed (typically using the Lanczos method [9, 26] as H is available in vector-product form) and H^α is replaced by the approximation

$$H^\alpha(u) \simeq I + \sum_{i=1}^{n} (\lambda_i^\alpha - 1) w_i w_i^*, \tag{1.12}$$

where α is a real number (for example, $\alpha = -1$ or $\alpha = -1/2$), and the summation bound n is much smaller than the dimension of the discrete Hessian H.

Although first-level preconditioning can be very helpful, it may not be sufficient in certain circumstances. First, the number of eigenvalues of H which are essentially distinct from unity depends on the observation impact, which is proportional to the number of observations and their accuracy. Taking into account the size of state and observation vectors used in modern realistic DA applications (usually $10^9 - 10^{12}$ in length for state and $10^6 - 10^9$ for observation vectors), the value of m in (1.12) required to obtain a limited-memory approximation of reasonable quality may still be prohibitively large in terms of memory. Furthermore, in forecasting problems, any computational result has a lifespan, that is, a time period when this result remains usable. Given that each Lanczos iteration requires evaluating a Hessian-vector product (which involves running both the tangent linear and the adjoint models), the time needed for calculating even a small fraction of the spectrum could easily exceed the lifespan of the resulting approximation. We are therefore interested in gaining additional savings in memory and computing time over and above those afforded by approximation (1.12). For the remainder of the paper, we will consider only the projected Hessian $H(\bar{u})$ given by (1.10), that is, we assume that first-level preconditioning has already been applied. In what follows, we develop a new approximation to H^{-1}, based on a multilevel structure, which requires less memory or less computational time than (1.12).

2. Multilevel eigenvalue decomposition algorithm. In this section we describe an algorithm for constructing a multilevel approximation to the inverse (and its square root) of a general symmetric positive definite operator A associated with problem (1.1) (for example, $H(\bar{u})$ in (1.10)). Such approximations have many different practical uses, for example, as preconditioners for linear systems of equations, as representations of covariance matrices, or in error/confidence interval analysis.

The key idea can be summarized as follows. Consider a limited-memory approximation to A^{-1} of the form in (1.12) (with $\alpha = -1$). If we assume that A is only available in operator-vector product form, that is, we can evaluate Av for some discrete function v on the underlying computational grid, the eigenvalues required can be calculated using the Lanczos method. Given a sequence of nested grids, a conceptual outline of the recursive multilevel process is as follows:

1. Represent A on the coarsest grid level.
2. Use a local preconditioner to improve the eigenvalue distribution.
3. Build a limited-memory approximation to its inverse, which forms the basis of the local preconditioner at the next finer level.
4. Move up one grid level and repeat.

As proof of concept, we describe the ideas in a one-dimensional setting, with commensurate numerical examples in section 5. However, the concept is equally valid for two- and three-dimensional problems.

2.1. Multilevel grid structure. We consider the spatial domain $\Omega = [0, 1]$ and construct a sequence of grids for discretizing the DA problem described in section 1.1. We suppose that the base grid on which the problem is defined has a uniform distribution of $m_0 = m + 1$ grid points and use this as our finest grid (denoted by grid level $k = 0$). We form the next grid by removing a grid point from in between each pair of existing points, to give a grid at level $k = 1$ with $m_1 = m/2 + 1$ uniformly spaced points. Continuing this refinement leads to a sequence of grids at levels $k = 0, 1, 2, \ldots, k_c$ (where k_c is the coarsest grid level), with the grid at level k containing $m_k = m/2^k + 1$ grid points.

2.2. Grid transfer operators. We introduce the prolongation operator $S_{k,k-i}$, $0 \le i \le k$, which maps (interpolates) a discrete function v_k defined at grid level k to a finer grid level $k - i$ (or to the same grid for $i = 0$). That is, the operator satisfies

$$v_{k-i} = S_{k,k-i}\, v_k, \qquad S_{k,k} = I_k, \tag{2.1}$$

where I_k is the identity operator at grid level k. Similarly, the restriction operator $S^*_{k,k-i}$ maps a discrete function v_{k-i} defined at grid level $k - i$ to a coarser grid level k. That is,

$$\tilde{v}_k = S^*_{k,k-i}\, v_{k-i}, \qquad S^*_{k,k} = I_k, \tag{2.2}$$

where $S^*_{k,k-i}$ is the adjoint operator to $S_{k,k-i}$. Combining operators (2.1) and (2.2) gives $\tilde{v}_k = S^*_{k,k-i} S_{k,k-i}\, v_k$, so $\tilde{v}_k = v_k$ only if

$$S^*_{k,k-i} S_{k,k-i} = I_k. \tag{2.3}$$

In other words, for $\tilde{v}_k = v_k$ we require $S_{k,k-i}$ to be an orthonormal projection: we will refer to this situation as *perfect interpolation*.

To construct a multilevel representation of A^{-1} and $A^{-1/2}$, we will also need the projection of the operator A at a finer (or coarser) grid level. Let A_k be a discrete

representation of A defined at grid level k and suppose we want to find its projection at a finer grid level $k-i$ (we will denote this projection by $P_{k-i}(A_k)$). Given that a discrete function v_k at level k has been obtained by projecting the corresponding fine grid function v_{k-i}, $0 < i \le k$, using the grid transfer operators described above, we consider what happens when A_k is applied to this projection. Specifically, we decompose the fine grid function v_{k-i} into two parts and write $v_{k-i} = v_{k-i}^{(A)} + v_{k-i}^{(B)}$ with

$$v_{k-i}^{(A)} = (I_{k-i} - S_{k,k-i}S_{k,k-i}^*)v_{k-i}, \qquad v_{k-i}^{(B)} = S_{k,k-i}S_{k,k-i}^* v_{k-i}.$$

Projecting each component separately to grid level k gives a corresponding coarse grid function of the form $v_k = v_k^{(A)} + v_k^{(B)}$, where

$$\begin{aligned} v_k^{(A)} = S_{k,k-i}^* v_{k-i}^{(A)} &= S_{k,k-i}^*(I_{k-i} - S_{k,k-i}S_{k,k-i}^*)v_{k-i} \\ &= (S_{k,k-i}^* - (S_{k,k-i}^* S_{k,k-i})S_{k,k-i}^*)v_{k-i} \end{aligned} \tag{2.4}$$

and

$$v_k^{(B)} = S_{k,k-i}^* v_{k-i}^{(B)} = (S_{k,k-i}^* S_{k,k-i})S_{k,k-i}^* v_{k-i}.$$

If we now assume perfect interpolation, that is, $S_{k,k-i}$ satisfies (2.3), then (2.4) simplifies to $v_k^{(A)} = 0$. In other words, $v_{k-i}^{(A)}$ contains modes of v_{k-i} which are not supported on the coarser grid k and, therefore, should not be transferred to this level at all. However, the second component $v_{k-i}^{(B)}$ must be projected to grid level k, the operator A_k applied, and the result projected back to grid level $k-i$. Combining these two steps gives

$$\begin{aligned} P_{k-i}(A_k)v_{k-i} &= v_{k-i}^{(A)} + S_{k,k-i}A_k S_{k,k-i}^* v_{k-i}^{(B)} \\ &= (I_{k-i} - S_{k,k-i}S_{k,k-i}^*)v_{k-i} + S_{k,k-i}A_k S_{k,k-i}^* v_{k-i} \\ &= (S_{k,k-i}(A_k - I_k)S_{k,k-i}^* + I_{k-i})v_{k-i}. \end{aligned}$$

We therefore define the projection of A_k at a finer grid level $k-i$, $0 \le i \le k$, using

$$\begin{aligned} P_{k-i}(A_k) &= S_{k,k-i}(A_k - I_k)S_{k,k-i}^* + I_{k-i}, \qquad 0 < i \le k, \\ P_k(A_k) &= A_k. \end{aligned} \tag{2.5}$$

Note that we will also need the adjoint of $P_{k-i}(A_k)$. It is easy to see $P_{k-i}^*(A_k) = P_{k-i}(A_k^*)$.

In a similar way, we define the projection of an operator A_{k-i} at a coarser grid level $k, 0 \le i \le k_c$ using

$$\begin{aligned} Q_k(A_{k-i}) &= S_{k,k-i}^*(A_{k-i} - I_{k-i})S_{k,k-i} + I_k, \qquad 0 < i \le k_c, \\ Q_k(A_k) &= A_k, \end{aligned} \tag{2.6}$$

with corresponding adjoint $Q_k^*(A_{k-i}) = Q_k(A_{k-i}^*)$.

Finally, we note that in the case of perfect interpolation the operator $P_{k-i}(A_k)$ in (2.5) has the important property that $P_{k-i}(A_k) = P_{k-i}(A_k^{1/2})P_{k-i}((A_k^{1/2})^*)$. Also, the expression (2.6) simplifies to $Q_k(A_{k-i}) = S_{k,k-i}^* A_{k-i} S_{k,k-i}$. However, $Q_k(A_{k-i}) \ne Q_k(A_{k-i}^{1/2})Q_k((A_{k-i}^{1/2})^*)$.

2.3. Multilevel algorithm. We now develop an algorithm for constructing a multilevel limited-memory representation of A^{-1} (and $A^{-1/2}$) in operator-vector product form, that is, as $A^{-1}v$ or $A^{-1/2}v$. This is achieved by separating the eigensystem of operator A into the subsystems associated with different representation levels. We assume that the operator-vector product is available at the finest grid level $k = 0$, that is, we have available $A_0 v_0$ for some fine grid function v_0. In section 2.3.1 we describe our algorithm based on a sequence of coarser grids $k = 1, 2, \ldots, k_c$, where k_c is the coarsest level. One key ingredient of this algorithm is a sequence of local preconditioners applied at each grid level: these are described in detail in section 2.3.2.

2.3.1. Structure of the algorithm. We begin by representing the finest grid operator A_0 on level k as $Q_k(A_0)$ using (2.6), then precondition this to obtain

$$\tilde{Q}_k(A_0) = B^*_{k,k+1} Q_k(A_0) B_{k,k+1}. \tag{2.7}$$

The level k preconditioner $B_{k,k+1}$ will be chosen so that the eigenvalues of $\tilde{Q}_k(A_0)$ are closer to unity than those of $Q_k(A_0)$: details of how this is done follow in section 2.3.2. We then use the Lanczos method to compute a specified number, n_k, say, of the largest eigenvalues of $\tilde{Q}_k(A_0)$ (measured in a log-squared sense; see section 5.4), together with their associated eigenvectors. The resulting n_k eigenpairs $\{\lambda^i_k, U^i_k\}$, $i = 1, \ldots, n_k$, are then used to construct a limited-memory approximation to $\tilde{Q}_k(A_0)$, namely,

$$\hat{Q}_k(A_0) = I_k + \sum_{i=1}^{n_k} (\lambda^i_k - 1) U^i_k (U^i_k)^T. \tag{2.8}$$

Note that, as in (1.12), an approximation to $\tilde{Q}_k(A_0)$ raised to any chosen power α is readily available. This means that $\hat{Q}_k^{-1}(A_0)$, $\hat{Q}_k^{1/2}(A_0)$ and, most importantly for the preconditioners defined in the next section, $\hat{Q}_k^{-1/2}(A_0)$ are easily computed.

The accuracy of approximation (2.8) is clearly critically affected by the number of eigenvectors which are calculated and stored at each grid level. To facilitate later investigation of how these values should be chosen, we introduce the notation

$$N_e = (n_0, n_1, \ldots, n_{k_c}), \qquad \widehat{N_e} = \sum_{k=0}^{k_c} n_k, \tag{2.9}$$

for the vector containing these values for a particular approximation and the sum of its entries.

2.3.2. Level k preconditioners. The algorithm above involves a preconditioner $B_{k,k+1}$ for $Q_k(A_0)$ local to the current grid level. The motivation for our choice of $B_{k,k+1}$ is the assumption that $Q_{k+1}(A_0)$ is a good approximation to $Q_k(A_0)$, so we can use the projection of the former to grid level k to precondition the latter. That is, we expect that the eigenvalues of the preconditioned operator

$$P_k \left(Q_{k+1}^{-1/2}(A_0) \right) Q_k(A_0) P_k \left(Q_{k+1}^{-1/2}(A_0) \right)$$

to be clustered around 1. Furthermore, it can be seen from (2.7) and (2.8) that

$$Q_k^{-1}(A_0) = B_{k,k+1} \tilde{Q}_k^{-1}(A_0) B^*_{k,k+1} \approx B_{k,k+1} \hat{Q}_k^{-1}(A_0) B^*_{k,k+1} \tag{2.10a}$$

and so

$$Q_k^{-1/2}(A_0) = B_{k,k+1}\tilde{Q}_k^{-1/2}(A_0) \approx B_{k,k+1}\hat{Q}_k^{-1/2}(A_0). \tag{2.10b}$$

Note that with this notation, preconditioner $B_{k,k+1}$ is applied on level k, using information projected from level $k+1$.

The above considerations are valid for grid levels $k = 0, \ldots, k_c-1$. On the coarsest grid level $k = k_c$, grid level $k_c + 1$ does not exist, so we set $B_{k_c,k_c+1} = I_{k_c}$. Note also that on the finest grid level $k = 0$ we can use A_0 directly, that is, $Q_0(A_0) \equiv A_0$. In practice, moving from the coarsest to the finest grid, we accumulate the eigenpairs $\{\lambda_k, U_k\}$, $k = k_c, \ldots, 0$, in (2.8) which allows us to define the required products $A_0^{-1}v_0$ and $A_0^{-1/2}v_0$ via a recursive algorithm as follows. At a general grid level k, the preconditioner is constructed in a recursive way using information from the previous (coarser) grid levels via

$$B_{k,k+1} = \begin{cases} P_k(B_{k+1,k+2}\hat{Q}_{k+1}^{-1/2}(A_0)), & k = 0, 1, \ldots, k_c - 1; \\ I_{k_c}, & k = k_c; \end{cases} \tag{2.11a}$$

$$B^*_{k,k+1} = \begin{cases} P_k\left(\hat{Q}_{k+1}^{-1/2}(A_0)B^*_{k+1,k+2}\right), & k = 0, 1, \ldots, k_c - 1; \\ I_{k_c}, & k = k_c; \end{cases} \tag{2.11b}$$

where

$$\hat{Q}_k^{-1/2}(A_0) = I_k + \sum_{i=1}^{n_k}\left((\lambda_k^i)^{-1/2} - 1\right)U_k^i(U_k^i)^*$$

(cf. (2.8)).

2.3.3. Summary. Using the above definitions, an operator representing the multilevel eigenvalue decomposition algorithm can be constructed as follows.

Algorithm 2.1 Multilevel eigenvalue decomposition algorithm.

$[\Lambda_0, \mathcal{U}_0]$= *MLEVD*$(A_0, N_e)$

for $k = k_c, k_c - 1, \ldots, 0$
  compute by the Lanczos method and store in memory:
  $\{\lambda_k^i, U_k^i\}$, $i = 1, \ldots, n_k$ *of* $\tilde{Q}_k(A_0)$ *in* (2.7)
  using $B_{k,k+1}$ *and* $B^*_{k,k+1}$ *from* (2.11)
end

The input to this algorithm is A_0 (available in the form of an operator-vector product A_0v_0 at the finest level) together with the vector N_e in (2.9) containing the number of eigenpairs to be calculated at each grid level. At a given level k ($\neq k_c$), the algorithm uses the eigenpairs obtained at the previous (coarser) levels. The final output is a pair of vectors $[\Lambda_0, \mathcal{U}_0]$ containing the multilevel eigenstructure of A_0, which can be represented as follows:

$$\begin{aligned} \Lambda_0 &= \left[\lambda_{k_c}^1, \ldots, \lambda_{k_c}^{n_{k_c}}, \lambda_{k_c-1}^1, \ldots, \lambda_{k_c-1}^{n_{k_c-1}}, \ldots, \lambda_0^1, \ldots, \lambda_0^{n_0}\right], \\ \mathcal{U}_0 &= \left[U_{k_c}^1, \ldots, U_{k_c}^{n_{k_c}}, U_{k_c-1}^1, \ldots, U_{k_c-1}^{n_{k_c-1}}, \ldots, U_0^1, \ldots, U_0^{n_0}\right]. \end{aligned} \tag{2.12}$$

Given $[\Lambda_0, \mathcal{U}_0]$, for any function v_k the products $Q_k^{-1}(A_0)v_k$ and $Q_k^{-1/2}(A_0)v_k$ can be recovered using (2.10), which in turn involves (2.11). In particular, for any fine grid

function v_0 we can evaluate

$$A_0^{-1}v_0 \approx Q_0^{-1}(A_0) = B_{0,1}\hat{Q}_0^{-1}(A_0)B_{0,1}^* v_0, \tag{2.13a}$$

$$A_0^{-1/2}v_0 \approx Q_0^{-1/2}(A_0) = B_{0,1}\hat{Q}_0^{-1/2}(A_0)v_0, \tag{2.13b}$$

where $B_{0,1}$ and $B_{0,1}^*$ are defined in (2.11).

Using the notation in (2.9), we see that the vector Λ_0 in (2.12) contains a total of $\widehat{N_e}$ entries. As the grid at level k contains $m_k = m/2^k + 1$ points, the vector $\mathcal{U}_0$ has a total of

$$\sum_{k=0}^{k_c} n_k m_k = \left(\sum_{k=0}^{k_c} \frac{n_k}{2^k}\right) m + \widehat{N_e}$$

entries. In what follows, we use the term "memory ratio" for the quantity

$$r = \sum_{k=0}^{k_c} \frac{n_k}{2^k} \tag{2.14}$$

as this gives a useful estimate of the ratio of the amount of storage required to m (where the finest grid has $m+1$ points). An investigation of this storage requirement in practice for various eigenvalue combinations N_e is given in section 5.4.

We conclude this section by noting that Algorithm 2.1 can be generalized by using $\tilde{Q}_k(A_{k-i})$ instead of $\tilde{Q}_k(A_0)$ in (2.7), where A_{k-i}, $1 \le i \le k$, is a direct representation of A_0 on grid level k. In particular, for the case of interest here (with $A_0 \equiv H(u)$), we can define H directly on a given level k using

$$H_k(u_k) = I_k + Q_k((V_b^{1/2})^*)R_k'^*(u_k)Q_k(C^*V_o^{-1}C)R_k'(u_k)Q_k(V_b^{1/2}), \tag{2.15}$$

where $R_k(u_k)$ and $R_k^*(u_k)$ are the tangent linear (1.4) and adjoint (1.6) operators discretized on level k. Note that, as $V_b^{1/2}$ and $C^*V_o^{-1}C$ are defined at the finest grid level, appropriate projections are still required in (2.15). This approach allows the PDE problems defining the Hessian-vector product to be solved at any level $k-i$, while solving the eigenproblem at level k. This may help to reduce computational time, although the multilevel approximation will become less accurate (given the same allocated memory). We have not included an investigation of this approach in this paper.

2.3.4. Example of constructing a limited-memory inverse. For any symmetric positive definite operator A, the MLEVD algorithm in section 2.3.3 can be applied to obtain a multilevel limited-memory approximation of A^{-1} or $A^{-1/2}$ (as in (2.13)). This approach can be particularly useful in the case where A is a matrix representing a discretized differential operator, especially when the matrix A is defined as a matrix-vector product and solving multiple systems of equations involving A is expected. In this section, we illustrate the usefulness of the technique by considering the particular example of approximating the inverse of a covariance matrix.

Consider a random vector v of length m, resulting from the discretization of a continuous function of one variable, and a multivariate Gaussian distribution $\mathcal{N}(\bar{v}, V)$. Here $\bar{v}$ is the mean vector and V is the covariance matrix which, due to the high dimension of v, we assume is available in the matrix-vector product form Vv. An example of such a product defined via the solution of the heat conduction equation

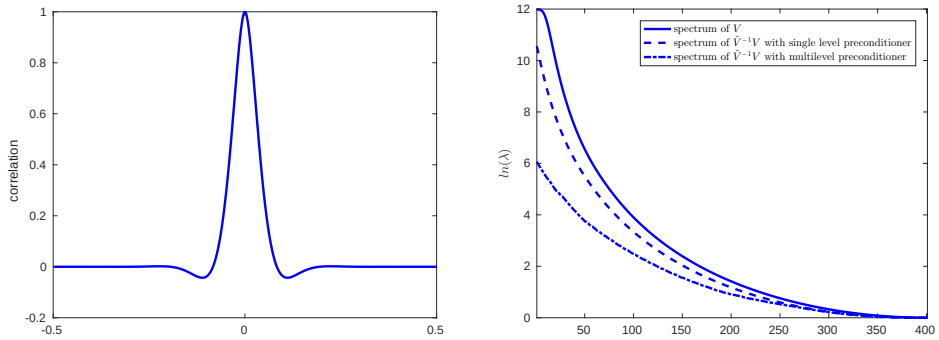

FIG. 1. *The correlation function for V (left) and the eigenvalues of V (solid line), and $\tilde{V}^{-1}V$ with single-level (dotted line) and multilevel (dot-dash line) preconditioners (right).*

can be seen in [31]. Checking the statistical hypothesis that a vector v_k from an observed sample belongs to $\mathcal{N}(\bar{v}, V)$ relies on computing the Mahalanobis distance

$$D_M = (v_k - \bar{v})^T V^{-1}(v_k - \bar{v}).$$

As V^{-1} is not readily available, obtaining $u = V^{-1}(v_k - \bar{v})$ is usually done by solving the system of linear equations $Vu = v_k - \bar{v}$ using an iterative method. This system itself needs a good preconditioner for V in order to accelerate iterative convergence, that is, we require a matrix $\tilde{V}^{-1}$ such that the eigenvalue spectrum of $\tilde{V}^{-1}V$ facilitates faster convergence, by either clustering the eigenvalues or reducing the condition number. In the numerical results below, we use both of these measures to evaluate preconditioner performance.

In this example, V is a 401×401 covariance matrix with unit diagonal, characterized by the uniform correlation function $\tilde{V}(i, i-j) = \tilde{V}(i-j)$ (shown on the left of Figure 1). The eigenvalues of V (plotted on a natural log scale) are represented by the solid line in the plot on the right of Figure 1. We compare them with the preconditioned spectrum of $\tilde{V}^{-1}V$ for two different preconditioners, both of which require the same amount of storage, as characterized by fixing the memory ratio r in (2.14) (we use $r = 16$ here). For the first preconditioner, all of the memory is deployed on the coarsest grid (with $m = 401$ grid points): this corresponds to a single-level degenerate case of the MLEVD approximation with $N_e = (r, 0, \ldots, 0)$. The resulting preconditioned spectrum is represented by the dashed line in the right plot in Figure 1. In the second case, we use four discretization levels for the MLEVD algorithm, with $N_e = (4, 8, 16, 32)$, showing the resulting spectrum as the dash-dot line in the same plot. It is clear that this multilevel preconditioner outperforms the classical (one-level) spectral preconditioner here in terms of clustering the eigenvalues. Further results for different values of r and N_e are summarized in Table 1. As well as the condition number of $\tilde{V}^{-1}V$ (which should be compared to the unpreconditioned value of $1.6e + 5$), the normalized Riemann distance D between V and $\tilde{V}$ is also tabulated (this is based on a comparison between the log-squared eigenvalues of the matrices; see section 5.4). In each case, a slash is used to distinguish values obtained using the multilevel and single-level approaches. Again, the multilevel MLEVD-based preconditioner is more effective.

3. Hessian decomposition. In the remainder of the paper, we focus on a specific application for the MLEVD algorithm, namely, the approximation of the inverse

TABLE 1
Comparison of multilevel and single-level preconditioners for various values of memory ratio r and eigenvalue combinations N_e.

Memory ratio r	N_e	Distance D	Condition number
4	(1,2,4,8) / (4,0,0,0)	0.84 / 0.96	5.48e+4 / 1.59e+5
8	(2,4,8,16) / (8,0,0,0)	0.71 / 0.92	5.82e+3 / 1.41e+5
12	(3,6,12,24) / (12,0,0,0)	0.61 / 0.86	1.28e+3 / 7.68e+4
16	(4,8,16,32) / (16,0,0,0)	0.54 / 0.81	4.31e+2 / 3.84e+4

Hessian matrix (1.10) in variational DA. Before we describe the specific algorithms proposed, we introduce a decomposition of the Hessian into a set of elementary Hessians which will prove to be useful later.

Using the factorization

$$V_o^{-1} = V_o^{-1/2}\,\bar{I}\,V_o^{-1/2},$$

where $\bar{I}$ is the identity on the M-dimensional space Y_{obs} and $V_o^{-1/2} : Y_{obs} \to Y_{obs}$ is the spectral square-root of V_o, the preconditioned Hessian in (1.10) can be rewritten in the form

$$H(u) = I + \left(V_b^{1/2}\right)^* R'^*(u)C^*V_o^{-1/2}\,\bar{I}\,V_o^{-1/2}CR'(u)V_b^{1/2}. \tag{3.1}$$

Now let $\mathcal{I}$ be a set of indices of the diagonal elements of $\bar{I}$, and suppose that $\mathcal{I}$ is partitioned into L disjoint subsets $\mathcal{I}^l$, $l = 1, \ldots, L$. Defining the diagonal matrices $\bar{I}^l$ such that

$$\bar{I}^l_{i,i} = \begin{cases} 1, & i \in \mathcal{I}^l, \\ 0, & i \notin \mathcal{I}^l, \end{cases} \quad i = 1, \ldots, M,$$

the identity can be written as

$$\bar{I} = \sum_{l=1}^{L} \bar{I}^l.$$

Combining this with (3.1), after some algebraic manipulation we obtain a Hessian decomposition as follows:

$$H(u) = I + \sum_{l=1}^{L} (H^l(u) - I), \tag{3.2}$$

where

$$H^l(u) = I + \left(V_b^{1/2}\right)^* R'^*(u)C^*V_o^{-1/2}\,\bar{I}^l\,V_o^{-1/2}CR'(u)V_b^{1/2}. \tag{3.3}$$

For a specific partition of $\mathcal{I}$, each elementary Hessian $H^l(u)$ can be presented in the limited-memory form (1.12) with the number of leading eigenpairs n^l required for its accurate representation less than n, since

$$\sum_{l=1}^{L} n^l \approx n.$$

In particular, it is possible to define a partition of $\mathcal{I}^l$ such that $n^l \approx n/L$, with the number of Lanczos iterations and, correspondingly, the amount of CPU time needed

for computing the eigenpairs of a single H^l proportionally reduced. We also note that the elementary Hessians can be computed in parallel, so that the amount of CPU time required for computing the full limited-memory Hessian should not exceed the largest time spent on computing a single H^l.

To define an optimal partition, we may use the fact that the influence of the observations made by sensors located within a given spatial subdomain is also spatially localized. For example, suppose we have a set of S sensors and let $\underline{x}(t) = (x_1(t), x_2(t), \ldots, x_S(t))^T$, where $x_s(t) \in \Omega$ represents the position of sensor s at time t. Note that if $x_s(t) = x_s(0)$ for all $t \in [0, T]$, then sensor s is stationary. Assuming that observations are recorded with a certain time period, we introduce an observation operator $C : Y \to Y_{obs}$ such that

$$C\varphi(t, x) = \varphi(t_j, x_s(t_j)), s = 1, \ldots, S,\ j = 1, \ldots, N,$$

where N is the number of observation nodes in time. The adjoint to C, $C^* : Y_{obs} \to Y$, is given by

$$C^*\varphi(t_j, x_s(t_j)) = \sum_{s=1}^{S}\sum_{j=1}^{N} \varphi(t_j, x_s(t_j))\delta(x - x_s(t_j))\delta(t - t_j),$$

where δ is the Dirac delta function. We also assume that the observation covariance matrix V_o is block-diagonal with S diagonal blocks of dimension N. We now divide the spatial domain Ω into a set of disjoint subdomains Ω^l, $l = 1, \ldots L$, and define $\mathcal{I}^l$ to be the set of indices $i = (s-1)N + j$ such that $x_s(t_j) \in \Omega^l$, that is,

$$\mathcal{I}^l = \{i \in \mathbb{N}, i = (s-1)N + j : x_s(t_j) \in \Omega^l\}.$$

As long as the subdomains Ω^l are nonoverlapping and cover the whole domain, decomposition (3.2)–(3.3) is defined for this partition. We will use this Hessian decomposition in section 4 as part of a preconditioner for inner iterations of a Gauss–Newton method. In this context, we refer to an elementary Hessian H^l as a *local* Hessian.

For a Hessian defined directly at a given level k (as per (2.15)), it is easy to see that the local Hessian decomposition is

$$H_k(u_k) = I_k + \sum_{l=1}^{L}(H_k^l(u_k) - I_k), \tag{3.4}$$

where

$$H_k^l(u_k) = I_k + Q_k\left((V_b^{1/2})^*\right) R_k'^*(u_k)Q_k(C^*V_o^{-1/2}\,\bar{I}^l\,V_o^{-1/2}C)R_k'(u_k)Q_k(V_b^{1/2}). \tag{3.5}$$

For a specific partition of $\mathcal{I}$, powers of each elementary Hessian $H_k^l(u_k)$ can be approximated in limited-memory form as

$$(\hat{H}_k^l)^\alpha(u_k) \simeq I_k + \sum_{i=1}^{n_k^l}((\lambda_{k,i}^l)^\alpha - 1)U_{k,i}^l(U_{k,i}^l)^* \tag{3.6}$$

using the leading n_k^l eigenpairs $\{\lambda_{k,i}^l, U_{k,i}^l\}$, $i = 1, \ldots, n_k^l$ (cf. (1.12)), where the length of each eigenvector $U_{k,i}^l$ is equal to m_k.

The calculation of a single local Hessian H_k^l in (3.5) is a relatively inexpensive task which requires much less computational time than computing the global Hessian

because each local Hessian deviates from the identity operator I only within the area of influence surrounding the subdomain Ω^l, with the size of this area depending on the transport mechanisms supported by the dynamical model. The number of eigenpairs used to describe this deviation could be relatively small; therefore a smaller number of Lanczos iterations may also be needed to evaluate the leading eigenvalues of H_k^l, with the corresponding eigenvectors being different from zero only within the local area of influence. Thus, a compact storage scheme for the eigenvectors can be utilized. To make further computational savings, the local Hessians H_k^l can be computed at discretization level $k = k' > 0$, as opposed to on the finest grid $k = 0$. Finally, if the local Hessians are computed in parallel, then a very significant reduction in computing time can be achieved. Specifically, if the eigenvalue analysis for each H_k^l can be carried out on an individual processor, the time needed for computing all L eigenpair sets will be reduced to the maximum time taken to compute the eigenpairs of an individual local Hessian. We also observe that, for computing H_k^l, a local area model rather than the global model has to be run, and it is also possible that only some sensors from the whole observation array, or from only some areas of the computational domain, will be of interest, so H_k^l need not be calculated for every Ω^l: these ideas are not investigated further in this paper.

4. Approximating the inverse Hessian. In this section we use the multilevel eigenvalue decomposition in Algorithm 2.1 to build various approximations to the inverse Hessian H^{-1}, where H is defined in (1.10). We describe three different possible algorithms, which may be useful depending on the constraints in place in terms of available computing time and memory. Some numerical experiments illustrating their relative accuracy and usefulness in practice are given in section 5.4.

4.1. Algorithm 4.1. This involves using a straightforward application of the multilevel decomposition in Algorithm 2.1 to H_0, resulting in an eigenstructure $[\Lambda_0, \mathcal{U}_0]$ which can be used to evaluate $H_0^{-1}v_0$ and $H_0^{-1/2}v_0$ via (2.13). For a given optimal solution $\bar{u}_0$, the Hessian-vector product H_0v_0 is defined by (1.11) discretized at the finest grid level $k = 0$. Appropriate values for the parameters in N_e are discussed in section 5. The algorithm can be outlined as follows.

Algorithm 4.1

define N_e
compute $[\Lambda_0, U_0] = MLEVD(H_0, N_e)$

4.2. Algorithm 4.2. In this approach, we assume that memory restrictions are not important but that computing time is limited. To this end we utilize the Hessian decomposition idea as described in section 3. This requires the choice of L further parameters, n_k^l, $l = 1, \ldots, L$, which determine the number of eigenpairs used in $\hat{H}_k^l$ (the limited-memory approximation to H_k^l in (3.5)). The algorithm can be outlined as follows.

Algorithm 4.2

define N_e, *level* $k' \geq 0$, *partition* $\mathcal{I}^l$, $n_{k'}^l$, $l = 1, \ldots, L$
for $l = 1, \ldots, L$:
  compute $\{\lambda_{k'}^i, U_{k'}^i\}^l$, $i = 1, \ldots, n_{k'}^l$ of $H_{k'}^l$ *in* (3.5)
compute $[\Lambda_0, U_0] = MLEVD(P_0(\hat{H}_{k'}), N_e)$ *where* $\hat{H}_{k'}$ *is in form* (3.4) *based on* $\hat{H}_{k'}^l$ *from* (3.6)

4.3. Algorithm 4.3. Although Algorithm 4.2 (in parallel form) requires less computational time than Algorithm 4.1, it requires more storage. In this third approach, we aim to alleviate this by reducing the memory requirements of Algorithm 4.2. We do that by applying MLEVD to the local *inverse* Hessians to obtain a reduced memory representation of the local Hessians before they are used in (3.4). The number of eigenpairs used in these additional approximations must now be specified: at level k, we denote these using a vector N_k^l with an entry for the number of eigenvalues used on each (local) level, in an analogous way to (2.9) as

$$N_k^l = (n_k^l, n_{k+1}^l, \ldots, n_{k_c}^l).$$

The algorithm can be outlined as follows.

Algorithm 4.3

define N_e, *level* k', *partition* $\mathcal{I}^l$, $n_{k'}^l$, $N_{k'}^l$, $l = 1, \ldots, L$
for $l = 1, \ldots, L$:
  compute $\{\lambda_{k'}^i, U_{k'}^i\}^l$, $i = 1, \ldots, n_{k'}^l$ *of* $H_{k'}^l$ *in* (3.5)
  compute $[\Lambda_{k'}, U_{k'}]^l = \mathit{MLEVD}((\hat{H}_{k'}^l)^{-1}, N_{k'}^l)$, *where* $(\hat{H}_{k'}^l)^{-1}$ *is the limited-memory inverse* (1.12) *of* $H_{k'}^l$ *based on* $\{\lambda_{k'}^i, U_{k'}^i\}^l$
compute $[\Lambda_0, U_0] = \mathit{MLEVD}(P_0(\hat{H}_{k'})), N_e)$ *where* $\hat{H}_{k'}$ *is in form* (3.4) *based on multilvel approximations to* $\hat{H}_{k'}^l$ *as defined by* $\{\Lambda_{k'}, U_{k'}\}^l$ *via* (2.10a)

Note that the last step involves use of the recursive functions from (2.11), with recursion starting from level k'.

5. Numerical experiments. In this section we provide some implementation details and report the results of some numerical experiments which illustrate the performance of the three algorithms presented in the previous section. We begin by describing the model problems used in the rest of the section.

5.1. Model problems. As a test problem, we use as a model a one-dimensional Burgers' equation with a nonlinear viscous term. Specifically, we consider

$$\frac{\partial \varphi}{\partial t} + \frac{1}{2}\frac{\partial}{\partial x}(\varphi^2) = \frac{\partial}{\partial t}\left(\mu(\varphi)\frac{\partial \varphi}{\partial x}\right), \qquad \varphi = \varphi(x,t), \quad t \in (0,T), \quad x \in (0,1), \tag{5.1}$$

with the Neumann boundary conditions $\frac{\partial}{\partial x}(\varphi(0,t)) = \frac{\partial}{\partial x}(\varphi(1,t)) = 0$ and viscosity coefficient

$$\mu(\varphi) = \mu_0 + \mu_1 \left(\frac{\partial \varphi}{\partial x}\right)^2,$$

where μ_0 and μ_1 are positive constants (in all computations we use $\mu_0 = 10^{-4}$ and $\mu_1 = 10^{-5}$). The corresponding analytic expressions for the tangent linear and adjoint models are presented in Appendix A. Burgers' equation is often considered for testing DA algorithms as a simple model describing elements of atmospheric flow motion [15].

We use an implicit time discretization and the power law first-order finite volume scheme for spatial discretization [35]. At each time step, we perform nonlinear iterations to converge on nonlinear coefficients. The tangent linear (1.3) and adjoint (1.8) models are obtained by using the automatic differentiation engine Tapenade [22], applied to the Fortran code implementing the above presented model.

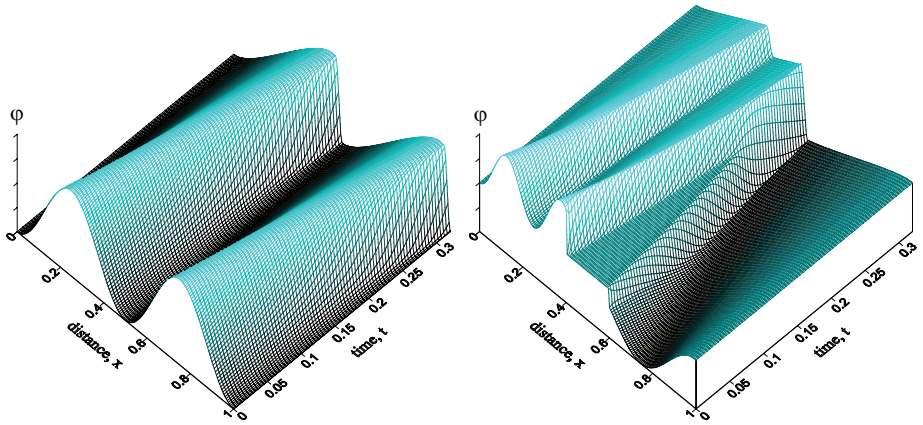

FIG. 2. *Flow evolution for initial states* (5.2): u^1_{true} *(left) and* u^2_{true} *(right).*

TABLE 2
Details of the four model problems studied.

Model problem	Observation scheme	Initial condition
MP1	A	u^1_{true}
MP2	A	u^2_{true}
MP3	B	u^1_{true}
MP4	B	u^2_{true}

We use two different initial states, defined by

$$u^1_{true}(x) = 0.1 + 0.35\left[1 + \sin\left(4\pi x + \frac{3\pi}{2}\right)\right], \qquad 0 < x < 1, \tag{5.2a}$$

$$u^2_{true}(x) = \begin{cases} 0.5[1 - \cos(8\pi x)], & 0 < x \le 0.4, \\ 0.5[\cos(4\pi(x-1)) - 1], & 0.6 \le x < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{5.2b}$$

The corresponding flow evolutions are presented in Figure 2.

We also consider two different configurations of the sensors which provide the observations to be assimilated. For this one-dimensional problem, scheme A has seven stationary sensors, fixed at points 0.3, 0.4, 0.45, 0.5, 0.55, 0.6, and 0.7 in $[0, 1]$. In scheme B, there is one moving sensor which traverses the domain $[0, 1]$ from left to right twice during the observation time (see below for implementation details). This is to emulate satellite observations. The combinations used in our four model problems are detailed in Table 2.

5.2. Miscellaneous implementation details. The multilevel structure comprises four grids, with 401 grid points on the finest grid level ($k = 0$) and 51 points on the coarsest grid level ($k = 3$). We use cubic splines to implement the prolongation operator $S_{k,k-i}$ in (2.1), with the subroutine for its adjoint operator $S^*_{k,k-i}$ (restriction) again obtained by using automatic differentiation. For test cases MP1 and MP2 (stationary sensors positioned as listed above), each local Hessian H^l corresponds to one sensor located at $x = x^l$. For test cases MP3 and MP4 (moving

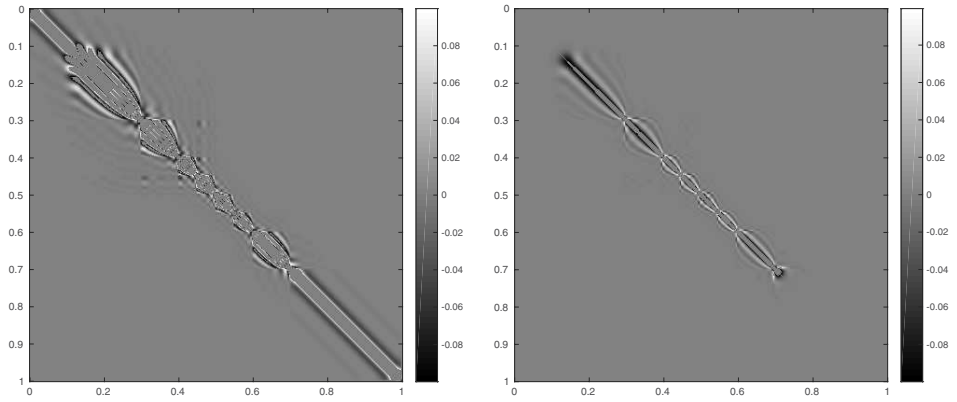

FIG. 3. *Correlation matrix for MP*1 *before (left) and after (right) first-level preconditioning has been applied. Unit diagonal entries have been omitted.*

sensor), the spatial domain $\Omega = (0, 1)$ is divided into six subdomains confined by points (0, 0.2, 0.4, 0.5, 0.6, 0.8, 1.0) and each local Hessian takes into account only measurements inside subdomain Ω_l.

The background error covariance matrix V_b is defined under the assumption that the background error belongs to the Sobolev space $W_2^2[0, 1]$ (see [17, section 5.1]). For each sensor configuration, the background error standard deviation (sd) can be seen later in Figure 4, where the correlation radius is uniform in space and equal to 0.2. The observation error is Gaussian and uncorrelated and has sd $\sigma_o = 0.016$. Thus, V_o is diagonal with uniform variance $(V_o)_{i,i} = \sigma_o^2$.

The eigenvalue problems are solved using standard ARPACK [27] to implement the implicitly restarted Lanczos method. We note that although calculating the eigenvalues of $\tilde{Q}_k(A_0)$ by the Lanczos method has to be performed in double precision, single precision storage of the results would be sufficient in practice because all eigenpairs calculated at level k are involved in preconditioning of $\tilde{Q}_{k-1}(A_0)$, so round-off errors introduced at level k are largely compensated for at level $k - 1$.

5.3. Preliminary results. In what follows, we interpret the inverse Hessian as a posterior covariance matrix. By the sd, here we mean a vector of the square roots of the diagonal elements of the inverse Hessian (variance). The correlation matrix $\bar{V}$ is the inverse Hessian matrix symmetrically scaled to have unit diagonal. That is, it has entries

$$\bar{V}_{ij} = \frac{\mathcal{H}_{ij}^{-1}}{\mathcal{H}_{ii}^{-1/2}\mathcal{H}_{jj}^{-1/2}}.$$

Figure 3 shows a plot of the correlation matrix for MP1 before (left) and after (right) first-level preconditioning is applied. Note that the (unit) diagonal entries have been omitted here to show the detail of the matrix structure. Although first-level preconditioning reduces a lot of the off-diagonal entries, what remains still deviates significantly from the identity, so there is scope for further preconditioning.

Figure 4 shows the sd (as defined above), plotted as a function of x on [0,1]. The top plots correspond to MP1 (stationary sensors) and the bottom plots to MP3 (moving sensor). In the left plots, the background sd is shown (labeled bg), along with the analysis sd based on $\mathcal{H}_k$, $k = 0, \ldots, 4$. The sd which corresponds to $k = 0$ ($\mathcal{H}_0$ is the Hessian (1.7) defined on the finest discretization level) can be used as a

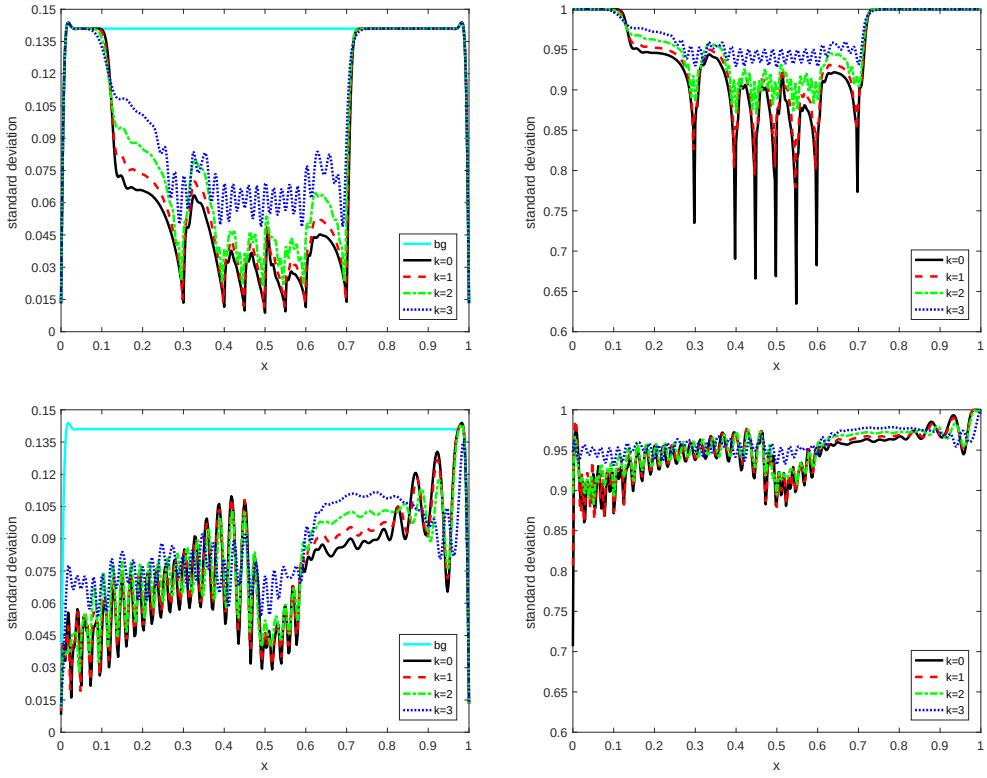

FIG. 4. *Standard deviation before (left) and after (right) first-level preconditioning is applied. The top plots correspond to MP*1 *(stationary sensors) and the bottom plots to MP*3 *(moving sensor).*

reference function value for comparing with sd estimates based on $\mathcal{H}_k$, $k > 0$ (defined in an analogous way to H_k in (2.15)). To make a meaningful comparison, we consider $\mathcal{H}_k$ projected to the finest representation level, that is, $P_0(\mathcal{H}_k)$. The right plots in Figure 4 correspond to the equivalent sd values where first-level preconditioning has been applied.

These results substantiate the key idea of our approach, namely, that the inverse Hessian at level k projected to level $k-1$ could be a good preconditioner for the Hessian at level $k-1$. Also, comparing the top and bottom subplots, an important difference between the impact made by stationary and moving sensors is apparent. A stationary sensor injects all information at one point; then the information is propagated over the spatial domain solely by the adjoint model. As a result, the variance at sensor location point tends toward the observation error variance σ_o^2. In contrast, a moving sensor injects a single piece of information at the current location point, which does not result in a sharp reduction of the local variance, and then it moves to another point, thus spreading information throughout the domain. In this case the distribution of information is less dependent on the model transport, which itself depends on the discretization of operators $R'(\bar{u})$ and $R'^*(\bar{u})$. Thus, the coarse grid approximations to the analysis variance should be better than in case of stationary sensors, as can be observed in Figure 4. Furthermore, the difference in the moving sensor case has a low-frequency nature, which can be accommodated by a relatively low number of

eigenpairs. This explains why our approach is more efficient for an observation scheme involving moving sensors, which will be apparent in the numerical results presented in section 5.5.

5.4. Investigating approximation accuracy. In the first set of experiments, we apply Algorithm 4.1 described in section 4.1 to $H(\equiv H_0)$ and use the resulting multilevel eigenvalue decomposition (2.12) to build a low-memory approximation to H^{-1} in recursive form (2.13), which we will denote by $\tilde{H}^{-1}$. To assess the accuracy of our approximations, we measure the difference between two matrices in terms of the Riemann distance. That is, for two symmetric positive definite $n \times n$ matrices A and B, we define

$$\delta(A,B) = \left\| ln(B^{-1}A) \right\|_F = \left(\sum_{i=1}^{n} ln^2 \lambda_i \right)^{1/2}, \tag{5.3}$$

where λ_i, $i = 1, \ldots, n$, are the eigenvalues of $B^{-1}A$ (see, for example, [32]). The Riemann distance can be considered as a symmetric measure of the difference between two Gaussian probability distributions having equal modes. Since the inverse Hessian is an approximation of the analysis error covariance matrix, it is very natural to use this measure in the current context. We will compare matrices after we have applied the first-level preconditioning described in section 1.3; it is easily shown that the Riemann distance remains unchanged on applying symmetric preconditioning to A and B, so the distance between two symmetric positive definite matrices in the original and projected spaces is the same.

The accuracy of a given approximation clearly will be critically dependent on the number of eigenvalues calculated at each grid level (that is, the choice of $N_e = (n_0, n_1, n_2, n_3)$). Ideally, we would like to be able to identify the "optimal" combination N_e for a given problem but this is nontrivial, particularly as the definition of optimality is itself dependent on the problem constraints. For example, one may want the best approximation which can be stored given a set amount of memory, or perhaps the approximation which ensures the biggest reduction in CPU time for computing $\tilde{H}^{-1}v$. Here we will focus on the former approach and assume that there is a fixed memory ratio r (see (2.14)) allowed for a particular problem. Within this fixed-memory framework, we measure the normalized Riemann distance

$$D = \frac{\delta(H^{-1}, \tilde{H}^{-1})}{\delta(H^{-1}, I)}, \tag{5.4}$$

where I is an appropriately sized identity matrix. By evaluating D for approximations constructed using all possible eigenvalue combinations N_e for a given memory ratio r, we can identify the best combinations N_e for this fixed memory problem.

Let us consider case MP1. If $\tilde{H}^{-1}$ is constructed from 64 eigenpairs of H^{-1} (corresponding to memory ratio $r = 64$), a good approximation with $D = 2.98e-4$ is achieved. However, if available memory is restricted to $r = 8$, so only 8 eigenpairs at level 0 can be stored, the resulting approximation is very poor ($D = 7.71e-1$). However, if $\tilde{H}^{-1}$ is constructed using the multilevel eigenvalue decomposition, much better approximations can be achieved. The eigenvalues of H^{-1} and $\tilde{H}^{-1}$ for MP1 with two representative eigenvalue combinations for $r = 8$ are illustrated in Figure 5. In each subplot, the first 70 eigenvalues of H^{-1} are plotted as blue circles (the rest of the eigenvalues are close to one and have not been plotted). The corresponding eigenvalues of $\tilde{H}^{-1}$ for two different N_e are plotted as red crosses. The values of N_e

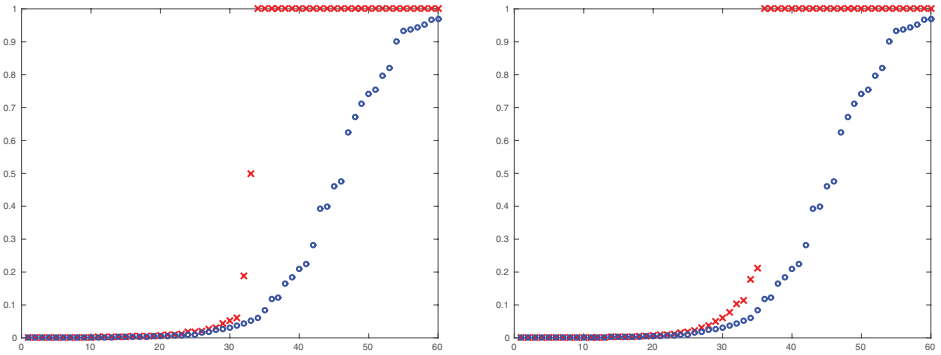

FIG. 5. *A comparison of eigenvalues of* H^{-1} *(blue circles) and* $\tilde{H}^{-1}$ *(red crosses) for two different eigenvalue combinations for MP*1*. Left:* $N_e = (0, 6, 13, 14)$*,* $D = 3.95e - 1$*. Right:* $N_e = (0, 0, 29, 6)$*,* $D = 3.39e - 1$*.*

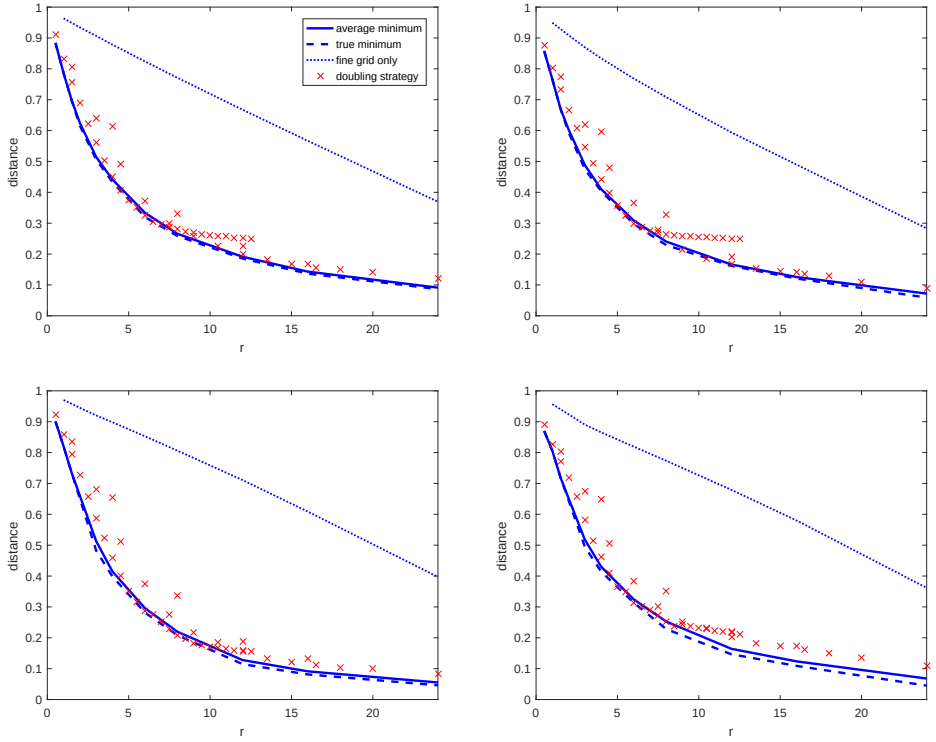

FIG. 6. *Distance* D (5.4) *plotted against memory ratio* r *for model problems MP*1 *(top left), MP*2 *(top right), MP*3 *(bottom left), and MP*4 *(bottom right).*

used have been chosen to represent typical behavior: we observe that even when no information is retained at the finest grid levels, the overall shape of the eigenvalue distribution of H^{-1} is still captured by the multilevel approximation.

As stated above, it is difficult to characterize a single "best possible" eigenvalue combination. However, certain traits can be identified. Plots of the minimum distance achieved for various memory ratios r are shown in Figure 6 for our four different test

problems. In each subplot, the dashed line shows the minimum distance achieved for a given r, and the solid line shows the average over the 5% of eigenvalue combinations satisfying (2.14) which resulted in the smallest distances. The dotted line shows the equivalent distance achieved using only fine grid vectors. The key observation here is that when there is only room to store a small number of fine grid vectors in memory, using a multilevel approximation clearly gives much better accuracy. Although we did not identify an "optimal" way of choosing N_e, our experiments did show that eigenvalue combinations which have no or very few eigenpairs on the finest level(s) appear to perform best. Based on this, we suggest the ansatz of doubling the number of eigenpairs calculated at each level (from fine to coarse grids). For example, the combinations which correspond to $r = 8$ are as follows: $N_e = (0, 0, 24, 48)$ and $N_e = (2, 4, 8, 16)$. The distances achieved using this strategy for various values of r are displayed in Figure 6 using red crosses. Although these combinations do not always give rise to the minimum distance, they usually give reasonable approximations. This doubling strategy will be adopted in the next section.

5.5. Using $\tilde{H}^{-1}$ as a preconditioner for Gauss–Newton. In the second set of experiments, we use our multilevel approximation to H^{-1} in a typical application. Specifically, we recall the case mentioned in section 1.2 of incremental 4D-Var, where the solution of a system of linear equations of the form (1.9) with coefficient matrix $\mathcal{H}$ has to be approximated at each step of a Gauss–Newton process. This is typically achieved using a few CG iterations. In realistic DA applications, the number of Gauss–Newton (outer) iterations as well as the number of CG (inner) iterations is limited by the time available in the forecast window. More details of approximate Gauss–Newton methods for large-scale DA problems can be found in [19].

After first-level preconditioning, system (1.9) takes the form

$$H(u_i)\delta v_i = -\left(V_b^{1/2}\right)^* G(u_i), \tag{5.5}$$

where $\delta v_i = V_b^{-1/2}\delta u_i$. As we are assuming that this preconditioning is always applied, we will refer to this case as "unpreconditioned Gauss–Newton." The convergence of the CG method applied to (5.5) can be further accelerated by preconditioning the system again using $\tilde{H}^{-1}$ to improve the resulting eigenspectrum, in other words, by solving

$$\tilde{H}^{-1}(u_i)H(u_i)\delta v_i = -\tilde{H}^{-1}(u_i)(V_b^{1/2})^* G(u_i).$$

Here the preconditioner $\tilde{H}^{-1}(u_i)$ is computed once per Gauss–Newton step (before iterating with CG): this is an important difference from the approach presented in [11], where the multigrid cycle is applied as a preconditioner for each CG iteration. In what follows, we apply incremental 4D-Var to the Burgers' test problems and investigate the effect of preconditioning using multilevel approximations $\tilde{H}^{-1}$ built using Algorithms 4.2 and 4.3. The Gauss–Newton stopping tolerance is 10^{-4}, and five CG iterations are carried out for each linear solve. In each simulation, the multilevel preconditioner is not switched on until after the first few (five in our examples) Gauss–Newton iterations have been carried out. This is to ensure convergence of the outer iteration as our problem involves strongly nonlinear constraints, in the form of Burgers' equation (see, for example, [33]).

Details of the specific algorithms and eigenvalue combinations used for the preconditioners tested are given in Table 3. Preconditioners P2a, P2b, and P2c use Algorithm 4.2 with different values of N_e in the multilevel representation of the inverse

TABLE 3
Description of parameters used in multilevel preconditioners.

Preconditioner	Algorithm	N_e	k'	$n^l_{k'}$	$N^l_{k'}$
P2a	2	(200,0,0,0)	1	8	-
P2b	2	(0,8,16,32)	1	8	-
P2c	2	(0,4,8,16)	1	8	-
P3a	3	(0,8,16,32)	1	8	(0,0,8,0)
P3b	3	(0,8,16,32)	2	8	(0,0,0,8)

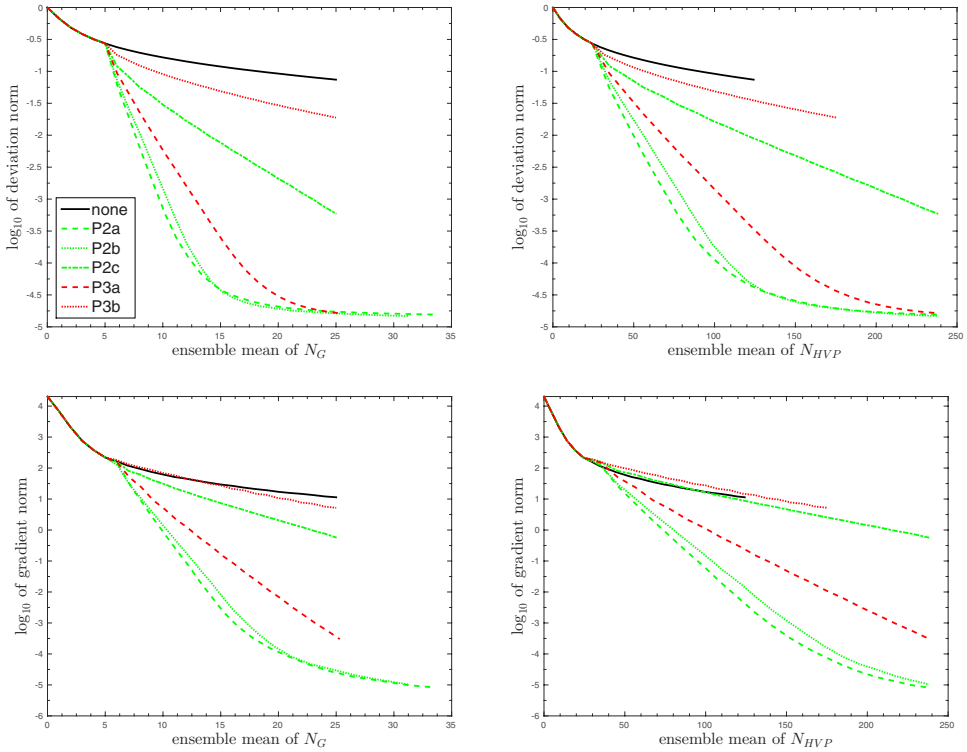

FIG. 7. *Convergence diagram for MP*1.

Hessian H^{-1} defined by (3.3). For each elementary Hessian $\hat{H}^l_{k'}$, the Hessian-vector product is defined at level $k' = 1$, $m = 201$, with $n^l_{k'} = 8$ eigenpairs used for its limited-memory representation. Preconditioners P3a and P3b use Algorithm 4.3 with one N_e (such that $r = 12$) in the multilevel representation of H^{-1}. For case P3a, the Hessian-vector product is again defined at level $k' = 1$, with $n^l_{k'} = 8$ eigenpairs used for limited-memory representation of $(\hat{H}^l_{k'})^{-1}$. The multilevel representation of $\hat{H}^l_{k'}$ is then built according to $N^l_{k'}$, that is, at level $k = 2$. For case P3b, the Hessian-vector product is defined at level $k' = 2$, $m = 101$, with $n^l_{k'} = 8$ eigenpairs used for limited-memory representation of $(\hat{H}^l_{k'})^{-1}$. The multilevel representation of $\hat{H}^l_{k'}$ is then built at level $k = 3$ (the coarsest level). For each test case, the minimization problem (1.2) was solved 25 times with perturbed data u_b and φ_{obs} as described in [16, section 5.2], and the resulting ensemble averaged values characterizing the convergence rate and the solution accuracy are presented in convergence diagrams shown in Figures 7–10.

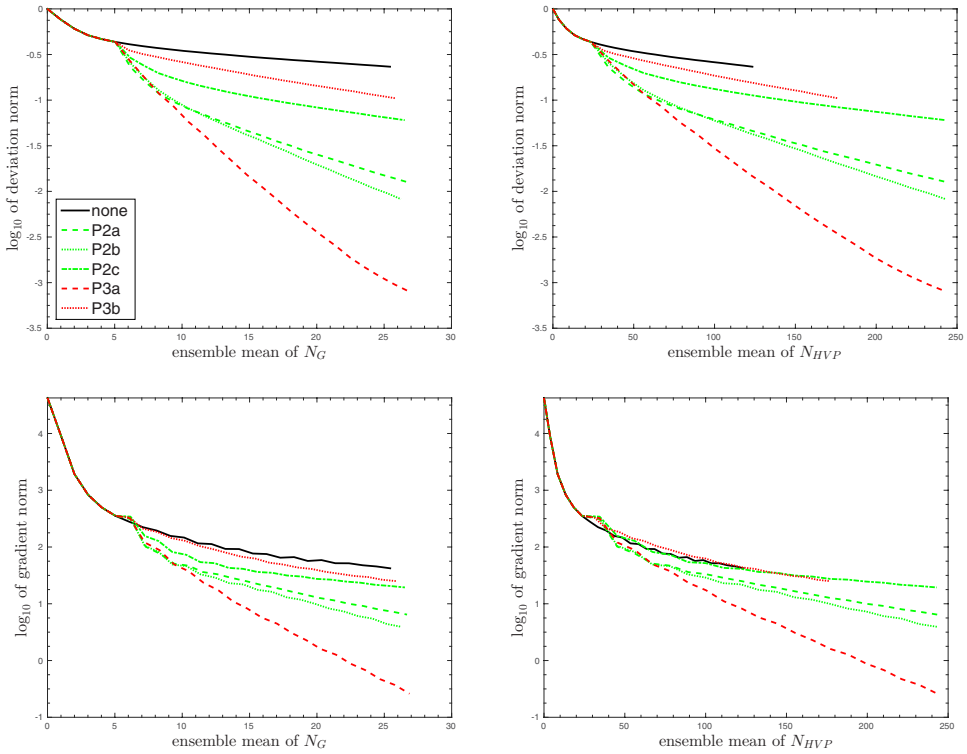

FIG. 8. *Convergence diagram for MP2.*

Each subplot of a diagram shows two major convergence indicators: the averaged deviation norm $\|u_i - u_\infty\|_2$ (where u_i is the iterate after i iterations and u_∞ is the final estimate $\bar{u}$) and the gradient norm $\|G(u_i)\|_2$, plotted in $\log_{10}$-scale against computational time measured in two different units. These are the number of the cost-function/gradient evaluations and the number of Hessian-vector product evaluations (denoted by N_G and N_{HVP}, respectively) at the finest discretization level needed to carry out the whole solution procedure (that is, the cost of constructing and applying any preconditioning is included in these figures). The former coincides with the actual number of the cost-function/gradient evaluations; the latter also includes the maximum time spent on evaluating an individual $\hat{H}^l$ (we assume that $\hat{H}^l$ are evaluated in parallel).

To assess preconditioner performance, as compared to unpreconditioned Gauss–Newton, the convergence diagrams should be interpreted together as follows. Suppose that one cost-function/gradient evaluation takes time t_G, and one Hessian-vector product evaluation takes time t_{HVP}.

Let us denote $\epsilon = \log_{10} \|u_i - u_\infty\|_2$; then the relative accuracy of u_i is $10^{\epsilon} \times 100\%$. For a given relative accuracy the number of cost-function/gradient evaluations N_G required to achieve it with a given preconditioner can be read from the top left subplot as the abscissa of the point where the level line ϵ crosses the relevant curve. The value of the gradient norm $\varepsilon = \log_{10} \|G(u_i)\|_2$ which corresponds to this ϵ can then be read from the lower left subplot as the value at this abscissa. The number of required Hessian-vector products N_{HVP} is available from the top right subplot in a similar way. The resulting values N_G and N_{HVP} can then be used to evaluate the total time

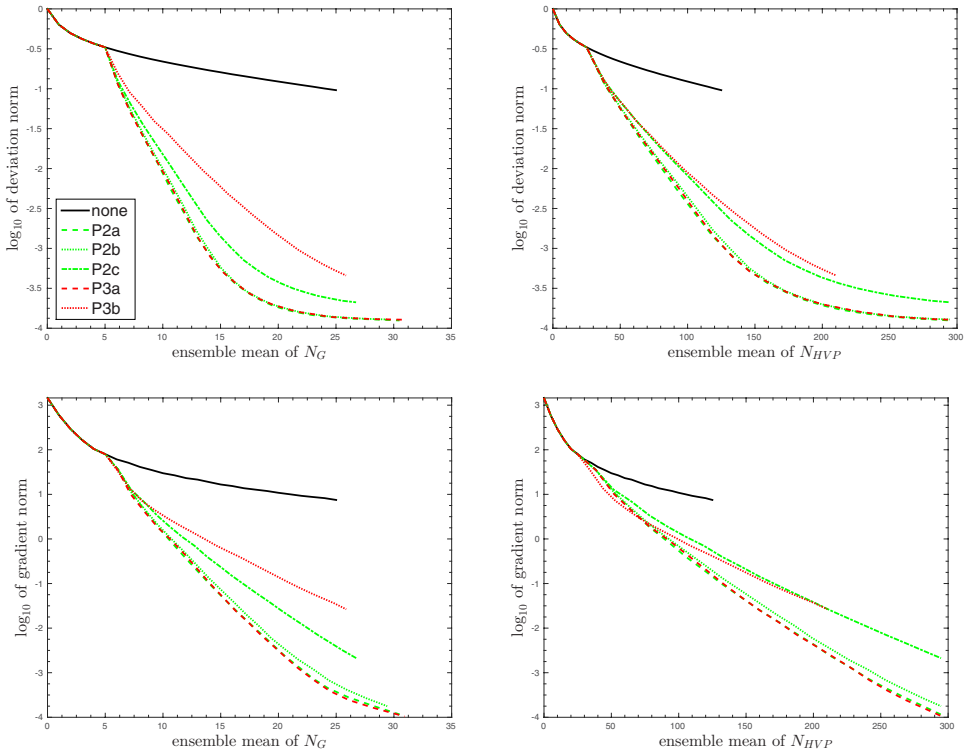

FIG. 9. *Convergence diagram for MP*3.

required as $t_{total} = N_G \times t_G + N_{HVP} \times t_{HVP}$. Since one unpreconditioned Gauss–Newton iteration takes a time of $t = t_G + N_{CG} \times t_{HVP}$, this total amount of time can be spent on i^* unpreconditioned iterations where

$$i^*(\rho) = \frac{t_{total}}{t} = \frac{\rho N_G + N_{HVP}}{\rho + N_{CG}}$$

and $\rho = t_G/t_{HVP}$. Once i^* has been identified, the accuracy ϵ^* achieved by unpreconditioned Gauss–Newton in the same time as the preconditioned version gives an accuracy of ϵ can be read off from the top left subplot as the value from the curve labelled "none" at the point $N_G = i^*$. The difference $\epsilon - \epsilon^*$ reflects the accuracy gain due to preconditioning, achieved in the time $i^* \times t_G$. The value of the gradient norm ε^* achieved by unpreconditioned Gauss–Newton in the same time as the preconditioned version gives the gradient norm ε can also be read from the bottom left subplot as the value of the corresponding graph at $N_G = i^*$. The difference $\varepsilon - \varepsilon^*$ provides the convergence gain due to preconditioning, achieved in the time $i^* \times t_G$. Alternatively, the same procedure can be replicated starting from a desired gradient norm reduction defined by ε, in which case the bottom right subplot will be involved.

Some results of such analysis with $\rho = 1$ and $\epsilon = -2$ are summarized in Table 4, which shows the relative accuracy achieved by unpreconditioned GN by time preconditioned GN achieved relative accuracy 1%. For case MP2, when the values cannot be identified from the figures shown, the table fields are empty, and where the values have been obtained by extrapolation, they are marked as approximate. The larger the value in Table 4, the less accuracy has been achieved by using the unpreconditioned

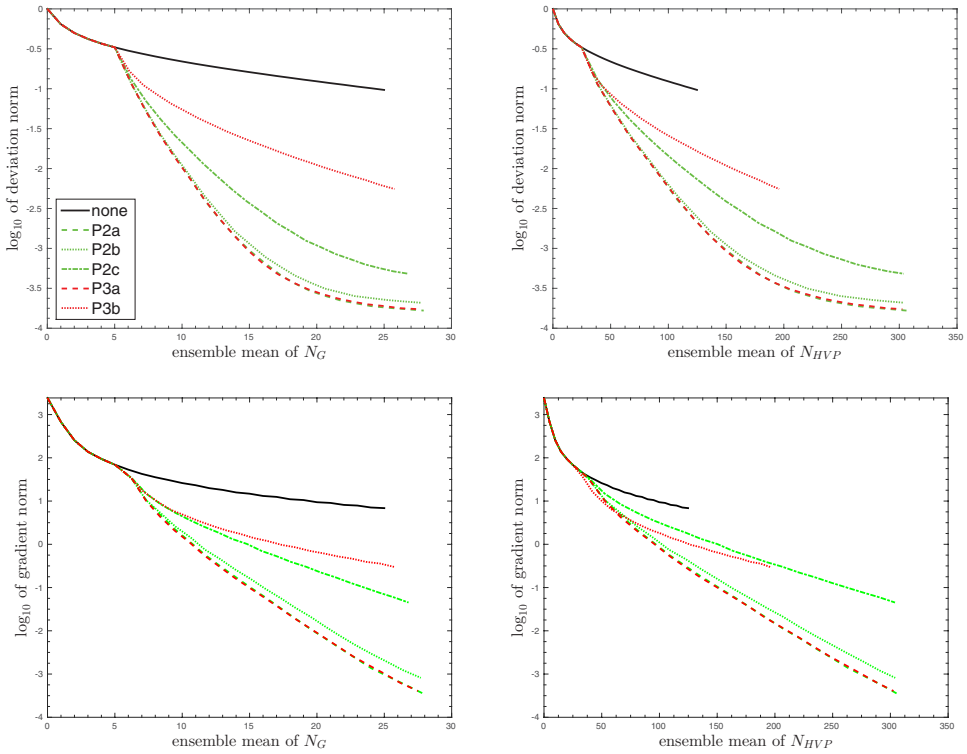

FIG. 10. *Convergence diagram for MP4.*

TABLE 4

Relative accuracy achieved by unpreconditioned GN by time preconditioned GN achieved relative accuracy 1%.

Preconditioner.	Case MP1	Case MP2	Case MP3	Case MP4
P2a	15.8%	15.7%	12.5%	12.0%
P2b	13.1%	17.5%	12.5%	12.0%
P2c	6.3%	-	10.0%	10.0%
P3a	10.0%	20.0%	12.5%	12.0%
P3b	$\approx 3.0\%$	-	9.0%	$\approx 7.0\%$

Gauss–Newton method, and the more efficient the chosen multilevel preconditioner has been. Note that $\rho \approx 1$ relates to the case where computational time for solving the original nonlinear model and the tangent linear/adjoint model sequence is comparable. Similar tables can be obtained for different values of ρ and ϵ.

These results demonstrate that given a fixed computational time, the optimal solution accuracy gain due to multilevel preconditioning is about an order of magnitude. Therefore, the suggested approach may be considered as an important resource for parallelization of the minimization process itself. It is clear that for the variants using Algorithm 4.2, the performance with P2a and P2b is quite similar, that is, P2b essentially replicates the performance of the full eigenvalue decomposition based P2a, with much-reduced memory requirements ($r = 12$ as opposed to $r = 200$). With MP1, the performance of preconditioner P2c ($r = 6$) degrades, as not enough information has been retained at the local level (although it still represents a significant

improvement over the unpreconditioned version). However, in case MP3 preconditioner P2c performs nearly as well as P2b. As discussed in section 5.3, there is a difference between stationary and moving sensors in terms of their influence on the inverse Hessian structure in that, in the case of moving sensors, a much better quality multilevel representation can be achieved within the same memory allowance. The performance of P2c in cases MP3 and MP4 certainly confirms this. For Algorithm 4.3, as expected, the additional level of approximation in P3a means that it does not perform quite as well as P2a in cases MP1 and MP2, but it is still a very competitive method. With preconditioner P3b in cases MP1 and MP2, however, it appears that the local Hessian calculations have been done on too coarse a grid so too much information has been lost. At the same time, P3b performs very well in cases MP3 and MP4 (with convergence curves almost indistinguishable from those of P2a), which is another indication that this type of preconditioning could be very useful in assimilating satellite data.

It is interesting to note that in case MP2, method P3a performs better than P2a, which on the surface seems out of line with the other results. However, from Figure 2 (right) it can be seen that the flow solution for initial state u^2_{true} contains much stronger field gradients than in case u^1_{true}. This is combined with singular-type data from stationary sensors in case MP1, which explains slower convergence of the Gauss–Newton process in comparison to all other cases. For linear model equations, the Gauss–Newton method must converge in one iteration if the inner problem (1.9) is solved exactly. In the case of nonlinear model equations, many Gauss–Newton iterations are needed, and solving the inner problem too accurately has rather a negative impact on the Gauss–Newton convergence rate. This explains the better performance with a less accurate approximation as preconditioner, as seen with P3a in case MP2.

6. Conclusion. The inverse Hessian of the auxiliary DA problem (1.5) plays an important part in different aspects of variational DA. The Hessian-vector product is defined by sequential solution of the tangent linear and adjoint problems; for the inverse Hessian (or its square root), no such definition is possible. In high dimensions, the requirement to work in a matrix-free environment means that compact representation schemes are of significant interest. The simplest one, based on the eigenvalue decomposition of the projected Hessian after first-level preconditioning, however, is not good enough. Here we have introduced the novel concept of a multilevel eigenvalue decomposition, which results in a much more efficient compact representation of the inverse Hessian (and its square root). At a given level, the eigensystem of an operator preconditioned by its own approximation from the next coarser level is computed, ascending from the coarsest to the finest level. The numerical results in section 5.4 demonstrated that, given a specified memory allowance, the inverse Hessian approximation accuracy is greatly improved as compared to a one-level eigenvalue decomposition scheme. We also believe that a similar algorithm can be utilized beyond the application considered here, for example, for any symmetric operator resulting from the discretization of a PDE, or for image compression and restoration. In such cases, the resulting low-memory representations of the inverse operator could be particularly useful for preconditioning in problems with multiple right-hand sides.

We have also considered the application of our compact inverse Hessian approximations as preconditioners for a Gauss–Newton minimization procedure. Here we introduced a further novel decomposition principle, namely, allowing the Hessian to be represented by the sum of elementary Hessians, which can be evaluated (and com-

pressed) in parallel. The numerical results in section 5.5 show that given a fixed execution time, a much more accurate approximation can be computed as compared to using unpreconditioned Gauss–Newton method. The new multilevel method therefore offers an important parallelizable resource applicable directly to minimization problems. This offers a significant advantage over the multigrid approach in [11], which is intrinsically sequential.

The applicability of our method in the context of present-day operational 4D-Var systems depends, in theory, on the eigenvalue structure of the associated Hessians (local and global). In practice, it will also depend on many different factors which are difficult to assess at this stage. As an intermediate step, we intend to test our method with a global shallow water equations (SWE) model defined on a sphere [34]. This will allow us to consider a few features which are not involved with the current Burgers' model. In particular, the SWE model is a full two-dimensional model which includes three state variables and also supports long wave transport (rather than advection/diffusion). We are hopeful that our new approach will also work well in this more realistic situation.

Appendix A. The tangent linear model $R'(u)v = \psi$ for Burgers' equation (5.1) is given by

$$\begin{cases} \dfrac{\partial \psi}{\partial t} = -\dfrac{\partial(\varphi\psi)}{\partial x} + \dfrac{\partial}{\partial x}\left(\mu(\varphi)\dfrac{\partial \psi}{\partial x}\right) + \dfrac{\partial}{\partial x}\left(\mu'(\varphi)\psi\dfrac{\partial \varphi}{\partial x}\right), \\ \left.\dfrac{\partial \psi}{\partial x}\right|_{x=0,1} = 0, \\ \psi|_{t=0} = v, \end{cases}$$

where $\varphi = \varphi(u, x, t),\ \psi = \psi(x, t),\ x \in [0, 1],\ t \in [0, T]$. The associated adjoint model $R'^*(u)\psi = v$ is

$$\begin{cases} -\dfrac{\partial \psi^*}{\partial t} = \varphi\dfrac{\partial \psi^*}{\partial x} + \dfrac{\partial}{\partial x}\left(\mu(\varphi)\dfrac{\partial \psi^*}{\partial x}\right) - \mu'(\varphi)\dfrac{\partial \varphi}{\partial x}\dfrac{\partial \psi^*}{\partial x} + \ \psi, \\ \left.\left(\mu(\varphi)\dfrac{\partial \psi^*}{\partial x} + \varphi\psi^*\right)\right|_{x=0,1} = 0, \\ \psi^*|_{t=T} = 0, \\ \psi^*|_{t=0} = v, \end{cases}$$

where $\varphi = \varphi(u, x, t),\ \psi^* = \psi^*(x, t),\ \psi = \psi(x, t),\ x \in [0, 1],\ t \in [T, 0]$.

Acknowledgment. The second author thanks Prof. Achi Brandt of the Weizmann Institute of Science, Israel, for joint work on using multigrid methods for solving control problems.

REFERENCES

[1] T. Amemiya, *Non-linear regression models*, in Handbook of Econometrics, North-Holland, Amsterdam, 2002.

[2] W. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.

[3] N. Bakhvalov, *On the convergence of a relaxation method with natural constraints on the elliptic operator*, USSR Comp. Math. Math. Phys., 6 (1966), pp. 101–113.

[4] A. Borzi and V. Schulz, *Multigrid methods for pde optimization*, SIAM Rev., 51 (2009), pp. 361–395.

[5] A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comp., 31 (1977), pp. 333–390.

[6] Y. Chen and D. Oliver, *Ensemble randomized maximum likelihood method as an iterative ensemble smoother*, Math. Geosci., 44 (2012), pp. 1–26.
[7] S. Costiner and S. Ta'asan, *Adaptive multigrid techniques for large-scale eigenvalue problems: Solutions of the Schrodinger problem in two and three dimensions*, Phys. Rev., 44 (1995), pp. 3704–3717.
[8] P. Courtier, J. Thepaut, and A. Hollingsworth, *A strategy for operational implementation of 4D-Var, using an incremental approach*, Quart. J. Roy. Meteor. Soc., 120 (1994), pp. 1367–1387.
[9] J. Cullum and R. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Vol.* I*: Theory*, Classics in Appl. Math., SIAM, Philadelphia, 2002.
[10] M. Dashti, K. Law, A. Stuart, and J. Voss, *Map estimators and posterior consistency in Bayesian nonparametric inverse problems*, Inverse Problems, 29 (2013), 095017.
[11] L. Debreu, E. Neveu, E. Simon, F.-X. L. Dimet, and A. Vidard, *Multigrid solvers and multigrid preconditioners for the solution of variational data assimilation problems*, Quart. J. Roy. Meteor. Soc., 142 (2016), pp. 515–528.
[12] F.-X. L. Dimet and O. Talagrand, *Variational algorithms for analysis and assimilation of meteorological observations: Theoretical aspects*, Tellus A, 38 (1986), pp. 97–110.
[13] M. Ehrendorfer and J. Tribbia, *Optimal prediction of forecast error covariances through singular vectors*, J. Atmos. Sci., 54 (1997), pp. 286–313.
[14] E. Epstein, *The role of initial uncertainties in prediction*, J. Appl. Meteor., 8 (1969), pp. 190–198.
[15] D. Furbish, M. Hussaini, F.-X. L. Dimet, P. Ngnepieba, and Y. Wu, *On discretization error and its control in variational data assimilation*, Tellus A, 60 (2008), pp. 979–991.
[16] I. Gejadze, F.-X. L. Dimet, and V. Shutyaev, *On analysis error covariances in variational data assimilation*, SIAM J. Sci. Comput., 30 (2008), pp. 1847–1874.
[17] I. Gejadze, F.-X. L. Dimet, and V. Shutyaev, *On optimal solution error covariances in variational data assimilation problems*, J. Comput. Phys., 229 (2010), pp. 2159–2178.
[18] I. Gejadze, F.-X. L. Dimet, and V. Shutyaev, *Computation of the analysis error covariance in variational data assimilation problems with nonlinear dynamics*, J. Comput. Phys., 230 (2011), pp. 7923–7943.
[19] S. Gratton, A. Lawless, and N. Nichols, *Approximate Gauss–Newton methods for nonlinear least squares problems*, SIAM J. Optim., 18 (2007), pp. 106–132.
[20] S. Haben, A. Lawless, and N. Nichols, *Conditioning and preconditioning of the variational data assimilation problem*, Comput. & Fluids, 46 (2011), pp. 252–256.
[21] W. Hackbusch, *On the computation of approximate eigenvalues and eigenfunctions of elliptic operators by means of a multigrid method*, SIAM J. Numer. Anal., 16 (1979), pp. 201–215.
[22] L. Hascoet and V. Pascual, *Tapenade* 2.1 *User's Guide*, Technical Report, Inria Sophia Antipolis, Paris, 2004.
[23] M. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stand., 49 (1952), pp. 409–436.
[24] T. Hwang and I. Parsons, *A multigrid method for the generalized symmetric eigenvalue problem: Part* I *— algorithm and implementation*, Internat. J. Numer. Methods Engrg., 35 (1992), pp. 1663–1676.
[25] A. Knyazev and K. Neymeyr, *Efficient solution of symmetric eigenvalue problems using multigrid preconditioners in the locally optimal block conjugate gradient method*, Electron. Trans. Numer. Anal., 15 (2003), pp. 38–55.
[26] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Natl. Bur. Stand., 45 (1950), pp. 225–282.
[27] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, Software Environ. Tools, SIAM, Philadelphia, 1998.
[28] J.-L. Lions, *Contrôle Optimal des Systèmes Gouvernés par des Équations aux Dérivées Partielles*, Dunod, Paris, 1968.
[29] C. Liu, Q. Xiao, and B. Wang, *An ensemble-based four-dimensional variational data assimilation scheme. part* i*: Technical formulation and preliminary test*, Monthly Weather Rev., 136 (2008), pp. 3363–3373.
[30] G. Marchuk, V. Agoshkov, and V. Shutyaev, *Adjoint Equations and Perturbation Algorithms in Nonlinear Problems*, CRC Press, New York, 1996.
[31] I. Mirouze and A. Weaver, *Representation of correlation functions in variational assimilation using an implicit diffusion operator*, Quart. J. Roy. Meteorol. Soc., 136 (2010), pp. 1421–1443.
[32] M. Moakher, *A differential geometric approach to the geometric mean of symmetric positive-definite matrices*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 735–747.

[33] L. Nazareth, *Recent approaches to solving large residual nonlinear least squares problems*, SIAM Rev., 22 (1980), pp. 1–11.

[34] B. Neta, F. Giraldo, and I. Navon, *Analysis of the Turkel-Zwas scheme for the two-dimensional shallow water equations in spherical coordinates*, J. Comput. Phys., 133 (1997), pp. 102–112.

[35] S. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing, New York, 1980.

[36] F. Rabier and P. Courtier, *Four-dimensional assimilation in the presence of baroclinic instability*, Quart. J. Roy. Meteorol. Soc., 118 (1992), pp. 649–672.

[37] F. Rabier, H. Järvinen, E. Klinker, J.-F. Mahfouf, and A. Simmons, *The ECMWF operational implementation of four-dimensional variational assimilation.* I: *Experimental results with simplified physics*, Quart. J. Roy. Meteorol. Soc., 126 (1992), pp. 1142–1170.

[38] G. Sleijpen and H. V. der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[39] W. Thacker, *The role of the Hessian matrix in fitting models to measurements*, J. Geophys. Res., 94 (1989), pp. 6177–6196.

[40] Z. Toth and E. Kalnay, *Ensemble forecasting at NMC: The generation of perturbations*, Bull. Amer. Meteor. Soc., 74 (1993), pp. 2317–2330.

[41] U. Trottenberg, C. Oosterlee, and A. Schller, *Multigrid*, Academic Press, London, 2001.