CIRCUMVENTING STORAGE LIMITATIONS IN VARIATIONAL DATA ASSIMILATION STUDIES

JUAN MARIO RESTREPO†
GARY K. LEAF†
ANDREAS GRIEWANK‡

†Mathematics and Computer Science Division
Argonne, National Laboratory, Argonne, IL 60439 U.S.A.
‡Institute of Scientific Computing,
Mathematics Department, Technical University of Dresden,
Mommsenstr. 13, 01062 Dresden, Germany

ABSTRACT. The aim of data assimilation is to infer the state of a system from a geophysical model and possibly incomplete or nonuniformly distributed spatiotemporal observational data. Used extensively in engineering control theory applications, data assimilation has relatively recently been introduced into meteorological forecasting, natural-resource recovery modeling, and climate dynamics.

Variational data assimilation is a promising assimilation technique in which it is assumed that the state of the system is an extrema of a carefully chosen objective function. Provided that an adjoint model is available, the required model gradients can be computed by integrating the model forward and its adjoint backward. The gradients are then used to extremize the cost function with a suitable iterative or conjugate gradient solver.

The problem we address in this study is the explosive growth in both on-line computer memory and remote storage requirements of large-scale assimilation studies. This imposes a severe physical limitation on the size of assimilation studies, even on the largest computers. By using a recursive strategy, a schedule can be constructed that enables the forward/adjoint model runs to be performed in such a way that storage requirements can be traded for longer computational times. This generally applicable strategy enables data assimilation studies on significantly larger domains than would otherwise be possible given particular hardware constraints. We show that this tradeoff is indeed viable and that when the schedule is optimized, the storage and computational times grow at most logarithmically.

1991 *Mathematics Subject Classification.* 86A05, 86A10, 86A22, 86-04, 86-08.

Key words and phrases. variational data assimilation, adjoint model, climate, oil recovery, meteorology, recursion, storage.

The authors thank Eli Tziperman for making his climate code available to us. J.M.R. became aware of this data assimilation problem in the course of many stimulating hours of conversation with Jochem Marotzke at MIT and Kirk Bryan at GFDL; their patience and their hospitality are much appreciated. This research was supported in part by an appointment to the Distinguished Postdoctoral Research Program sponsored by the U.S. Department of Energy, Office of University and Science Education Programs, and administered by the Oak Ridge Institute for Science and Education. Further support was provided by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

Typeset by AMS-TEX

[1]. INTRODUCTION

Data assimilation has relatively recently become an important tool in many areas of geophysics, such as weather and climate forecasting [1–6], model sensitivity analysis [7, 8], and in the inclusion of field data sets into theoretical model-studies [9–11]. In weather forecasting, field data that may be spatially and/or temporally heteregeneous is continuously blended into dynamical models as soon as the field data is available. As a result, the predictive capabilities of today's weather models have significantly improved [1,12]. Ocean forecasting has, on the other hand, not experienced comparable success. Reasons for this are that (1) the spatial and temporal scales of the relevant oceanic dynamics are several orders of magnitude smaller and larger, respectively; (2) oceanic data gathering is at present very limited in coverage and sometimes of incompatible quality; (3) boundary fluxes at the air/sea interface are poorly understood and yet have a major influence on oceanic flows; and (4) the computing demands of oceanic forecasting have only recently become marginally suitable for some but not all of the types of studies at reasonable resolutions.

A specific approach to data assimilation is called variational data assimilation [12]. An objective function is defined that provides a norm of the distance or misfit of the state set to observational data. The state set may comprise model predictions, parameters, boundary data, and/or initial conditions. The misfit is usually weighted in order to account for measurement errors, model uncertainties, etc. The object is to find the state set that extremizes the objective function. This procedure is usually carried out as a constrained optimization problem, which is generally solved iteratively by some extention of Newton's method or a descent algorithm.

The optimization problem requires the computation of the gradient of the model with respect to the state set. One of the other strategies that accomplishes the calculation of the gradient is the "adjoint method" [3]. Provided an adjoint to the tangent linear model exists, the process of computing the gradient involves integrating the original model forward in time (the forward problem) recording the model's history, and then using the history in the adjoint model to integrate backward in time to the point of origin (the adjoint problem). Along the way the partial differentials that constitute the gradient of the results at some t final with respect to the state set at some particular time step are multiplied in reverse order until the adjoint model reaches the origin once again. By the chain rule, the multiplication will yield the gradient, and it will do so at a computational cost roughly twice that of the forward problem.

As described above, the adjoint method is what we will call the "conventional approach." Its main advantage is its low computational cost. However, its disadvantage is that it quickly encounters computer memory storage problems even in low-resolution studies. In this paper we present an alternative to the conventional approach that circumvents in a significant way the storage problems of the adjoint method at the expense of a possibly greater, but manageable computational expense.

The problem is motivated in Section 2. The alternative gradient method is presented in Section 3 and is compared with the conventional approach. Section 4 demonstrates how such alternative is implemented in practice in an ocean climate problem, and we describe how it compares with the conventional approach in terms of computational effort

and memory usage. Section 5 summarizes our findings, provides details of the strategy's computer implementation, and tells where to obtain code that implements the method.

[2]. Statement of the Problem

For the sake of clarity we will assume that the physical problem in question can be modeled by an evolutionary equation. The physical m-dimensional real domain is $\mathcal{R} \subseteq \mathbf{R}^m$ with boundary $\partial\mathcal{R}$. The evolution equation is discretized in time so that the problem is defined at physical times $t_l = t_{l-1} + \delta t_l$. Without loss of generality we may assume that discrete time progresses in equal-interval steps, hence $\delta t_l = \delta t$ and $t_l = l\delta t$. In the discrete forward/adjoint method of computing a gradient the state set is required at periodic time intervals of time. The state set is computed using the evolution equation, which, for simplicity, will be assumed to be computed at equally-spaced intervals of time Δt. In most instances $\delta t \leq \Delta t$. We define the time-index $i \in \mathcal{T} \subset \mathbb{Z}^+$ so that $i\Delta t = l\delta t$. The semi-discretized "forward problem" is defined as

$$u_i = F_i(\{u_j\}) \qquad i = 1..n,\ 0 \leq j \leq i \tag{2.1}$$

$$u_0 = U,$$

$$u_i|_{\partial\mathcal{R}} = V_i, \tag{2.2}$$

where the completely or partially unknown U and V are respectively the initial and boundary data for the state set that minimize an objective function. The "reverse problem" is the adjoint of (2.1),

$$u_i^* = F_i^*(\{u_k^*\}, \{u_j\}) \qquad i = n..0,\ i \leq k \leq n,\ j \in [0, n]. \tag{2.3}$$

If the forward problem is a semi-discretization of an evolution equation, we think of u_i and u_i^* with domain $\mathcal{R} \times \mathcal{T}$ as vectors of the state variables and their adjoints.

Equations (2.1) and (2.3) will be solved in some high-level computer language such as Fortran or C. Define $S = \cup_j s_j$ and $S^* = \cup_k s_k^*$ as the set of computer memory addresses required to represent the vector set $\{u\}$ and $\{u^*\}$ at index location i, so that u_j and u_k^* have temporary memory locations s_j and s_k^*, respectively. It is assumed that $s_j \cap s_k = \emptyset$, $s_j^* \cap s_k^* = \emptyset$, and $s_j \cap s_k^* = \emptyset$. We call this temporary computer storage medium the "register".

Let f and f^* be the representations of F and F^*, respectively in some high level computer program, or "program" for short. These take the form of subroutines, functions, etc. The action of $f : S \to S$ and $f^* : S^* \to S^*$. Define the m- and t- norms as the memory and time of execution of some program Q as $\|Q\|_m$ and $\|Q\|_t$, respectively. As will be evident in what follows, these norms amount to simple direct sums. The register memory of the state set is $\|S\|_m = R$, and it is safe to assume that $\|S^*\|_m \leq R$. The other type of memory that will play an important role in the analysis is the available memory external to the program. This is usually some external storage device such as a memory

disk or tape. For simplicity we call this recording device the "tape" and assume that it has fixed memory of size T. The specific use of the term "writing" will be reserved for the process of recording to tape. Similarly, the term "reading" is reserved for the process of accessing information from tape. The distinction between a non-reading or non-writing program procedure f_i and the same procedure that reads or writes the state set on tape will be indicated as $\hat{f}_i$. It will be convenient to define the following specific m- and t-norms:

$$\begin{aligned} \mu &= \max_{0 \le i \le n} \|f_i\|_m \\ \tau &= \max_{0 \le i \le n} \|f_i\|_t, \end{aligned} \tag{2.4}$$

respectively, the maximum memory required to restore f_i given S and the maximum computing time (wall-clock time) to execute f_i. It is worth noting that μ is essentially fixed regardless of the number of processors, while τ can vary significantly depending on the number of processors. Since f^* is a linear mapping on S^*, it can be assumed that $\tau^* \le \tau$ and

$$\tau^* \le c\hat{\tau}^* \le c'\hat{\tau},$$

where τ^* and $\hat{\tau}$ refer respectively to analogous norms to (2.4) of f^* and $\hat{f}$, and the c's are positive multiplicative constants. Note that $\|f_i\|_t \ge R$, since the subroutines may require working registers.

In the discretization and coding of a typical evolution equation (for example, of a climate or meteorology problem) we can identify f_i as the collection of subroutines and functions that take the state set from time t_i to t_{i+1} (forward integration) in which $\|f_i\|_m$ and $\|f_i\|_t$ are approximately the same for each level $0 < i \le n$ and thus equal to μ and τ, respectively. In the same fashion f_i^* is the collection of subroutines that take the state set from time t_i to t_{i-1} (reverse integration) in which $\|f_i^*\|_m$ and $\|f_i^*\|_t$ are approximately the same for each level $0 < i \le n$ and thus equal to μ^* and τ^* respectively. Let us consider the memory and the time norms of two strategies that may be used in the $n-$step gradient computation by the adjoint method.

In one strategy the minimal memory norm is achieved by writing nothing on tape. It requires stepping forward from u_0 to u_n using f_i, followed by a single reverse step from u_n to u_{n-1} using f_n^*. The process starts again from u_0 forward to u_{n-1} using f_i followed by a reverse f_{n-1}^* from u_{n-1} to u_{n-2}. This process is repeated until the reverse integration reaches step 0 once again. The t- and m- norms for this strategy are respectively

$$\begin{aligned} \Upsilon + \Upsilon^* &= \frac{(n-1)n}{2}\tau + n\tau^* \le \frac{(n+1)n}{2}\hat{\tau} \\ \|S\|_m + \|S^*\|_m &= 2R, \end{aligned} \tag{2.5}$$

where only register memory is used. For simplicity we are ignoring here, as we will do from now on, the register memory that is used for working arrays, etc. For an explicit fourth-order Runge-Kutta time integration scheme, for example, this register memory can be significant but can be easily accounted in the estimates provided.

Another strategy is the conventional approach, which steps forward from u_0 to u_n using $\hat{f}_i$, then steps in reverse using f_i^*, reading the appropriate state variables from tape. The time and memory norms for the latter strategy are

$$\hat{\Upsilon} + \hat{\Upsilon}^* = n\hat{\tau} + n\hat{\tau}^* \leq 2n\hat{\tau}$$

$$\|\hat{S}\|_m + \|S\|_m + \|S^*\|_m = nR + 2R = T + 2R. \tag{2.6}$$

Hence the conventional approach yields the adjoint as a fixed multiple of the time for the forward program. However, the tape grows linearly in both number of steps and size of the state set, which for typical geophysical applications will quickly overwhelm even the largest storage capabilities of computer facilities [13].

[3]. Recursive Adjoint Method

The recursive strategy or "schedule" is specifically designed to circumvent the storage limitations of the conventional adjoint method at the expense of a larger computational effort. The computational effort will be defined more precisely below, but for now it suffices to know that the computational effort is directly proportional to the wall-clock time, which in turn depends on the number of processors. One strategy that reduces the tape size is to produce the gradient by using the usual forward/adjoint sweep but writing less often than is really required. While this alternative saves some tape space, it produces a degraded gradient. It will be shown below that the gradient produced by the recursive method will be identical to its nondegraded counterpart obtained in the conventional way.

The description that follows will present a heuristic explanation of the theoretical development that appears in [14]. The basis of this strategy is to limit the tape size to dR, , where $d \leq n$ snapshots (snaps, for short) of states $\{u\}$ at any given point during the program execution. This is done by carefully overwriting. It requires at most an additional r-fold increase in additional full forward unrecorded computations, or "reps". The recursive strategy is not unique. However, from Theorem 6.1 due to Griewank [14], among the partitioning algorithms the "binomial partitioning" schedule is optimal. The theorem states that an n-step gradient calculation with the adjoint method can be solved recursively by using up to $d \geq 0$ snaps and at most $r \geq 0$ reps if and only if

$$n \leq n(d,r) = \frac{(d+r)!}{d!r!}. \tag{3.1}$$

Note that $n(d,r) = n(r,d)$ and $n(0,r) = n(d,0) = 1$. To illustrate the sense in which this method is superior we appeal to Stirling's formula and find that for a fixed d or r,

$$r = O(n^{1/d})$$

or

$$d = O(n^{1/r}). \tag{3.2}$$

To see more clearly the relationship between n and the number of snaps and reps, a contour plot of $\ln n$ as a function of the number of snaps and reps based on (3.1) we present

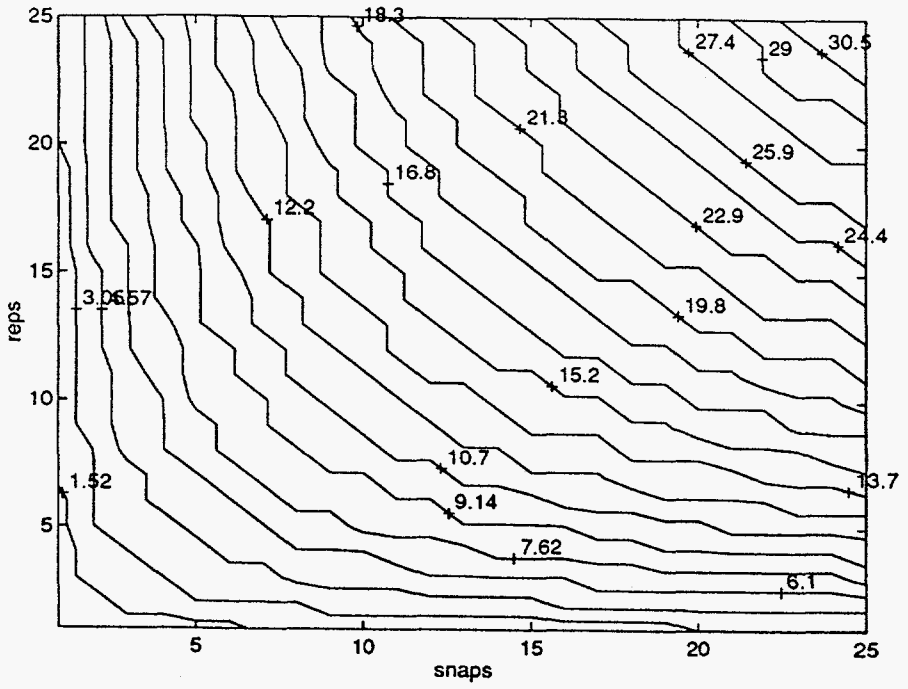

FIGURE 1. Contours of $\ln n$ versus snaps d and reps r.

Figure 1. Since the values that the binomial takes are discrete, the contours appear jagged. The figure clearly illustrates the logarithmic rate of growth of n when $d \approx r$. In fact, when $d = r$ these grow as $\log_4 n$.

The schedule for $n = 56$, $r = 5$ reps, and $d = 3$ snaps appears in Figure 2 and is worth explaining in some detail. Note that

$$n = 56 = \binom{r+s}{s}.$$

Along the horizontal is the number of reps, and along the vertical the time step i. The tree structure of the schedule is evident. Horizontal lines are drawn at locations in which writing is performed. As is evident, when reading the figure from left to right, there are five self-similar groups or pennants. The top pennant and the first to be executed has three snaps at $i = 0, 35$, and 50. A write occurs at $35 = \binom{r-1+s}{s}$ and the write at time step $50 = 55 - r$. Execution requires a forward sweep from $i = 0$ to 56. The state at 50 is restored once more, and a forward sweep to 55 follows. A forward/adjoint from 55 to 56 and back again to 55 is executed then. The first pennant is completely swept by repeating the last two steps until the adjoint reaches 50. State 35 is then restored and a forward sweep follows, writing at $45 = 49 - (r - 1)$. After the second uppermost triangle is swept through, state 35 is recovered, and a forward sweep follows, writing at $41 = 44 - (r - 2)$. After completing the first pennant, state 0 is restored, and a forward sweep is initiated that ends at $35 = \binom{r-1+s}{s}$, writing along the way at $\binom{r-2+s}{s}$. At this point, the schedule should be obvious. the last pennant is performed when $\binom{r-5+s}{s} = 1$. Note that at no instant will the depth of the tape be more than three records long. In addition, if the tape is thought of

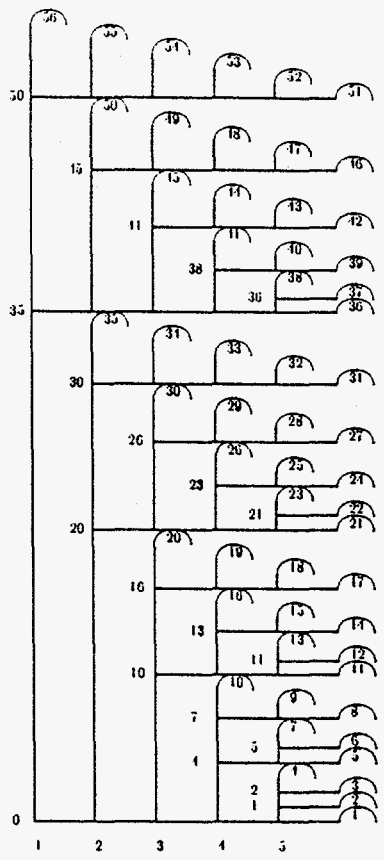

FIGURE 2. Schedule for $n = 56$, $r = 5$ reps, and $d = 3$ snaps.

as a stack, the order of the records is maintained, as a result of its last-in-first-out nature. It is evident from the figure that there are a total of 1 forward recorded sweep, 1 adjoint reverse sweep, and r forward unrecorded sweeps.

From Figure 2 it may be concluded that the t-norm and m-norm of the recursive schedule are, respectively,

$$D_t = \hat{\Upsilon} + \hat{\Upsilon}^* + r\Upsilon \leq (2 + r)n\hat{\tau}, \tag{3.3}$$

$$D_m = T + 2R = (d + 2)R, \tag{3.4}$$

since $T = dR$. The first expression on the right-hand side of Equations (3.2) and (3.3) hold generally for any $n(d, r)$, $d \geq 0$, and $r \geq 0$ recursive adjoint problem and the far

right-hand side for any general recursive adjoint problem involving the evolution equation typically encountered in climate or meteorology studies. Also note that if the number of reps r and sweeps d are similar, then

$$\frac{D_m - 2R}{R} \approx \frac{D_t - 2n\hat{\tau}}{\hat{\tau}} \leq \log_4 n.$$

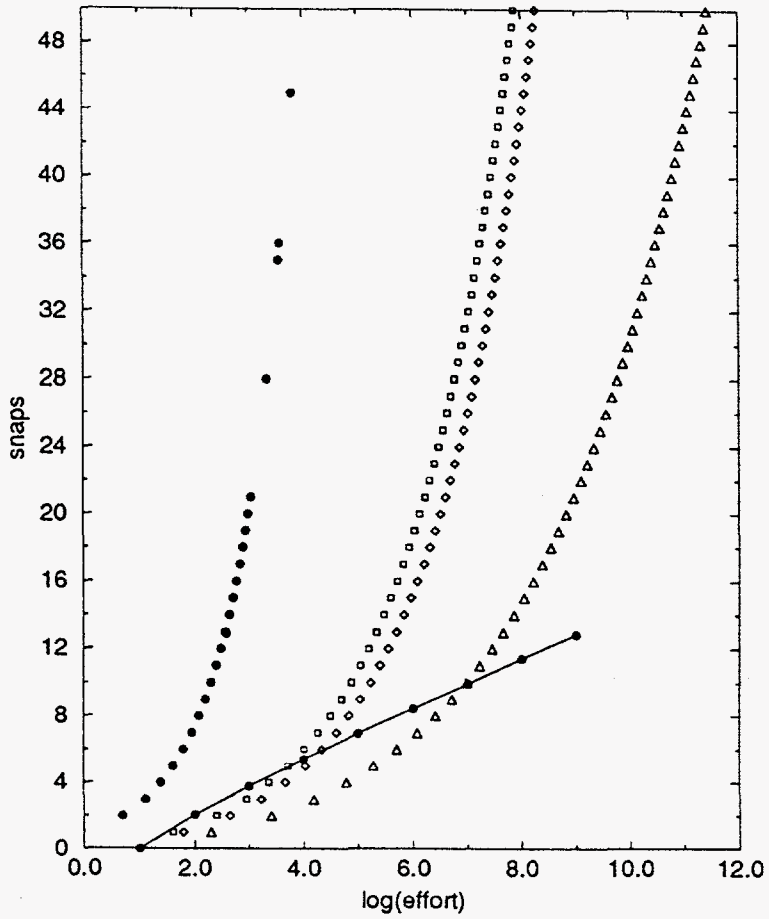

FIGURE 3. Conventional versus Recursive strategy comparison. The added effort due to increased reps r. From left to right, the conventional case, then $r = 1$, $r = 2$, $r = 3$. The curve represented by stars corresponds to $r = d$.

Comparison of (3.2) with (2.6) leads to a working measure of the "computational effort," which is proportional to the wall-clock time: a convenient measure is the total number of forward steps. We shall employ this measure in this and in the following section, in which a comparison between the recursive and the conventional approach is effected. Table 1 shows the schedule characteristics for several values of n, d, and r. From Table 1 confirms some of the particulars of the recursive strategy which have previously been mentioned, such as the fact that the number of reverses and the n is identical. It can also be surmised that the number of reads is one less than the number of reverses because every reverse

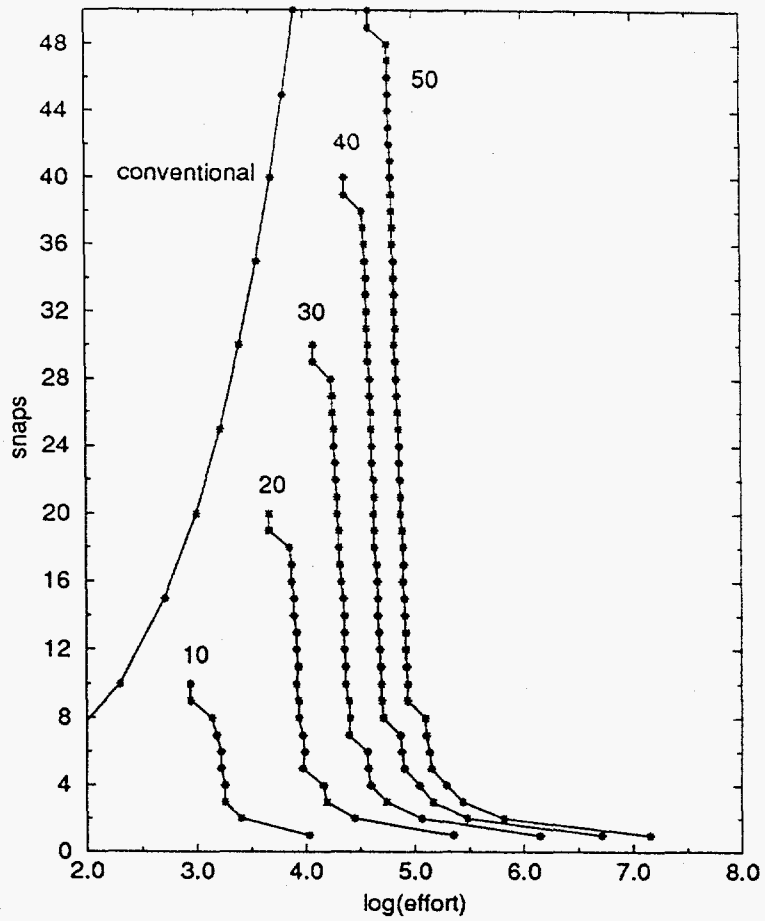

FIGURE 4. Conventional versus Recursive strategy comparison. The points on the conventional curve correspond to n in intervals of 5, the first point being $n = 10$. The other curves are labeled with their corresponding n.

requires a prior read, except for the last reverse. By inspection, the number of writes is $\binom{d-1}{r}$ so that $d/(d+r)$ is the ratio of writes to n.

The performance of the recursive method compared with the conventional one may be assessed graphically. Figure 3 illustrates the relation of the memory, measured in snaps, and the wall-clock time, assuming it is proportional to the effort. The conventional approach is represented by the left-most curve. All other curves represent different snap and rep combinations. In both the conventional and the recursive case, the memory required to solve a problem will be equal to dR, where R is defined as before and depends on the resolution and the number of spatial dimensions in the problem. On the other hand, the effort for the conventional case is basically n, while in the recursive strategy it depends on the choice of snaps and reps. From left to right the recursive strategy curves correspond to decreasing the number of snaps. The line-connected curve in the lower corner corresponds to the case of snaps and reps being equal. The conventional case is, in effect, the limit of snaps d equal to n in the recursive strategy. As can be surmized, the curves reflect the previously mentioned characteristic of the recursive method, namely, that the effort

TABLE 1. Schedule details for several sets of snaps d, reps r, and steps n.

steps	snaps	reps	effort	reverses	reads	writes
252	5	5	1302	252	251	126
126	5	4	546	126	125	70
126	4	5	630	126	125	56
70	4	4	294	70	69	35
56	5	3	196	56	55	35
56	3	5	266	56	55	21
35	4	3	119	35	34	20
35	3	4	140	35	34	15
21	5	2	56	21	20	15
21	2	5	91	21	20	6
20	3	3	65	20	19	10
15	4	2	39	15	14	10
15	2	4	55	15	14	5
10	3	2	25	10	9	6
10	2	3	30	10	9	4
6	5	1	11	6	5	5
6	2	2	14	6	5	3
6	1	5	21	6	5	1
5	4	1	9	5	4	4
5	1	4	15	5	4	1
4	3	1	7	4	3	3
4	1	3	10	4	3	1
3	2	1	5	3	2	2
3	1	2	6	3	2	1
2	1	1	3	2	1	1

increases for the recursive method when fewer snaps are used. Hence, in practice, the user wishes to maximize the number of snaps in the calculation rather than the number of reps. Figure 4 illustrates in greater detail the memory and computational effort dependence on the number of snaps and reps. In this figure it is possible to gauge the relative additional effort required by the recursive strategy over the conventional procedure for a given n. For example, for $n = 50$ the conventional strategy requires 50 snaps and an effort of 3.9, whereas the recursive strategy for the same n requires between 11 and 48 snaps with an effort of about 4.8. Hence, we expect an order of magnitude increase in the wall clock time,

a very reasonable price to pay for the significant savings in terms of tape memory. The non-smooth changes in the curves corresponding to the recursive strategy in the Figure 4 are a result of changing the value of the rep count. A comparison of Figure 4 and Figure 3 bears this conclusion. recursive curves

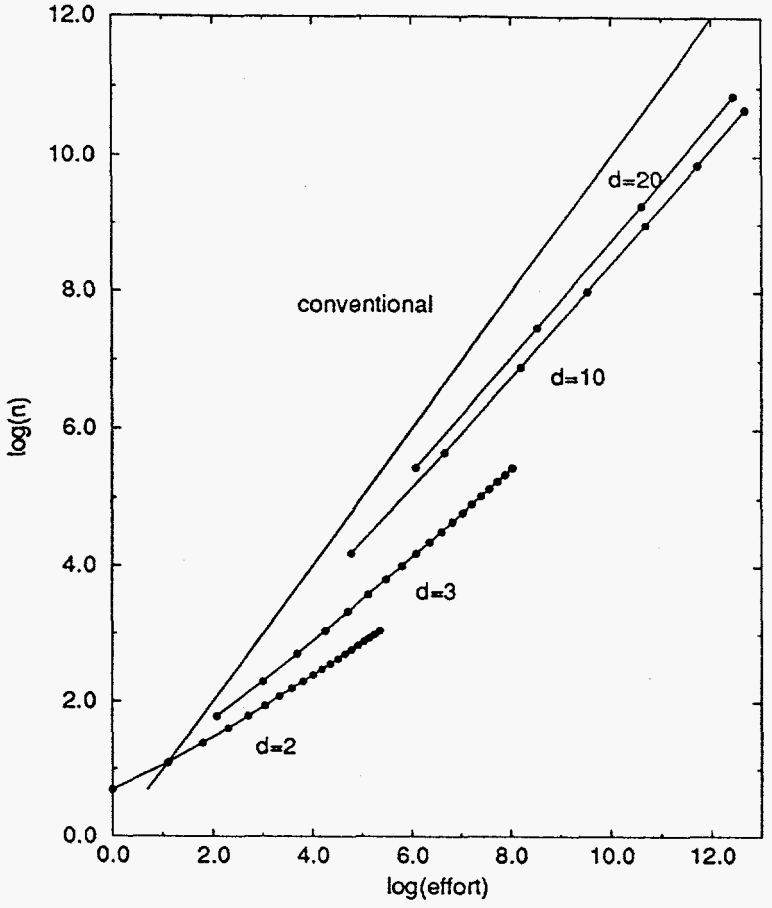

FIGURE 5. Comparison of the conventional and recursive strategy. The memory requirement of the conventional case is n. The recursive curves are labeled according to the number of snaps d used. Natural logarithms are used.

Figure 5 shows a comparison of the conventional strategy (the left-most solid curve) with the recursive strategy with regards to the effort given by n. The finite extent of the lines joining the points as well as the density of points per curve is a result of the way in which the graph was generated: the maximum number of snaps and reps was limited to 20. Bounding the snaps and reps this way limits the number of points belonging to each line and the density of points corresponding to $d = 2$, say, is much greater than the number of points corresponding to $d = 20$. The slope of the recursive curves gets closer to the slope of the conventional case the more snaps are used. Note that in the conventional case the number of snaps is equal to n. Hence, this figure shows the clear advantage of the recursive method with regard to memory. Specifically, whereas an increase in n in the

conventional case leads to an increase in tape usage, the recursive strategy enables the user to consider a wider range of n for a fixed tape size dR. The feasibility of this latter strategy is dictated by the speed of the machine or the willingness to pay for the higher effort involved. Compare this to the previous figure, which shows the price paid in higher wall-clock times as a result of the smaller number of snaps employed. It may be that the effort required in large problems is significant, but this must be weighed against the fact that these problems may be simply impossible to consider with the conventional strategy.

[4]. Application to a Quasi-Geostrophic Ocean Problem

The recursive procedure's viability will be demonstrated by applying it to a quasi-geostrophic model [**15**] [**16**], which was considered in Tziperman and Thacker's study [**13**], hereafter referred to as T&T. The dimensionless equations over a unit-square box in x and y are

$$\begin{aligned} \zeta_t + \psi_x + RJ(\psi,\zeta) &= -\epsilon_b\zeta + \epsilon_h\Delta\zeta + \mathrm{curl}\tau \\ \zeta &= \Delta\psi, \end{aligned} \tag{4.1}$$

where $\psi(x,y,t)$ and $\zeta(x,y,t)$ are the streamfunction and the vorticity, $\tau(x,y)$ is the wind stress, $J(\cdot,\cdot)$ is the Jacobian of its arguments, and Δ is the Laplacian operator. The dimensionless real parameters R, ϵ_b, and ϵ_h are the Rossby number, the bottom friction factor, and the horizontal friction factor, respectively. The state variables evolve in time t and are subject to no-flux and no-stress boundary conditions at the edges of the box.

The equations were discretized using multigrid finite-difference techniques. In what follows it will be understood that the state variables are defined only on the uniformly discretized grid in x and y. For the sake of clarity w e will omit explicit mention that these quantities are discretized in space. On a discrete time grid $t = i\Delta t$, the state variables ζ^i and ψ^i evolve to a steady state $\widetilde{\zeta}$ and $\widetilde{\psi}$. Following [**13**], an assimilation problem is defined as follows. The observational data will be the steady-state vorticity $\widetilde{\zeta}$, which is independent of time. The state set is taken to be the forcing term curlτ, the initial vorticity ζ^0, and the parameters ϵ_b and ϵ_h. The observations $\widetilde{\zeta}$ are determined from a particular (fixed) choice of friction factors $\widetilde{\epsilon}_b$ and $\widetilde{\epsilon}_h$, initial vorticity ζ^0 and forcing curl$\widetilde{\tau}$. The system is then integrated forward in time until a steady state is reached, at which point the observations are written. For purposes of this artificial assimilation problem, we now "forget" the state set values which produced the observations. The task of the assimilation will then be to reconstruct the state set that generated the observations. To this end, a cost function is chosen that measures the fit of the model result to the observations. Since the observations represent the steady state, the cost function should measure the departure of the model from steady state as well as the departure from the observations. In [**13**] the authors use the following discrete cost function:

$$\mathbf{H}^n(\mathrm{curl}\tau, \zeta^0, \epsilon_b, \epsilon_h) = \sum \left[C(\zeta^0 - \widehat{\zeta})^2 + D(\zeta^n - \zeta^0)^2 \right],$$

where the sum indicates a sum over all the discrete values of the variables over the unit box. The first term measures the deviation from the observations, while the second term

in conjunction with the first measures the deviation from steady state. The matrices C and D are the inverse of the covariance matrices of the observations. The final time step, n, is arbitrary in this problem. It is chosen to be sufficiently large so that steady state is achieved. A small value of n reduces the computational cost per optimization iteration; however, it increases the number of optimization iterations. Since the number of written histories depends on the number of time steps n, the storage requirements are reduced when n is small.

The optimization task is to find the state set $\{\text{curl}\tau, \zeta^0, \epsilon_b, \epsilon_h\}$ for which $\mathbf{H}^n$ is a minimum subject to the constraints of the model equations. A common strategy for computing the minimum is to introduce Lagrange multipliers and the corresponding Lagrange functions for which we seek an unconstrained extremum. A gradient-based iterative algorithm such as the conjugate gradient method is then applied to this unconstrained problem. For the discrete quasi-geostrophic model, the Lagrange function has the form

$$\mathbf{L}^n = \mathbf{H}^n + \sum\sum_{i=0}^{n} \mu^i[\zeta^i - \Delta\psi^i] + \sum\sum_{i=1}^{n} \lambda^i \left\{ \frac{\partial \zeta^i}{\partial t} + \frac{\partial \psi^{i-1}}{\partial x} + RJ(\psi^i, \zeta^i) + \epsilon_b \zeta^{i-1} - \epsilon_h \Delta\zeta^{i-1} - \text{curl}\tau \right\}.$$

The descent algorithm requires the calculation of the gradient of $\mathbf{L}^n$ with respect to the state set. The gradient involves the Lagrange multipliers $\{\mu^i, \lambda^i\}$, which are determined from the gradients of $\mathbf{L}^n$ with respect to $\{\zeta^i, \psi^i\}$. Equating these gradients to zero generates the adjoint equations for $\{\mu^i, \lambda^i\}$, which may be symbolically expressed as

$$\begin{aligned} \mu_t + \lambda_x + R[J(\lambda, \Delta\psi) - \Delta J(\lambda, \psi)] &= \epsilon_b \mu - \epsilon_h \Delta\mu + \Psi, \\ \Delta\lambda &= \mu, \end{aligned} \tag{4.2}$$

where Ψ is the forcing term arising from the gradients of the cost function with respect to $\{\zeta^i, \psi^i\}$. The discrete adjoint equations are integrated backward in time to generate the Lagrange multipliers λ^i used in computing the gradients of the cost function as needed by the conjugate gradient procedure. Thus, in the conventional approach, each conjugate gradient iteration requires a forward integration of n steps, which generates the value of the cost function, followed by a backward integration of the adjoint equations. This adjoint integration generates the gradients used in the conjugate gradient iteration. Observe that the state set is required to effect the calculation of the Lagrange multipliers from the adjoint equations. Thus, in the conventional approach involving n time steps, n state sets have to be saved. Since only the state variables are time dependent in this particular problem, we need only to write the state variables ζ^i, ψ^i at each time step. The remaining components of the state set need to be written only once during the forward-backward sweep. The observations were synthesized by running the discretized version of (4.1) to steady-state using $\text{curl}\tau = -\sin(\pi x)\sin(\pi y)$, $\epsilon_b = 0.05$, $\epsilon_h = 0.0001$, and $R = 0.01$.

To demonstrate the performance of the recursive forward-backward integration strategy for the calculation of the gradient, we compared model runs of this experiment using the original multigrid Fortran code against a version of the code which was identical in all

respects to T&T's code, except for a subroutine that generates the schedule and for minor modifications to the program to enable us to implement the schedule. As a first step, we verified that our program results yielded identical results to the conventional case. The wall-clock time was negligibly higher for the recursive program running in the conventional mode, reflecting the additional computational expense of generating the schedule.

In the experiments to be reported, the optimality tolerance for the NAG conjugate gradient routine was set to 10^{-3} in all model runs. The square of $\psi^i - \psi^{i-1}$ summed over the box was used as the error tolerance in the conjugate-gradient calculation. The forward run used to create the observations stepped in time until the residual was below 10^{-7}. The multigrid depth was fixed at four levels for all experiments that follow. The codes were executed on a Sparc 10/51 running SunOS 4.1.3U1. The Fortran Sun compiler used was Fortran Version 1.4 with optimization flags turned off. All runs were performed in double-precision arithmetic. Wall-clock times reported encompass the solution to the full problem. In all experiments performed, the answers from both strategies were identical.

In T&T's study, $n = 1$. In their experiment such a choice is possible since the assimilation occurs at just one time level. The role of the integration time length in connection to T&T's problem was investigated by Marotzke [9], where he concluded that in this quasi-geostrophic model, advective phenomena would not adjust quickly enough. He suggested that the assimilation be carried out over longer time spans. Hence there is some flexibility in choosing the integration time, since the only requirement is that it must be longer than n^*, where n^* is the minimum number of steps for a steady-state solution. In the general case, assimilations may occur at multiple time levels, in which case the number of time steps used is determined by the problem and cannot be arbitrarily chosen.

Suppose that for a particular resolution the problem "fits" and thus can be solved on a particular machine using the conventional approach. In order to double the spatial resolution, the conventional strategy would require a sixteenfold increase in tape storage: fourfold due to the increase in resolution, and fourfold for the increase in the number of time steps. The doubly resolved experiment no longer could be performed on this particular machine. However, the problem could be solved by using the recursive approach as long as the maximum tape length was not exceeded. Suppose that the maximum tape length on this machines is 100000 floats. The requirement of the singly resolved T&T problem with $n = 56$ and a 32×32 spatial grid with four refinement levels is 60984 floats. Table 2 provides the results of a couple of runs using the recursive strategy for the doubly-resolved problem. Supposing that the conventional procedure could be could be carried out, for $n = 224$, the tape length for the doubly-resolved problem would be 946400 floats and it would have taken 153.56 seconds to execute. The table demonstrates that the doubly-resolved problem can be succesfully carried out in approximately twice the amount of time that it would take to run the conventional procedure assuming that it could be possible to compute conventionally in the first place.

A different situation in which tape length is a limiting factor in assimilation studies arises when the integration times are very long, causing the state set history stored on tape to be extremely large. Figure 6 shows a comparison of tape usage for the conventional and the recursive strategy. In the recursive trials the snap count was held fixed at five, explaining why its curve for tape usage is a vertical straight line. As mentioned previously, for the

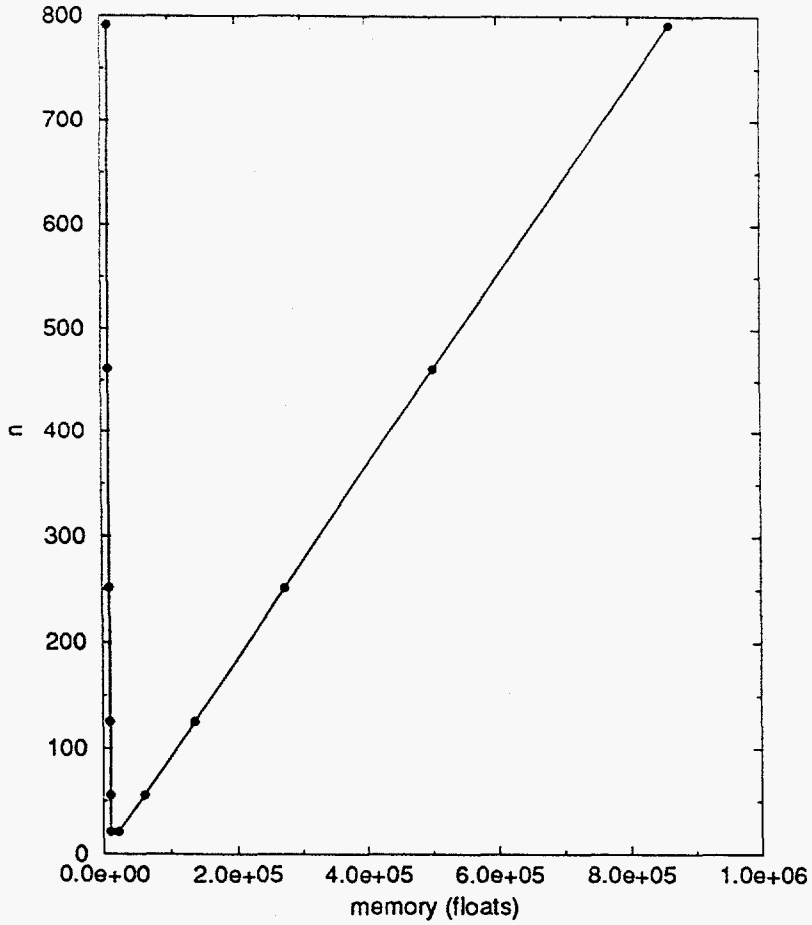

FIGURE 6. Comparison of the conventional and recursive strategy on the T&T problem. In the recursive strategy the snap count was held fixed at $d = 5$. The recursive strategy has a fixed tape length of 10890 floats.

conventional case the tape usage is proportional to the number of time steps n. From Figure 6 the tape $T = 1089n$ for the conventional case. It follows from this experiment that with a fixed amount of tape on a particular machine, the conventional approach would quickly fail as the number of time steps increased. Figure 7 shows the wall-clock time for the same experiment. In all trials the conjugate gradient procedure converged in three iterations. The conventional strategy took a wall-clock time of $t = 0.147n + 0.0571$ seconds. The recursive strategy took longer to complete, and its growth is not linear. Table 3 contains further information on this particular set of trials.

[5]. CONCLUSIONS

We have shown in this study how a recursive strategy for the adjoint-method calculation of the gradient may be applied to variational data assimilation studies of large-scale geophysical problems. The main result is that significantly larger assimilation studies can be performed with this recursive strategy than is possible with the conventional forward-adjoint methods, given the physical limitations of available computer storage hardware.

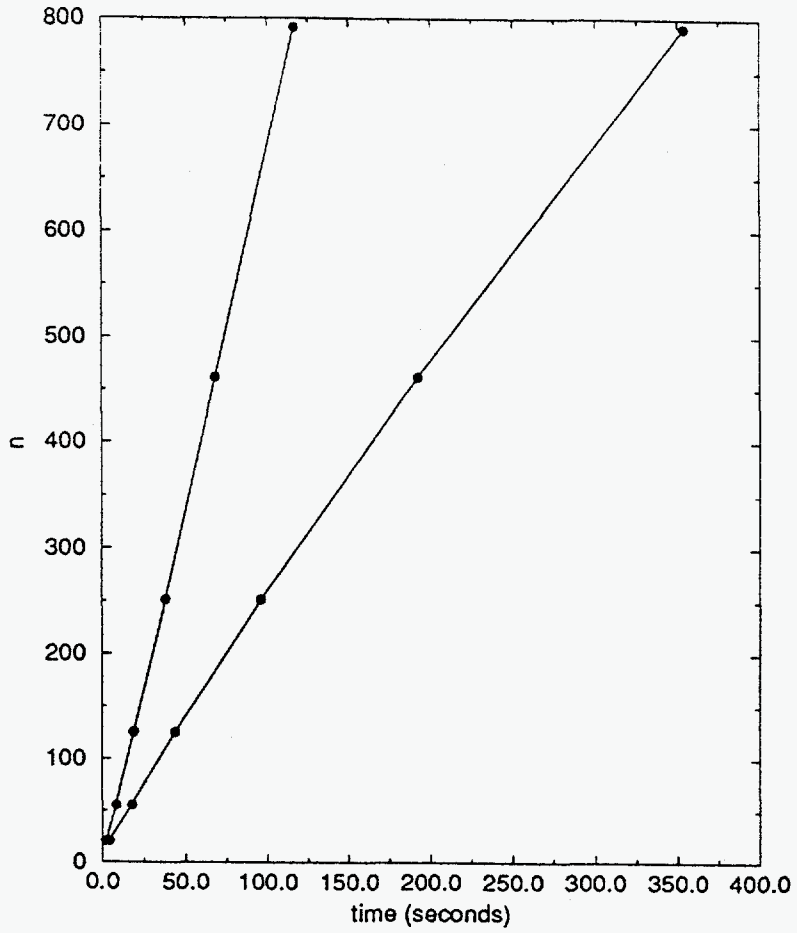

FIGURE 6. Comparison of the conventional (left) and recursive (right) strategy for the T&T problem. In the recursive strategy the snap count was held fixed at $d = 5$.

TABLE 2. Wall-clock time and tape length for the recursive approach in the T&T problem for a doubling of resolution.

n	Time (sec)	Tape (floats)	Snaps	Reps
224	356.94	42250	10	3
224	315.92	84500	20	2

While the recursive strategy requires additional computational effort (or wall-clock time) the strategy is viable. Furthermore, the recursive strategy yields the gradient with no degradation, as compared with the conventional approach.

In theory, when the number of snaps and reps (i.e., the number of storage units measured in R, and the number of additional unrecorded forward runs) is equal, these are both bounded by $\log_4 n$, where n is the number of time steps in the evolution equation. In practice, the strategy is best used by picking the maximum number of snaps that the particular computer hardware can manage, thus minimizing the number of reps.

Insofar as computer program design, the best strategy for large-scale problems is to con-

TABLE 3. Ratio of the wall-clock time for the recursive ($d = 5$) and conventional approach versus n and number of reps for the T&T problem.

n	Time Ratio	Reps
21	1.7665	2
56	2.0687	3
126	2.3139	4
252	2.5274	5
462	2.8306	6
792	3.0425	7

struct programs that are as compute-intensive as possible and the least memory-intensive. This yields the greatest variation in the computational effort for any given choice of snaps and reps. This is especially true in parallelized programs because the computational effort will drop as more processors are used, whereas the storage requirements remain fixed independent of the number of processors.

The implementation of the recursive strategy requires minimal modification of conventional codes that compute forward and adjoint problems. The requirements are that four modules be provided: (1) a forward module that runs without writing the state set between a specified starting and an ending time step; (2) a module that computes a single unrecorded forward and a single adjoint step, given a specific time step; (3) a module that writes to tape the state set at the current time step; and (4) a module that retrieves from tape the last recorded state set. An additional module, which is to be considered the driver, runs the above-mentioned modules according to the recursive schedule. The driver requires as input the total number of time steps, the number of snaps, and the number of reps.

One approach in the implementation of the schedule driver is to have the schedule computed only once at the top of the program. The schedule instructions are saved in integer arrays, which are then called in sequence to drive the four modules. The benefit of precomputing the schedule is not warranted in some applications, since the schedule module increases insignificantly the overall computational effort. The preferred alternative is to use the schedule driver to control the above-mentioned modules, thus not wasting register memory for the schedule arrays needed in the first approach that could otherwise be used in the adjoint problem. An estimate of the additional memory for the integer schedule arrays of the first approach is as follows: a "schedule array" with the instruction directives of size $2rn$ is required, plus one or two arrays of similar size that direct the writing and reading of snaps from tape. The total register overhead is then on the order of $4rn$ integers. The user's particular application will clearly dictate which alternative works best.

This schedule driver is available via anonymous ftp from `info.mcs.anl.gov`. The file is called `/pub/tech_reports/restrepo/schedule.tar.Z`. Alternatively, the schedule software is available in either Fortran or C versions from the Word-Wide-Web in the software section of `http://www.mcs.anl.gov/people/restrepo/index.html`.

REFERENCES

1. National Academy Press, *Four-Dimensional Model Assimilation of Data, A Strategy for the Earth System Sciences*, National Research Council, 1991.

2. M. Ghil, *Metereological Data Assimilation for Oceanographers. Part I: Description and Theoretical Framework*, Dynamics of Atmospheres and Oceans **14** (1989), 171–218.

3. F. X. Le Dimet, O. Talagrand and Variational Algorithms for Analysis and Assimilation of Metereological Observations, Tellus, Series A **38** (1986), 97–110.

4. W. C. Thacker, *Oceanographic Inverse Problems*, Physica D (1992).

5. J. C. Derber, *Variational Four Dimensional Analysis Using Quasi-Geostrophic Constraints*, Mon. Wea. Rev. **115** (1987), 998–1008.

6. J. Marotzke and C. Wunsch, *Finding the Steady State of a General Circulation Model Through Data Assimilation: Application to the North Atlantic Ocean*, Journal of Geophysical Research **98, C11**, 20149–20167.

7. J. Schröter, *Variational Assimilation of XBT Data: I*, J. Phys. Ocean. **20** (1990), 672–688.

8. J. Schröter and C. Wunsch, *Solution of Nonlinear Finite Difference Ocean Models by Optimization Methods with Sensitivity and Observational Strategy Analysis*, Journal of Physical Oceanography **16** (1986), 1855–1874.

9. J. Marotzke, *The Role of Integration Time in Determining a Steady State through Data Assimilation*, American Metereological Society **79** (1992), 35–38.

10. B. F. Farrell and A. M. Moore, *An Adjoint Method for Obtaining the Most Rapidly Growing Perturbation to Oceanic Flows*, Journal of Physical Oceanography **22** (1992), 338–349.

11. W. C. Thacker and R. B. Long, *Fitting Dynamics to Data*, Journal of Geophysical Research **93, C2**, 1227–1240.

12. O. Talagrand and P. Courtier, *Variational Assimilation of Meteorological Observations with the Adjoint Vorticity Equation, Part I, Theory*, Quart. J. Roy. Met. Soc. **113** (1987), 1311–1328.

13. E. Tziperman and W. C. Thacker, *An Optimal Control-Adjoint Equations Approach to Studying the Oceanic General Circulation*, J. Phys. Ocean. **19** (1989), 1471–1485.

14. A. Griewank, *Achieving Logarithmic Growth in Temporal and Spatial Complexity in Reverse Automatic Differentiation*, Optimization Methods and Software **1** (1992), 35–54.

15. G. Veronis, *Wind Driven Ocean Circulation- Part 2. Numerical Solutions of the Nonlinear Problem.* , Deep Sea Research **13** (1966), 31–55.

16. J. Pedlosky, *Geophysical Fluid Dynamics*, Springer-Verlag, New York, 1979.

MATHEMATICS AND COMPUTER SCIENCE DIVISION, ARGONNE NATIONAL LABORATORY, ARGONNE, IL 60439 U.S.A
E-mail address: restrepo@mcs.anl.gov

MATHEMATICS AND COMPUTER SCIENCE DIVISION ARGONNE NATIONAL LABORATORY, ARGONNE, IL 60439 U.S.A.
E-mail address: leaf@mcs.anl.gov

INSTITUTE OF SCIENTIFIC COMPUTING, MATHEMATICS DEPARTMENT, TECHNICAL UNIVERSITY OF DRESDEN, MOMMSENSTR. 13, 01062 DRESDEN, GERMANY
E-mail address: griewank@math.tu-dresden.de