Efficient Computation of Observation Impact in 4D-Var Data Assimilation

Alexandru Cioaca[1], Adrian Sandu[1], Eric De Sturler[2],
and Emil Constantinescu[3]

[1] Department of Computer Science
[2] Department of Mathematics
Virginia Tech
Blacksburg, VA, USA
[3] Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL, USA {alexgc, sandu, sturler, emconsta}@vt.edu

Abstract. Data assimilation combines information from an imperfect model, sparse and noisy observations, and error statistics, to produce a best estimate of the state of a physical system. Different observational data points have different contributions to reducing the uncertainty with which the state is estimated. Quantifying the observation impact is important for analyzing the effectiveness of the assimilation system, for data pruning, and for designing future sensor networks. This paper is concerned with quantifying observation impact in the context of four dimensional variational data assimilation. The main computational challenge is posed by the solution of linear systems, where the system matrix is the Hessian of the variational cost function. This work discusses iterative strategies to efficiently solve this system and compute observation impacts.

Keywords: Four dimensional variational data assimilation, observation impact, nonlinear optimization, preconditioning, adjoint model, iterative linear solvers

1 Introduction

This paper discusses a framework for computing the impact of observations in four dimensional variational (4D-Var) data assimilation. The purpose of this framework is to reveal the degree of usefulness of the observational data in the process of correcting the *a priori* knowledge of the state of the system. From the analysis of observation impact, certain decisions can be inferred about the strategy to collect observations. The main computational challenge associated with estimating observation impact is the solution of a linear system where the matrix is the Hessian of the 4D-Var cost function. This Hessian matrix is usually very large and accessible only through matrix-vector products. A high accuracy solution is typically sought. In applications such as real-time sensor deployment, this solution needs to be obtained quickly.

This paper discusses the particular characteristics of the linear system involved in observation impact calculations. We study the use of several iterative linear solvers that use only Hessian-vector products, and propose several inexpensive preconditioners to improve convergence. Numerical tests using shallow water equations illustrate the improvements in efficiency resulting from this methodology.

The paper is organized as follows. Sect. 2 introduces variational data assimilation. Sect. 3 zooms in on the topic of observation impact in the context of data assimilation. Sect. 4 presents the shallow water equations test problem and its implementation. Sect. 5 gives details of the data assimilation experiment. A discussion of the main computational issues and the proposed solutions can be found in Sect. 6.

2 Data Assimilation

Data assimilation is the process by which measurements are used to constrain model predictions [1,2]. The information from measurements can be used to obtain initial and boundary conditions that approximate better the real state of the model at a particular moment. Variational data assimilation allows the optimal combination of three sources of information: *a priori* (background) estimate of the state of the system, knowledge of the interrelationships among the variables of the model, and observations of some of the state variables. The optimal initial state $\mathbf{x}^a$ (also called the analysis) is obtained by minimizing the cost function

$$\begin{aligned} \mathcal{J}(\mathbf{x}_0) = &\frac{1}{2}(\mathbf{x}_0 - \mathbf{x}^b)^T \cdot B^{-1} \cdot (\mathbf{x}_0 - \mathbf{x}^b) \\ &+\frac{1}{2}\sum_{k=1}^{N} (H_k(\mathbf{x}_k) - \mathbf{y}_k)^T \cdot R_k^{-1} \cdot (H_k(\mathbf{x}_k) - \mathbf{y}_k) \qquad (1) \\ \mathbf{x}^a = &\text{ Arg min } \mathcal{J} \end{aligned}$$

The first term of the sum quantifies the departure from the background state $\mathbf{x}^b$ at the initial time t_0. The second term measures the distance to the observations $\mathbf{y}_k$, which are taken at N times t_k inside the assimilation window. When assimilating observations only at the initial time t_0, the cost function is known as 3D-Var. Otherwise, when the observations are distributed in time, the cost function is known as 4D-Var. In this study we will focus on 4D-Var. The block-diagonal background error covariance matrix B is built to take into account the spatial correlations of the variables, as well as the periodic boundary conditions. H_k is the observation operator defined at assimilation time t_k. It maps the discrete model state $\mathbf{x}_k \approx \mathbf{x}(t_k)$ to the observation space. R_k is the observations error covariance matrix. The weighting matrices B and R need be synthesized in order to have a fully-defined problem and their quality will influence the accuracy of the analysis.

The efficient numerical minimization of (1) requires the gradient of the cost function or second-order derivative information when available. 4D-Var usually

relies on adjoint sensitivity analysis to provide information about the first and second-order derivatives of the objective function. For instance, the gradient of the cost function can be given by the first-order adjoint model, while the second-order information can be computed under the form of a Hessian-vector product from the second-order adjoint. Variational data assimilation is an example of PDE-constrained optimization as it is a problem of nonlinear optimization where the the minimization of the cost function is constrained by a numerical model associated with a set of PDEs. The minimizer of the 4D-Var cost function can only be computed iteratively using gradient-based methods, because an analytical solution is almost impossible to derive. When using 4D-Var to correct initial and boundary conditions in a real-time operational setting, we have a limited number of model runs that can be used before the time window expires. This usually implies the minimization process is halted after a number of iterations, so the global minimum might not be attained. Although the most significant decrease in the cost function usually happens during the first iterations, it is important to take into account that the computed solution might not satisfy the optimality conditions. For more details, see the studies in [3,4].

3 Observation Impact in 4D-Var Data Assimilation

Methods for computing observation impact have been initially developed for the framework of 3D-Var data assimilation in [5,6]. The extension of this theory to address 4D-Var data assimilation was only recently accomplished and the details can be found in [7,8].

If the 4D-Var problem is defined and solved accurately, then the analysis $\mathbf{x}^a$ should provide a better forecast than the background $\mathbf{x}^b$. Since one does not usually have access to the real state of the system (otherwise we would not have to perform data assimilation), the accuracy of the analysis is verified through a forecast score against a solution of higher accuracy. Let M be a nonlinear forecast model that discretizes a system of time-dependent partial differential equations, while we will denote its tangent linear and first-order adjoint models with $\mathcal{M}$ and $\mathcal{M}^T$. A forecast score $e(\mathbf{x}^a)$ is defined on M as a short-range forecast error measure:

$$e(\mathbf{x}^a) = (\mathbf{x}_f^a - \mathbf{x}_f^v)^T C(\mathbf{x}_f^a - \mathbf{x}_f^v) \tag{2}$$

where $\mathbf{x}_f^a = M_{t_0 \to t_f}(\mathbf{x}^a)$ is the model forecast at verification time t_f, $\mathbf{x}_f^v$ is the verifying analysis at t_f and C is a matrix that defines the metric in the state space.

Following the derivations in [7], the equation of forecast sensitivity to observations is:

$$\nabla_{\mathbf{y}} e(\mathbf{x}^a) = R^{-1} H A \nabla_{\mathbf{x}} e(\mathbf{x}^a) \tag{3}$$

where

$$A = [\nabla^2_{\mathbf{xx}} \mathcal{J}(\mathbf{x}^a)]^{-1} \tag{4}$$

denotes the inverse of the Hessian matrix of the cost at $\mathbf{x}^a$ and it is assumed to be a positive definite matrix.

The same paper [7] provides an approximation to the impact of observation, which is defined as the change the observation brings to the value of the 4D-Var cost function. If we consider the innovation vector as $\delta y = y - h(\mathbf{x}^b)$, then a first-order formula would be

$$\delta e = e(\mathbf{x}^a) - e(\mathbf{x}^b) = (\delta \mathbf{y})^T R^{-1} H A \nabla_{\mathbf{x}} e(\mathbf{x}^a) \tag{5}$$

The elements of the innovation vector are individual observations that get multiplied with their corresponding sensitivity. For those observations that contributed to the decrease of the cost function, the impact (individual value of δe) will be a negative value. The matrix vector product in the matrix A can be computed as a linear system of the form

$$A^{-1} \cdot \mu_0 = \nabla_{\mathbf{x}} e(\mathbf{x}^a) \tag{6}$$

The value of μ_0 is associated with the sensitivity at the initial time. By initializing the tangent linear model with this vector of sensitivities and integrating it from this time to the final time of the assimilation window, we can obtain the sensitivities to observations corresponding to the rest of the assimilation windows.

The problem of computing observation impact has to be solved in real-time since these results are used in applications such as sensor network deployment. Numerical models that simulate the atmosphere are very expensive from a computational standpoint and their adjoint models even more so. Most of the computation bulk is in computing the solution of the linear system $A^{-1} \cdot \mu_0 = \nabla_{\mathbf{x}} e(\mathbf{x}^a)$. This linear system has to be solved with a high degree of accuracy because its solution μ_0 is a vector representing a "supersensitivity". Any errors that it contains can be propagated and amplified by the numerical models used for forecast or sensitivity analysis. At the same time, there is also a constraint on the number of model runs one can execute before the results are due for operational use. The Hessian matrix of a system associated with a numerical model underlined by PDEs is usually sufficiently complicated that an iterative solver generally will not converge to a satisfactory solution in a few iterations unless it is tweaked accordingly. Accuracy of the solution and fast convergence of the solver are two of the main challenges when computing observation impact in data assimilation, along with means of computing the second-order information.

4 The Shallow Water Equations Test Case

We consider the two dimensional Saint-Venant PDE system that approximates fluid flow inside a shallow basin (also known as "Shallow Water Equations" (SWE)):

$$\frac{\partial}{\partial t}h + \frac{\partial}{\partial x}(uh) + \frac{\partial}{\partial y}(vh) = 0$$
$$\frac{\partial}{\partial t}(uh) + \frac{\partial}{\partial x}(u^2h + \frac{1}{2}gh^2) + \frac{\partial}{\partial y}(uvh) = 0$$
$$\frac{\partial}{\partial t}(vh) + \frac{\partial}{\partial x}(uvh) + \frac{\partial}{\partial y}(v^2h + \frac{1}{2}gh^2) = 0\ .$$

The spatial domain is square shaped ($\Omega = [-3,\ 3]^2$), and the integration window is $t^0 = 0 \leq t \leq t^F = 0.1$. Here $h(t, x, y)$ denotes the fluid layer thickness, and $u(t, x, y)$, $v(t, x, y)$ are the components of the velocity field. g is the standard value of the gravitational acceleration. The boundary conditions are periodic in both directions. For ease of presentation, we arrange the n discretized state variables in a column vector

$$x = \begin{bmatrix} \hat{h} \\ \hat{uh} \\ \hat{vh} \end{bmatrix} \in R^n\ . \tag{7}$$

In order to solve these equations, we implemented a numerical model that makes use of a finite volume-type scheme for space discretization and a fourth-order Runge-Kutta scheme for timestepping. This method was introduced by Liska and Wendroff in [9]. In the following, forward solves will be denoted by FWD.

The tangent linear model (TLM), first-order (FOA) and second-order (SOA) adjoints were generated through automatic differentiation of the forward model using TAMC [10,11]. TAMC is a source-to-source translator that generates Fortran code for computing the derivatives of a numerical model. For more details about automatic differentiation see [12]. The FOA and SOA are integrated backwards in time and they require the values computed at each timestep for the FWD and TLM, respectively. There are two ways to generate these numerical values. The first one is to run the FWD and TLM in advance and to store their evolution in time; this approach has the disadvantage of taking up a lot of storage space. The second one is to recompute the FWD and TLM at each timestep where the FOA and SOA need their values. Theoretically this is necessary at each timestep, which means the latter approach saves storage space at the expense of increasing demand for computation.

A question that arises naturally is how large is the overhead of the code generated through automatic differentiation. Table 1 illustrates the CPU times of TLM, FOA and SOA models, normalized with respect to the CPU time of a single forward model run. It is seen that a full SOA integration is about 3.5 times more expensive than a single first-order adjoint run, while this FOA takes 3.7 times longer than the forward run. The time for SOA can be considered to be a large value as it takes longer time to compute the SOA than to approximate it through finite differences (two FOA runs). However, for the purpose of this test they are computationally feasible. Please note that these values apply only for our

FWD	1		
TLM	2.5	FWD + TLM	3.5
FOA	3.7	FWD + FOA	4.7
SOA	12.8	FWD + TLM + FOA + SOA	20.0

Table 1. Ratio between the CPU time of adjoint models and the forward model

particular implementation (method of lines) and the overhead can greatly vary with the complexity of the numerical method or the automatic differentiation tool used.

We mentioned that each time the tangent linear model or first-order adjoint are run, the forward model is run beforehand for storing its evolution. Similarly, in order to compute the second-order adjoint one must first run the forward, first-order adjoint and tangent linear models. As such, the actual timing values are the ones displayed in the right hand side of the table.

5 Data Assimilation Scenario

Prior to computing observation impact, we have to perform the 4D-Var data assimilation experiment with the SWE model presented in the previous section. Following are the parameters of the Data Assimilation System (DAS) we are using:

- The 2-D grid of the SWE model is divided into 40 grid points on each direction (1600 in total). The number of time steps for the model runs is fixed at $N = 1000$.
- The reference solution is generated component-wise: the height component h will be a Gaussian pulse of amplitude $A = 30$ while the velocity fields u and v are assigned a value of 2 at each grid point.
- The background solution $\mathbf{x}^b$ is generated by applying a correlated perturbation on the reference profile for h, u and v.
- The background error covariance B was generated for a standard deviation of 5% with a nondiagonal structure and correlation distance of five grid points. This will help the 4D-Var method to spread the information from observations in each cell based on information passed from its neighbors.
- The model was run with the reference as initial condition in order to generate the synthetic observations $\mathbf{y}_k$. The observation frequency is set to once every 20 time steps. In order to simulate the effect of noise over observations, we apply a normal random perturbation to the perfect synthetic observations.
- The observation error covariance R is a diagonal matrix, based on the assumption that the observational errors are uncorrelated. The standard deviation of these errors was set to 1% of the largest absolute value of the observations for each variable.

The observation operator H is configured to select observations for each variable at each point of the grid, hence we say the observations are dense in space. In an operational setting, the operator H is enforced by the availability of observations provided by meteorological stations and other types of sensors.

In order to minimize the 4D-Var cost function, the L-BFGS-B iterative solver will be employed [13]. The optimization parameters of interest were assigned the following values:

- The stopping criterion is set to $\left\|\nabla \mathcal{J}(\mathbf{x}^0_{[k]})\right\| < 10^{-6} \cdot \max(1, \left\|\mathbf{x}^0_{[k]}\right\|)$.
- Wolfe conditions parameters are set to $c_1 = 10^{-4}$ and $c_2 = 0.9$.
- A maximum number of 500 iterations is allowed for each optimization.

6 Computing Observation Impact

6.1 Second-Order Information

The matrix of coefficients is the Hessian of the 4D-Var cost function, evaluated at the analysis. For large-scale models like the atmosphere, this matrix cannot be manipulated in its full explicit form due to high computational demands, storage restrictions and I/O bottlenecks. The dimensions of such Hessian matrices, associated with models of many variables, could reach orders of magnitude of 10^6 or more. In practice, one usually tries to evaluate directly the action of the Hessian matrix on a vector, an approach known as "matrix-free".

Evaluating the Hessian or its action on a vector is itself another computational issue. The methodology of adjoint models provides ways for the computation of first-order and second-order derivative information. Second-order adjoint models are considered to be the best approach to computing Hessian-vector products. Unfortunately, they are not that popular because of the tedious process of deriving them. At the same time, the adjoint models usually have higher computational demands than the forward model, unless they are built around a clever strategy to reuse computation from the forward model [4]. When one does not have access to the second-order adjoint but only to the first-order adjoint, Hessian-vector products can be computed by using finite differences. This approach might be more economical in terms of computation than using second-order adjoints, but is also less reliable.

These two arguments indicated to us that there is only a limited class of methods we can use in order to solve, precondition or transform the system. More specifically, we will use those methods that need only matrix-vector products. By elementary analysis, the Hessian of any reasonable system is symmetric. However, this symmetry can be lost due to numerical inaccuracies. At the same time, we cannot guarantee the Hessian is positive definite when the 4D-Var first-order optimality condition is not met and the resulting analysis is far from the true minimizer. While using iterative methods on our linear systems seems to be feasible, it appears to be significantly more difficult to obtain preconditioners, even simple ones based on the diagonal. Also, we can exclude factorizations or

direct solve methods since these usually require the full matrix or at least a significant portion of it.

In Figs. 1–4 we present some graphical representations of the explicit form of the Hessian, built through the second-order adjoint methodology from matrix-vector products with the e_i unity vectors. For the scenario presented in Sect. 5 the size of the Hessian is 4800 rows by 4800 columns. The first 1600 rows correspond to the variable h, the next 1600 rows to u and the last 1600 to v. The Hessian thus generated proved to be symmetric. Regarding the minimization of the 4D-Var cost function, we let L-BFGS run for as many iterations as needed to obtain a solution close to the optimum (around 500 iterations). As a result, the Hessian was indeed positive definite. Although not shown here, we can confirm that early termination of L-BFGS after 50 or 100 iterations produced an analysis whose corresponding 4D-Var Hessian was no longer positive definite. As said before, the mere purpose of generating the full Hessian is to analyze it offline. In an operational setting it would not be possible, due to constraints imposed by the computational resources.

The first thing noticed about the Hessian is its somewhat regular structure that looks near block-Toeplitz. Although the diagonals are not constant, their values vary inside a well-defined interval. At the same time, we could notice that these diagonals occur at fixed distance from the main diagonal, spaced at every 40 or 1600 rows/columns. These patterns can be seen in Figs. 3 and 4, which are graphical representations of the upper-left corner of the matrix. The patterns repeat themselves across the whole matrix. Since the 4D-Var Hessian is the inverse of the covariance matrix of the analysis [14], we can attribute the patterns in the Hessian to the covariance of each grid point with the grid points in its discretization scheme stencil. Thus, each point of the grid is correlated to the points to the East and the West (corresponding to adjacent rows/columns in the Hessian) but also to the points found to the North and the South (corresponding to rows/columns situated at a distance of 40) and with the other variables at the same points (corresponding to a distance of 1600 rows/columns). This can be verified in Fig. 3 where the diagonal breaks down after 40 cells and is displaced, due to periodic boundary conditions.

The eigenvalues of the Hessian are displayed in Fig. 5, sorted in ascending order. Based on their values, the condition number for the matrix is of $7.2095e{+}3$. The magnitude of the condition number does not indicate the matrix is extremely ill-conditioned, but the eigenvalues are not clustered together and many of them are close to zero.

6.2 Solving the Linear System

Next, we will present our results and comments on solving the linear system with Krylov-based iterative methods when no preconditioner is used to accelerate the convergence. For our experiments we used MATLAB and its built-in linear solvers. We chose three algorithms for our comparison of their efficiency at solving the system. These three methods are "generalized minimum residual" (GMRES), "conjugate gradients" (CG) and "quasi-minimum residual" (QMR).

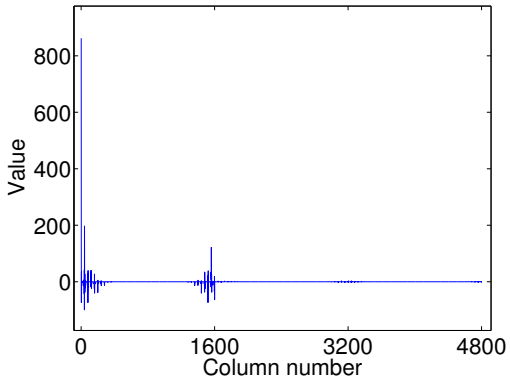

Fig. 1. The values found on the first row of the Hessian matrix

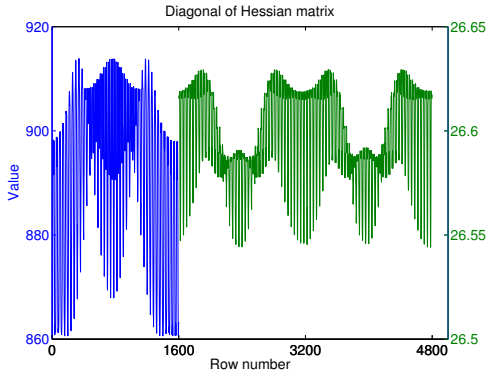

Fig. 2. The values found on the diagonal of the Hessian matrix. (Divided into two scaling groups: first 1600 and last 3200)

There are many other variations of these methods but these three are each representative of one class of such methods. While GMRES and QMR operate on general matrices, CG needs the matrix to be SPD which might not be true all the time for the 4D-Var Hessian.

As we can see from Fig. 6 and Table 2, all three solvers converge to a solution whose norm of the residual is close to zero. GMRES and QMR converge to the same solution and they take the same iterates. This means they are equivalent when applied to this problem. CG converges to a more accurate solution from the perspective of its error norm, but the residual norm is larger. This can cause confusion in practice because we would not have access to the reference solution. Also, the norm of the residual for CG fluctuates, instead of decreasing monotonically. Based on these results, we decided to use GMRES for subsequent tests.

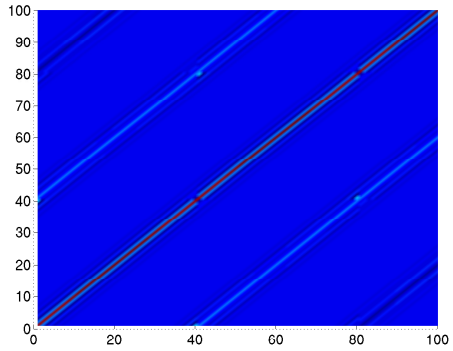

Fig. 3. Density plot of elements in the (100×100 minor) of the Hessian matrix

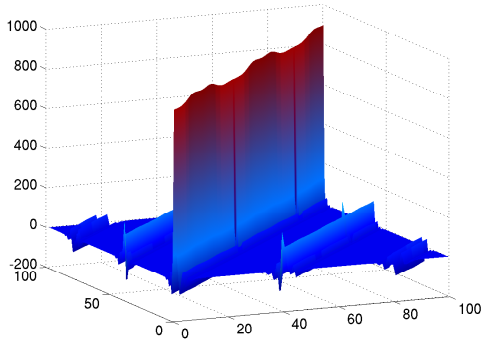

Fig. 4. The values found in the upper-left (100×100 minor) of the Hessian matrix

All solvers start with a vector of zeroes as initial solution.

6.3 Preconditioning

The convergence speed of Krylov-based linear solvers depends on the spectrum of the system matrix. These algorithms perform better when the eigenvalues are clustered together, but as seen in Fig. 5 the eigenvalues of our Hessian matrix are scattered across various orders of magnitude. Also, many of these eigenvalues are close to zero which makes them more difficult to be estimated by the solvers. Such a matrix is called "ill-conditioned" and this explains why neither algorithm presented in the comparison of the previous section converged in 100 iterations to a solution that is highly accurate.

In order to alleviate slow convergence, a linear system can be preconditioned by multiplying it with a matrix M (called preconditioner) that approximates the inverse of the system matrix or some of its features, but is easier to invert

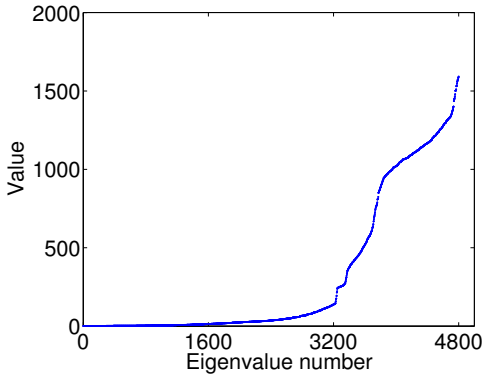

Fig. 5. The computed eigenvalues of the Hessian matrix (in ascending order)

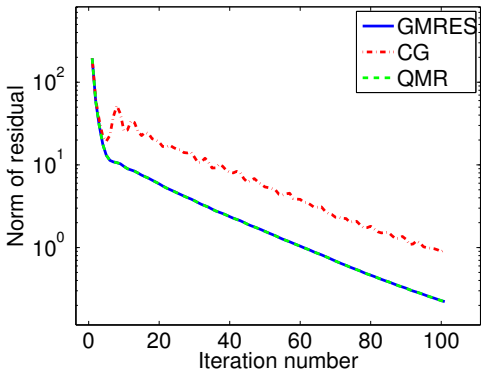

Fig. 6. The evolution of the solution residuals with iteration number for different linear solvers.

or factorize. One way to use this matrix is to multiply the linear system with it to the left, which leads to an equivalent linear system. This system should be easier to solve if the preconditioner approximates the inverse of the system matrix, since their multiplication will yield a new system matrix that is better conditioned. In the ideal setting when the preconditioner is exactly the inverse of the original system matrix, the new system matrix becomes the identity matrix and the linear solver will converge in one iteration. However, finding the exact inverse of a matrix is usually a more difficult problem then solving the linear system.

The problem of finding a preconditioner for our system is the fact that we do not have access to the full system matrix. This means we have to exclude the possibility of using certain preconditioning techniques such as incomplete factorizations, wavelet-based, or variations of Schur complement. Moreover, we

Solver	GMRES	CG	QMR
Norm of residual	0.222	0.893	0.222
Norm of error	0.357	0.267	0.357

Table 2. Norm of residual and error for solutions computed with different iterative methods (100 iterations)

cannot just use matrix-vector products to construct basic preconditioners such as the main diagonal or a diagonal band, unless we construct the full matrix.

Knowing or anticipating the structure of the matrix can be of great help when trying to devise a preconditioner. For instance, we know our Hessian matrix is the inverse of the covariance matrix of the 4D-Var analysis. If the data assimilation process is accurate enough, one can observe a near block-diagonal structure corresponding to each variable. On a 3D grid these blocks correspond to the vertical levels of each variable. This was noted by Zupanski in [15] who hinted at the possibility to approximate the diagonal in a block-wise fashion, where the diagonal values for a certain block are equal to a constant value. We will try to approximate the diagonal by using Hessian-vector products to "probe" the matrix. A straightforward way of accomplishing this for our three-variable model is to run three Hessian-vector products with unity vectors that extract one column (row) of the Hessian at one time and then use the value corresponding to the diagonal element for all diagonal elements in that block. For example, consider three unity vectors for our 4800 Hessian that have the value 1 at position 1, 1601 and 3201 respectively, and zeros everywhere else. The Hessian-vector product realized with these three vectors will extract the columns number 1, 1601 and 3201 which correspond to the three different variables in our Hessian structure. Building an approximation of the diagonal means using the value found at coordinates 1,1 for the entire first block (so up to coordinates 1600,1600), the value found at coordinates 1601,1601 for the entire second block and so forth. This approximation can be refined by probing for more elements from the same block. If there are many blocks that have to be probed and the computational burden increases significantly, one can employ coloring techniques to probe for more than one element with the same matrix-vector product. We shall refer to this technique as "block-diagonal preconditioner".

Another characteristic of covariance matrices is their diagonally dominant structures and under certain favorable cases, strongly diagonally dominant. In this former case, an approximation of the diagonal can be computed with just one matrix-vector product, where the seed vector is composed only of ones. This is equivalent to computing the sum of each row, which is a good approximation of the diagonal if the diagonal element supersedes the sum of all other elements from the same row. This is the second preconditioner we tried in our experiments.

The Hessian matrix can also be approximated from data collected throughout the minimization process of 4D-Var. Iterative solvers such as L-BFGS work by generating approximations of the Hessian evaluated at the analysis in a certain

Preconditioner	None	Diagonal	Block-diagonal	Row sum	L-BFGS
Norm of residual	0.222	0.009	0.009	62.346	0.222
Norm of error	0.357	0.005	0.005	5.301	0.339

Table 3. Norm of residual and error for solutions computed with different preconditioners (100 iterations)

subspace and minimize the cost function upon it. This approximation is very efficient as it preserves the properties of the matrix (positive definiteness and symmetry) and because it is designed for limited storage. We built a preconditioner by reusing the approximation of the Hessian generated over the last 10 iterations of L-BFGS. Our tests showed this to be as accurate as if using more iterations, a result that confirms the theory.

The norm of the error against the reference solution and that of the residual are shown in Table 3. We also present the results obtained with the exact diagonal, although this will not be available in practice, as stated before. The preconditioner obtained from the sum of elements on each row did not improve the convergence at all. This is due to the fact that in our case, the Hessian was not strongly diagonally dominant so this preconditioner was far from approximating the diagonal. The L-BFGS preconditioner brought a slight improvement in the solution and has the advantage that no extra computation is required. The block-diagonal preconditioner improved significantly the accuracy of the solution and behaved as good as the exact diagonal.

7 Conclusions

This paper presents an adjoint-based framework to compute observation impact in 4D-Var data assimilation. The observation impact calculations need to be performed accurately and rapidly in real-time sensor network deployment problems. The main computational task is solving a linear system whose matrix is the Hessian of the 4D-Var cost function, evaluated at the analysis state. This matrix is typically very large, as each row corresponds to one model state, and it is only accessible through matrix-vector products.

The main contributions of this work are to outline the characteristics of the linear system, to investigate iterative linear solvers, and to propose three efficient preconditioners. Two of the preconditioning methods approximate the diagonal of the Hessian from matrix-vector products, while the third method uses data generated during the 4D-Var minimization to provide a quasi-Newton approximation of the Hessian. Our study shows that these inexpensive preconditioners accelerate convergence. Future work will include the development of improved preconditioners for the linear system, and will study the use of additional iterative methods such as multigrid.

Acknowledgements. This work was supported by National Science Foundation through the awards NSF DMS-0915047, NSF CCF-0635194, NSF CCF-0916493 and NSF OCI-0904397.

References

1. Daley, R.: Atmospheric data analysis. Cambridge University Press (1991)
2. Kalnay, E.: Atmospheric modeling, data assimilation and predictability. Cambridge University Press (2002)
3. Alekseev, A.K., Navon, I.M., Steward, J.L.: Comparison of Advanced Large-scale Minimization Algorithms for the Solution of Inverse Ill-posed Problems. Journal of Optimization Methods & Software 24, 63–87 (2009)
4. Cioaca, A., Alexe, M., Sandu, A.: Second Order Adjoints for Solving PDE-constrained Optimization Problems. Optimization Methods and Software, to appear (2011)
5. Baker, N.L., Langland, R.H.: Diagnostics for evaluating the impact of satellite observations. In: Park, S.K, Xu, L. (eds.) Data Assimilation for Atmospheric, Oceanic and Hydrologic Applications, pp. 177-196. Springer-Verlag, Berlin (2009)
6. Baker, N.L., Daley, R.: Observation and Background Adjoint Sensitivity in the Adaptive Observation-targeting Problem. Q.J.R. Meteorol. Soc. 126: 1431–1454 (2000)
7. Daescu, D.N.: On the Sensitivity Equations of Four-dimensional Variational (4D-Var) Data Assimilation. Monthly Weather Review 136 (8), 3050–3065 (2008)
8. Daescu, D.N., Todling, R.: Adjoint Sensitivity of the Model Forecast to Data Assimilation System Error Covariance Parameters. Quarterly Journal of the Royal Meteorological Society 136, 2000–2012 (2010)
9. Liska, R., Wendroff, B.: Composite Schemes for Conservation Laws. SIAM Journal of Numerical Analysis 35 (6), 2250–2271 (1998)
10. Giering, R., Kaminski, Th.: Recipes for Adjoint Code Construction. ACM Transactions On Mathematical Software 24 (4), 437–474 (1998)
11. Giering, R.: Tangent Linear and Adjoint Model Compiler, Users Manual 1.4. `http://www.autodiff.com/tamc` (1999)
12. Griewank, A.: On Automatic Differentiation. In: Iri, M., Tanabe, K. (eds.) Mathematical Programming: Recent Developments and Applications, pp. 83–109. Kluwer Academic Publishers (1989)
13. Zhu, C., Byrd, R.H., Lu, P., Nocedal, J.: L-BFGS-B: A Limited Memory FORTRAN Code for Solving Bound Constrained Optimization Problems. Technical Report, NAM-11, EECS Department, Northwestern University (1994)
14. Gejadze, I., Le Dimet, F.-X., Shutyaev, V.: On Error Covariances in Variational Data Assimilation. Russian Journal of Numerical Analysis and Mathematical Modeling 22 (2) 163–175 (2007)
15. Zupanski, M.: A Preconditioning Algorithm for Large-scale Minimization Problems. Tellus (45A) 478–492 (1993)

DISCUSSION

Speaker: Adrian Sandu

Van Snyder : There are two possibilities to assimilate remote sensing data into chemical transport models (CTMs). One is to incorporate the measured quantity, e.g., radiances. The other is to incorporate geophysical quantities that result from separate analyses. The latter is a nonlinear parameter estimation problem, which depends upon having a good starting point. Further, the problem is ill-posed, which requires *a priori* information or Tikhonov for stabilization. If the CTM initializes and stabilizes the remote sensing problem, whose results are then assimilated into the CTM, is there a positive, negative, or neutral effect on the solution quality?

Adrian Sandu : This is a very good question. In principle the direct assimilation of measured quantities is to be preferred, since it does not introduce additional errors/biases in the data. In order to do this, one needs good models of the instrument, e.g., that produce radiances from the concentration fields computed by the CTM.

In practice, there are very many instruments that produce data, and for which good models are not available. For this reason assimilation of geophysical quantities, derived from the raw data through an off line estimation process, is often employed. To the best of my knowledge no comprehensive tests have been carried out to date in order to quantify the impact of assimilating geophysical quantities in lieu of measured quantities.