Development and Tests of a New Distributed-Memory MM5 Adjoint

FRANK H. RUGGIERO

Space Vehicles Directorate, Air Force Research Laboratory, Hanscom AFB, Massachusetts

JOHN MICHALAKES

National Center for Atmospheric Research, Boulder, Colorado

THOMAS NEHRKORN AND GEORGE D. MODICA

Atmospheric and Environmental Research, Inc., Lexington, Massachusetts

XIAOLEI ZOU

Department of Meteorology, The Florida State University, Tallahassee, Florida

(Manuscript received 1 March 2005, in final form 1 September 2005)

ABSTRACT

Updated versions of the Tangent Linear Model (TLM) and adjoint of the fifth-generation Pennsylvania State University–National Center for Atmospheric Research Mesoscale Model (MM5) have been developed and are now available to the meteorological community. The previous version of the MM5 TLM and adjoint were designed for single-processor computer architectures, based on version 1 of MM5, and were hand coded, which made it difficult to maintain up-to-date versions of the TLM and the adjoint as MM5 evolved. The new TLM and adjoint are based on version 3 of MM5 and run efficiently on multiple-processor computers. The TLM and adjoint were developed with the aid of the Tangent Linear and Adjoint Model Compiler (TAMC) automatic code generator. While some manual intervention is still necessary, the use of the automatic code generator can significantly speed code development and lower code maintenance costs. The new TLM and adjoint contain most of the physics packages and observation operators that were available in the MM5 version 1 TLM and adjoint. The new adjoint has been combined with the MM5 version 3 nonlinear model and an updated minimization module in a four-dimensional variational data assimilation analysis configuration. Accuracy of the new TLM and adjoint has been verified by individual unit and system tests as well as comparisons with the adjoint from MM5 version 1. Timing tests showed substantial decreases in time to solution when increasing the number of processors devoted to the problem.

1. Introduction

While there is potential in developing improved physical parameterizations and/or increasing model resolution, a substantial source of improvement in numerically simulating weather systems can be achieved by more accurately obtaining the correct initial state. Obtaining this initial state for meso- time and space scales has been virtually impossible due to the coarse distribution of conventional observations as well as the complete unavailability of conventional data in many areas, particularly over the oceans. The deployment of remote sensing systems has the potential to change this situation. For example, Doppler radar and satellite sensors can provide data at very fine temporal and spatial resolutions. A significant obstacle to the use of remotely sensed data is that the quantities typically measured by these instruments are not directly usable by numerical weather prediction models and tend to be irregularly distributed in time and space. To be truly useful, these data must be converted into regular 4D fields (time and three spatial dimensions) for meteorological quantities or be directly assimilated into the forecast model to produce the 3- and 4D fields for atmospheric variables. This latter option is made possible

Corresponding author address: Frank H. Ruggiero, AFRL/VSBYA, 29 Randolph Rd., Hanscom AFB, MA 01731-3010.
E-mail: frank.ruggiero@hanscom.af.mil

in four-dimensional variational data assimilation (4DVAR) through the use of a forward observation operator that converts the model variables into the observed quantities. Variational data assimilation using the adjoint formalism was first proposed in a concise formulation by Le Dimet and Talagrand (1986) for meteorological applications and has been implemented by many others since.

While research studies have shown the utility of 4DVAR in assimilating data (e.g., Zupanski and Zupanski 2002; Sun 2005; Gérard and Saunders 1999; Vukicevic et al. 2004), its high computational cost and difficulty in maintaining the adjoint as the nonlinear model evolved has limited its use. The European Centre for Medium-Range Weather Forecasts and Météo-France have been running 4DVAR for several years (Rabier et al. 2000; Gauthier and Thépaut 2001), generating analyses for global forecast models. More recently the meteorological centers of the United Kingdom, Canada, and Japan have followed suit. In addition, the Japan Meteorological Agency has started running a mesoscale 4DVAR data assimilation system. Typically, at the operational centers a form of incremental 4DVAR (e.g., Courtier et al. 1994; Veersé and Thépaut 1998) is used where the adjoint is run at a coarser resolution and/or with simpler physics than the forecast model in order to decrease the time to solution. In the operational centers much use is made of satellite observations in the cost functions (e.g., see Andersson et al. 2005). Additional computational efficiency is gained by optimization of the codes to run on distributed-memory multiple-processor computers.

To be sure, there are limitations in the current implementations of the 4DVAR approach. One limitation is the linear assumption in the use of the adjoint. This can be a particular problem when dealing with nonlinear processes such as boundary layer fluxes (e.g., Mahfouf 1999) or convectively driven dynamics (e.g., Errico and Raeder 1999). Another issue is the determination of the cost function, particularly when it comes to the determination of the forecast model's errors. Often, the model's errors are ignored and a "perfect model" is assumed. Finally, there is the issue of the limited physics used in some adjoints. Recently there has been some interest in the use of ensemble Kalman filtering (EnKF) in data assimilation. Lorenc (2003) explored the differences between EnKF and 4DVAR. While EnKF may offer some advantages, such as ease of implementation, it does not seem as well suited for the use of high-resolution/density observations as does 4DVAR.

The objective of the effort described in this paper is to develop a new 4DVAR system based on the fifth-generation Pennsylvania State University–National Center for Atmospheric Research (Penn State–NCAR) Mesoscale Model (MM5). This is a full-field 4DVAR system in contrast with those 4DVAR systems using the incremental approaches mentioned above. The goals of the project are to increase the computational efficiency to allow for the possibility of using 4DVAR for real-time forecast applications and to ease the task of developing and maintaining the adjoint as the nonlinear model is updated through the use of automatic code compilers. The format of this paper is as follows. First in section 2, there is a review of 4DVAR, MM5, and the prior 4DVAR system based on MM5. Then the methodology used for development of a new MM5 Tangent Linear Model (TLM) and adjoint, including coding enhancements that allow interprocessor communications for running on distributed-memory multiple-processor computers is described in section 3. Results of 4DVAR runs that show the accuracy and computational efficiency of the new code are presented in section 4. Finally, the work is summarized, the conclusions presented, and future work discussed in section 5.

2. Background

The variational method solves a minimization problem, in which a model initial state $\mathbf{x}(t_0)$ is found that minimizes an objective function. In meteorological applications, the cost function often takes a form similar to the following (see also, e.g., Ide et al. 1997):

$$J[\mathbf{x}(t_0)] = \frac{1}{2}[\mathbf{x}(t_0) - \mathbf{x}^b(t_0)]^{\mathrm{T}}\mathbf{B}_0^{-1}[\mathbf{x}(t_0) - \mathbf{x}^b(t_0)] + \frac{1}{2}\sum_{i=0}^{n}(\mathbf{y}_i - \mathbf{y}_i^o)^{\mathrm{T}}\mathbf{O}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^{\mathrm{o}}), \quad (1)$$

where $\mathbf{x}^b$ is the a priori background or first-guess field with assumed error covariance $\mathbf{B}_0$ and $\mathbf{y}_i^{\mathrm{o}}$ denotes the vector of observations at time t_i, $\mathbf{O}_i$ is the corresponding observation error covariance matrix, and n is the number of times there are observations to assimilate. Given the model predicted variables,

$$\mathbf{x}(t_i) = M(t_i, t_0)\mathbf{x}(t_0), \quad (2)$$

where M represents the forecast model, then the simulated observations $\mathbf{y}_i$ are obtained by applying the (in general, nonlinear) observation operator H to the model-predicted variables:

$$\mathbf{y}_i = H[\mathbf{x}(t_i)]. \quad (3)$$

The minimization is performed over an analysis time window (t_o, t_n). Minimization of the variational analysis problem requires an estimate of the gradient of J with

respect to the solution vector $\mathbf{x}(t_0)$, which is most efficiently computed with the adjoint of the observation operator (and, for 4DVAR, the adjoint of the forecast model).

In a practical 4DVAR system, the observation operators act to convert the model variables into the variables actually being observed (i.e., for satellite sounders, radiances). As the forward, nonlinear, model advances in time, the residual between the model-predicted and actual observations are noted and stored. The adjoint of the forecast model is essentially a linearized forecast model in reverse that uses the residuals between the actual and predicted observations to calculate the full gradient of the cost function. The gradient solver, or minimizer, then determines how best to adjust the initial conditions so that $J[\mathbf{x}(t_0)]$ decreases in size in the succeeding iteration. Nonlinear forecast models are the typical mesoscale forecast models, of which several are readily available. Gradient solvers are also readily available and are independent of the other two components. However, in order for the system to work, the adjoint of the nonlinear forecast model is needed and by definition is dependent on the forecast model. Actually, it is the adjoint of a linearized version of the Tangent Linear Model (TLM) forecast model. The TLM is then a by-product of the adjoint-generating process and as we discuss later has applications of its own. Therefore, as the forecast model is updated and recoded, the adjoint must be changed as well.

In this effort we used the MM5, which is the latest version of the mesoscale model originally developed by Anthes and Warner (1978), as the forecast model on which to base the TLM and adjoint. Version 1 of MM5 (MM5v1) was released in 1994. MM5v1 was the first generation of Penn State–NCAR mesoscale models to include nonhydrostatic dynamics (Grell et al. 1995) and is based on a set of equations for a fully compressible atmosphere in a rotating frame of reference. The basic prognostic variables are pressure perturbation, momentum, temperature, and five moisture variables. It uses the Arakawa and Lamb (1977) B-grid-type structure, which permits greater efficiency in the computations. Its vertical coordinate is terrain following:

$$\sigma = \frac{p_0 - p_t}{p_s - p_t}, \tag{4}$$

where p_0, p_t, and p_s are the reference, model top, and surface pressures, respectively. MM5 was designed to run at horizontal grid spacings as small as 10 km. The dynamical equations are solved by finite-difference methods. Advection is handled by a second-order-centered scheme. Time splitting is used to compute separately the fast- and slow-moving waves. For slower-moving waves a long time step is used with the leapfrog approach and the Asselin (1972) filter. For the faster-moving waves, the semi-implicit method of Klemp and Wilhelmson (1978) is used. MM5v1 also included a modest list of physics parameterizations such as the Dudhia (1989) microphysics, Kuo (1974) and modified Arakawa–Schubert (Grell 1993) convection schemes, and Blackadar (1979) planetary boundary layer parameterization.

A few years after the release of MM5v1, the TLM and adjoint of MM5v1 were released (Zou et al. 1997). The MM5v1 TLM and adjoint were based on a simplified set of physics parameterizations available with MM5v1 such as the Kuo convection scheme, a bulk aerodynamic formulation of the planetary boundary layer (Deardorff 1972), a dry convective adjustment, and grid-scale resolvable large-scale precipitation. Even with this relatively primitive configuration, useful results were obtained. For instance, Zou et al. (2001) showed that the 4DVAR technique combined with MM5 produced superior forecasts of the position and central pressure of Hurricane Felix compared with forecasts from the National Hurricane Center. Wee and Kuo (2004) used the MM5v1 4DVAR to assess the impact of using a digital filter as a weak constraint in 4DVAR. Kuo et al. (1996) used the phase delay associated with radio signals propagating from global positioning system satellites as observations for MM5v1 4DVAR in order to improve the assimilation of precipitable water. Kleist and Morgan (2005) employ the MM5v1 adjoint to calculate the sensitivities of response functions associated with forecasted wind and temperature fields in order to provide a synoptic interpretation of corresponding forecast sensitivity gradients. As part of this work they have run the adjoint (for a single iteration) in real time for the last 2 yr.

As the MM5 nonlinear model evolved over the years, undergoing several revisions, it added more complex convective, planetary boundary layer, moisture, and other physics parameterization options. In addition, the software was upgraded to increase its computational performance and to run on additional computer platforms. The most important software change was the implementation of processor-to-processor communication to allow the model to run on distributed-memory multiple-processor computers (Michalakes 2000). The increased efficiency resulted in faster wall-clock run times of the model. This allowed wider use of the MM5 by both research (Mass and Kuo, 1998) and operational organizations (e.g., Air Force Weather Agency and some local National Weather Service offices). During

this period of evolution of MM5, the TLM and adjoint remained relatively static. The original MM5 TLM and adjoint had been laboriously hand coded. This posed a problem in that every time the forecast model was upgraded, the adjoint needed to be recoded and rigorously tested. This was a time-consuming process and there were no resources available to complete the job. Therefore, the MM5 TLM and adjoint fell behind the MM5 forecast model in terms of available physical parameterizations and computational efficiency.

In this project the original objective was to modify the existing MM5v1 TLM and adjoint to allow it to run on multiple-processor computers. It quickly became apparent, however, that a parallel 4DVAR system based on the current version of MM5 would gain wider user acceptance. Therefore, a commitment was made to build a new TLM and adjoint that would be based on the current version of MM5, version 3 (MM5v3).

3. Development of MM5v3 TLM, adjoint, and observation operators

There were two distinct phases in developing the new MM5v3 TLM and adjoint. First, the TLM and adjoint were created by making use of the Tangent Linear and Adjoint Compiler (TAMC; Giering and Kaminski 1998). The resulting TLM and adjoint programs were serial codes designed to run on a single processor. The second step was to adapt the codes to run optimally on multiple-processor machines. The two steps are detailed below as well as a description of the observation operators that were either developed or adopted from the MM5v1 4DVAR system.

a. TLM and adjoint development

1) The Tangent Linear and Adjoint Model Compiler (TAMC)

TAMC is a source code to source code translator that generates FORTRAN code for the TLM or adjoint from the FORTRAN code of the nonlinear forecast model (NLM). TAMC was developed and tested by Giering and Kaminski (1998), and it has been used in oceanographic and other modeling applications (e.g., Kaminski et al. 1996). The NLM source code must conform to the FORTRAN-77 standard, with some additional restrictions (some features of the FORTRAN-90 standard are also supported[1]). Examples of the code conventions that cannot be handled by TAMC can be found in Giering (1999). They include the use of FORTRAN-77 pointers, FORTRAN-90 include statements, common blocks with different numbers or kinds of arguments at different occurrences, intrinsic functions passed as arguments, and comments that occur before continuation statements. The NLM derivatives are computed in the reverse mode (to create the adjoint) or in the forward mode (TLM). TAMC normalizes the source code and applies a control flow analysis. TAMC also applies an intraprocedural data dependence and an interprocedural data flow analysis. Given the independent and dependent variables of the specified top-level routine, TAMC determines all active routines and variables and produces derivative code only for those. It is possible to incorporate the TAMC as part of the NLM compilation process, requiring the maintenance of just the NLM code. However, certain constructs within the MM5 forward model are not handled in TAMC and so the NLM code must be modified to allow for complete and correct generation of TLM and adjoint code by TAMC. For this project, we have found that amount of code modification to be prohibitive. For the development of the MM5 adjoint, we have chosen instead to use TAMC as a development tool only, and separately maintain the TLM and adjoint versions of the model code. This approach makes it possible to minimize changes to the current MM5 code as supported by NCAR, but it requires a mixture of manual and automatic code generation. The number of lines that required manual changes amounted to approximately 5% of the total number of lines of new code. Initial code generation took approximately four to six man-weeks for the TLM and adjoint each. Setting up the testing software, running the tests, and debugging errors of the code took several months total. Compared to manual code generation only, this method still substantially reduces the time for coding and error debugging.

2) Code generation and testing

As part of the TLM and adjoint development, the MM5 NLM code is temporarily modified to remove some nonstandard features (primarily, the use of FORTRAN-77 pointer variables, a nonstandard but widely used extension to FORTRAN-77) before being passed to TAMC. Since the MM5 NLM is maintained by a different organization (NCAR) there was limited opportunity to make these changes permanent and eliminate this step in subsequent TLM and adjoint builds. The output generated by TAMC is then compared manually against the existing, hand-coded MM5v1 TLM and adjoint codes, taking into account differences between the MM5v1 and MM5v3 NLM codes. Manual corrections are made as needed. Both

[1] A commercial version of TAMC is now available that supports most of the FORTRAN-90 standard.

the TLM and adjoint are tested for correctness. We use the standard comparison of the TLM and finite-difference NLM gradients to check for the correctness of the TLM (e.g., see Zou et al. 1997):

$$\lim_{\delta x \to 0} \frac{[J(\mathbf{x} + \delta\mathbf{x}) - J(\mathbf{x})]}{\delta J(\mathbf{x}, \delta\mathbf{x})} = 1 \tag{5}$$

where $J(\mathbf{x} + \delta\mathbf{x})$ and $J(\mathbf{x})$ are from the NLM, and $\delta J(\mathbf{x}, \delta\mathbf{x})$ is from the TLM. Here, $\delta\mathbf{x}$ represents a perturbation from the initial conditions. The definition of the adjoint is used to check for consistency between the TLM and adjoint via the equivalences of the sets:

$$\langle \mathbf{y}, L\mathbf{x} \rangle = \langle L^*\mathbf{y}, \mathbf{x} \rangle, \tag{6}$$

where $\mathbf{x}$ is the TLM input, $\mathbf{y} = L(\mathbf{x})$ is the TLM output, and L, L^* are the TLM and ADJ, respectively. This testing is performed for individual subroutines as well as the complete model integration.

Overall, there was considerable agreement between the TAMC-generated derivative (TLM and adjoint) code and the MM5v1 hand-coded versions, not only in the code for the computation of the derivatives themselves, but also in the choice between storing and recalculating intermediate NLM quantities. Almost all of the problems encountered by TAMC have to do with the recomputation of intermediate NLM quantities of the adjoint code, not with the derivative code itself. The most common types were caused by known limitations of TAMC, which are complicated control flows with "go to" statements and dependencies on array elements. Detailed examples of these are given in the appendix.

The codes used to do the correctness tests from Eqs. (5) and (6) are retained and maintained alongside the newly generated TLM and adjoint source code. Thus, when the MM5v3 underwent an upgrade from v3.3 to v3.4, a MM5v3.4-compatible TLM and adjoint were generated and successfully tested within 2 days. The specific procedure for updating the TLM is as follows. The new version of MM5 nonlinear model was downloaded from the NCAR repository. A comparison between the v3.3 and v3.4 nonlinear models was done and the v3.3 version used for the TLM generation was updated as needed. Using TAMC, a new version of the TLM was generated using the nonlinear model with v3.4 updates. A comparison between the v3.3 and v3.4 versions of the TLM was done and the v3.3–v3.4 changes were merged into the TLM. The TLM unit tests were redone. This step resulted in some further changes and corrections to parts of the TLM code and test code. When the process was completed, 22 files of the TLM had been affected with a total of 600 lines being changed. The procedure for the adjoint revision was similar.

b. Modifications for distributed memory computers

Once complete, the full TLM and adjoint needed to be modified to run on distributed-memory multiple-processor machines. The developers of TAMC do state that they can provide basic support to the widely used Message Passing Interface (MPI) library in the TLM and adjoint construction. We did not try to follow this path because we wanted to retain control of the parallelization process in order to maximize the computational efficiency. Parallelization of the code is carried out using two-dimensional horizontal domain decomposition and additional parallelization techniques developed at Argonne National Laboratory that preserve the original look and operation of the existing serial code. These techniques have been successfully demonstrated with MM5 (Michalakes 2000). Since the finite differencing in the forward nonlinear model on which they are based is explicit, the TLM and adjoint codes require only communication between adjoining subdomains resulting from the decomposition over processors. This communication is accomplished with the MPI library. As with MM5, this and other aspects of the parallel code architecture are hidden from the user by the use of an application-specific parallel library and a compile time source translator. The Runtime System Library (RSL; Michalakes 1997b) provides domain decomposition, local address space computation, distributed input/output (I/O), and interprocessor communication supporting parallelization of both the solver and mesh refinement code. The FORTRAN Loop and Index Converter (FLIC; Michalakes 1997a) translates code at compile time to generate parallelized code (which only the compiler sees) from a single version of the source model. Adaptation of these techniques to the TLM was relatively straightforward since the TLM is simply the linearized version of the MM5v3 model to which these techniques have already been adapted. Applying the parallelization techniques to the adjoint model, however, raised new issues. In particular, it was necessary to modify loops and index the arithmetic to reestablish "Owner Computes" and to eliminate instances of false recursion introduced with the creation of the adjoint model. Specifically, producing an adjoint version of a code can sometimes result in array index offsets appearing on the left-hand side of the assignment statements; that is, array elements other than (I, J)—for example, $(I, J + 1)$—may be assigned. These neighboring elements may actually lie not on this processor, but on a neighboring processor's subdomain. Offset indices on the left-hand side of an assignment

```
DO 20 J = 2, JLXM
    DO 20 I = 2, ILXM
    F (I,J+1) = -(UA9(I+1,J+1) + UA9(I,  J+1)) * FTEN(I) / ...
    F (I,J)   = -(UA9(I+1,J+1) + UA9(I,  J+1)) * FTEN(I) / ...
    F (I,J)   =  (UA9(I+1,J, )  + UA9(I,  J, ))  * FTEN(I) / ...
    F (I,J-1) =  (UA9(I+1,J, )  + UA9(I,  J, ))  * FTEN(I) / ...
20 CONTINUE
```

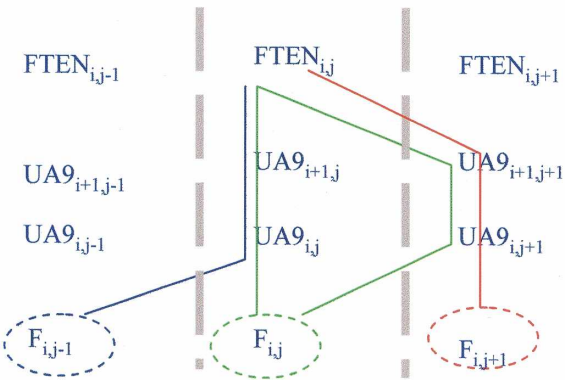

FIG. 1. Simplified adjoint computation for the tendency of temperature (FTEN) from horizontal advection showing just the *U* component wind. Here, UA9 is the NLM's trajectory value of *U*. The colored lines show the variables used in the equations with matching colors in the code sample. The gray dashed lines indicate potential borders of the processor subdomains.

statement generally indicate recursion (the value of an array element depends, sequentially, on the computation of the previous array element). In most cases in the MM5 adjoint, however, the dependency is introduced as a programming convenience for expressing the adjoint and the actual recursive data dependency does not exist. Thus, the adjoint loops and expression can usually be reordered to factor out the index offsets on the left-hand sides of assignments, reestablishing the owner computes properly for parallelism.

An example of this can be seen in Fig. 1. This simple example is for the horizontal advection of temperature. With the index arithmetic on the left-hand side of the equation the adjoint value of the temperature (F) for each grid point is arrived at by summing over three iterations of the loop. When the ($J + 1$) grid column lies off the edge of the processor subdomain, interprocessor communications are needed in order to get the wind values at the $J + 1$ grid points in order to compute $F(I, J)$. The same is true for FTEN(I, J) when computing $F(I, J + 1)$. Since FTEN is subject to interprocessor communications, it needs to be fully dimensioned in I and J, whereas in MM5 and the resulting adjoint the arrays for FTEN are only I-dimensional. This required the restructuring of the tendency arrays in the adjoint. When the (J-1) grid point lies off the processor subdomain, a spurious value of $F(I, J\text{-}1)$ is computed. This is allowed to happen in the code in order not to complicate the code but the computed value is not used. The result of the above is that by communicating the off-processor values prior to the call of the subroutine and by allowing some spurious calculations of F on the boundaries, all the contributions to the F array can be computed without modifying the loops or loop bodies and still ensure bit-for-bit agreement with the nonparallel version of the adjoint.

When a 4DVAR system is executed, the nonlinear forecast model is run forward in time and typically its output from each time step, called the model trajectory, is written to disk. The adjoint of the model is then executed, essentially running backward in time. However, since it is based on the linearized version of the forward model, it needs to read the model trajectory information from each time step of the forward model as it retraces the forward model's steps. Reading and writing this trajectory information to and from large numbers of parallel processors has a significant effect on the parallel efficiency of the 4DVAR system. Initially for the NLM, TLM, and adjoint, the thought was to write and read the forward model data needed for each subdomain to and from the disks mounted locally on each processor on systems such as the IBM SP. In reality this was not practical given the amount of disk space and I/O needed and the other system needs of the local scratch disks. Therefore, all the trajectory information was written to a single file using a parallel file system such as the General Parallel File System on an IBM SP P3. This was immediately recognized as a bottleneck in the code's execution. To mitigate the amount of I/O calls for the time step, data were buffered from time step to time step and actually read or written to disk less frequently, only every five time steps. This reduced the number of I/O calls. The I/O was done asynchronously with computation and only for data needed for a particular processor. Nevertheless, performance degradations were still seen because the parallel file server was still being saturated with simultaneous I/O requests. An additional mitigation measure was then added in which the I/O for the subdomains was staggered so that at any particular time step only one in four of the subdomains needed to write to the disk file. While these steps helped alleviate the I/O bottleneck substantially, it was still impacting the ability of the code to perform efficiently when using multiple processors. A final additional mitigation measure was implemented to take advantage of the fact that running the code over multiple computational nodes provides access to the aggregate memory of these nodes. By accessing additional processing nodes, one gets access to additional memory. We found that we

could make use of this additional memory by storing the trajectory information internally in memory instead of writing and reading it to disk. Results presented below show that this has a significant impact on reducing wall-clock processing time.

c. Observation operators

A small set of observation operators have been developed to work with the MM5v3 NLM, TLM, and its adjoint. Two of the observation operators were leveraged off previous efforts at NCAR. One of the operators is for rawinsondes. Since rawinsondes provide direct measurements of model variables, the only function of the observation operator is to interpolate the model data to the observation locations. The second observation operator adapted from the NCAR 3DVAR work is the *Geostationary Orbiting Environmental Satellite-8* (*GOES-8*) sounder data. The *GOES-8* sounder measures top-of-atmosphere radiances; therefore, in addition to interpolating the model data to the measurement locations, the observation operator must also execute radiative transfer codes to compute the model variables into radiances. Likewise, an observation operator was developed for the 12-channel Radiometrics vertically pointing, ground-based microwave profiling radiometer (Ware et al. 2003). An observation operator for surface observations has also been developed. The only part of the observation operators that requires interprocessor communications is the horizontal interpolation of the data. All the observation operators use the same horizontal interpolation module, which carries out the necessary interprocessor communication in order for it to run efficiently on distributed-memory multiple-processor computers. Modifying the cost function to include the additional observations is simple to do and involves the modification of one subroutine.

4. Results

a. Accuracy checks

As each module of the TLM and adjoint was assembled it underwent unit testing using Eqs. (5) and (6). When the entire TLM and adjoint were completed, the same equations were used to test the entire TLM and adjoint in system tests. The results of these tests showed that the new TLM and adjoint were accurate to within machine precision. To best do the accuracy testing, it is necessary that the code be run in 64-bit mode (i.e., double precision). However, when running the adjoint in 4DVAR mode, it may not be necessary to run in double precision. For all the runs presented in this paper, the code was executed in double precision.

Along with the unit and system testing, checking of the accuracy of the adjoint was accomplished by comparisons of analyses generated by the MM5v1- and MM5v3-based 4DVAR systems using the respective v1 and v3 adjoints. For these tests, the same gradient solver developed by Liu and Nocedal (1989) is used in both 4DVAR versions. One of the comparisons was done using a case from the Severe Environmental Storms and Mesoscale Experiment (SESAME). The case in question was a severe weather outbreak in the Red River Valley region of Texas on 10–11 April 1979 (Alberty et al. 1980). For these comparisons we assimilated gridded analyses ("direct obs") over an assimilation window of 3 h and the perfect model assumption was used. In this case the cost function from Eq. (1) does not include the background term (the first term on the right-hand side). Furthermore, the observation error covariance matrix, $\mathbf{O}$, was assumed to be diagonal, and the observation operator, Eq. (3), reduces to

$$\mathbf{y}_i = \mathbf{x}(t_i). \tag{7}$$

The physics used included the Kuo (1974) convective parameterization, grid-scale precipitation, and the MM5 bulk boundary layer parameterization (Deardorff 1972). The bulk boundary layer parameterization was used in the comparison because the two versions of the MM5 adjoint have different multilayer boundary layer parameterizations. The analyses were constructed using a horizontal grid domain of 19×17 points with a grid spacing of 120 km.

Figure 2 contains the comparisons of the analysis low-level temperature and wind fields at 1800 UTC 10 April 1979. One can see that the analyses are quite similar. There are some very small differences in the position of the isotherms and the wind barbs. The cost function computed from Eq. (1) for the analyses computed from MM5v1 and MM5v3 4DVAR runs are presented in Fig. 3. As the number of iterations increases, the runs follow slightly different paths, although they eventually do converge close to the same point. As in the results in Fig. 2, we can see that the two versions of MM5 do not produce exactly identical results. This is to be expected since in the course of the migration from MM5v1 to MM5v3 some fixes and restructuring have been made to the nonlinear model code. However, the closeness of the results indicates that the new version of the MM5 adjoint is qualitatively performing similarly to the previous version.

b. Computational efficiency results

From an operational weather forecasting standpoint, the critical parameter is time to solution. Therefore, the

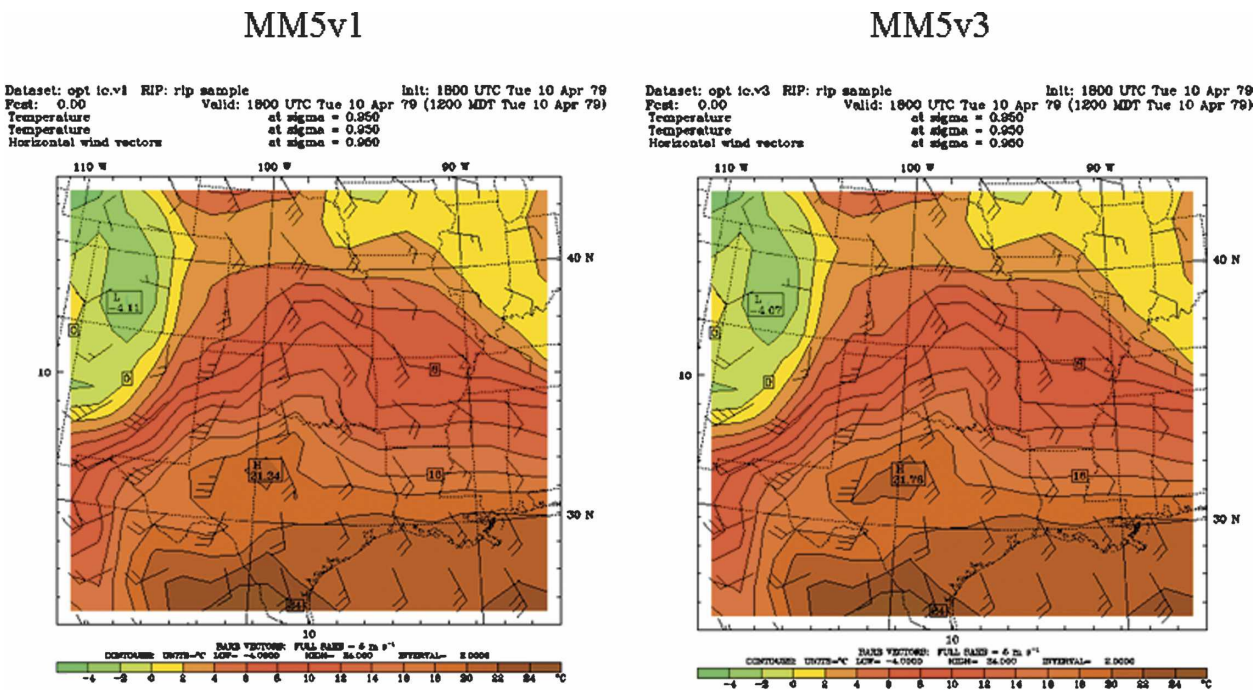

FIG. 2. Comparison of low-level temperature and winds for 1800 UTC 10 Apr 1979 from MM5v1 4DVAR and MM5v3 4DVAR analyses.

main computational metric we are concerned with is the speedup of wall-clock time (i.e., the time from program start to completion):

$$\text{Speedup} = \frac{\text{Time}(1_{\text{processor}})}{\text{Time}(N_{\text{processors}})}. \tag{8}$$

The SESAME case as structured above is too small to make efficient use of increasing numbers of processors. To measure the wall-clock time speedup for this case, the horizontal grid spacing was artificially reduced from 120 to 15 km while slightly enlarging the domain. The resulting grid was 152 × 136 points. For the parallel-processing runs a distributed-memory multiple-processor version of the Liu and Nocedal (1989) gradient solver was employed. A 6-h data assimilation window was employed using direct obs. The same physics configuration as used in the comparisons above was employed. The new adjoint was optimized to run on distributed-memory multiple-processor computers. Timing tests were run on an IBM SP P3 and a Compaq ES-45, which were located at the Aeronautical Systems Center at Wright-Patterson AFB, Ohio, and on a Silicon Graphics Inc. (SGI) Origin 3800 located at the Army Research Laboratory at the Aberdeen Proving Ground, Maryland. All of the experiments were run to 10 iterations. The results are shown in Fig. 4. Comparing the results from one to four processors, all of the computers have speedups that scale less than ideal. The speedup values show dramatic improvement between 4 and 16 processors. This is due to the turning on of the internal I/O feature for the trajectory information when using 16 or more processors. On the IBM, the speedup remains near ideal at 64 processors, but for a problem of this size, additional benefit is not achieved beyond 64 processors for it or the SGI as the interprocessor communication becomes a bigger time bottleneck. The tests on the Compaq show dramatic improvement when turning on the internal I/O, far surpassing the IBM and SGI and apparently achieving superlinear speedups. This is explained by the fact that the parallel I/O on the Compaq is not very efficient and definitely less efficient

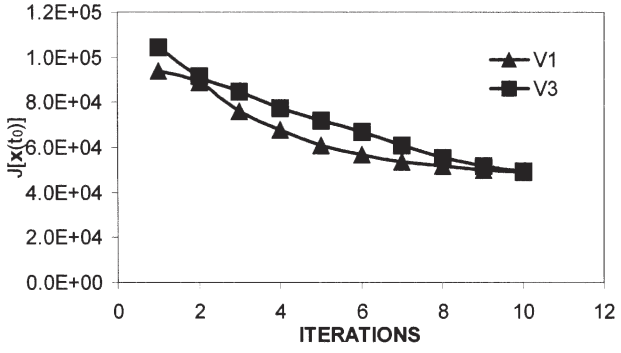

FIG. 3. Value of the cost function computed after each iteration using the MM5v1 (V1) and (V3) 4DVAR systems for the optimal analysis generated for 1800 UTC 10 Apr 1979 in the SESAME case study shown in Fig. 2.

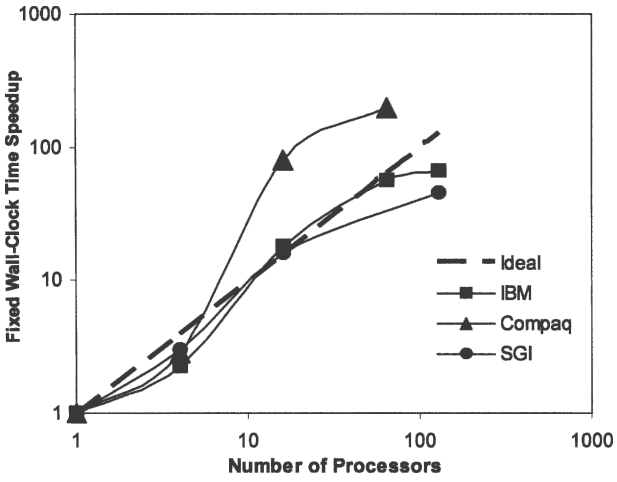

FIG. 4. Wall-clock speedup time as a function of the number of processors for the 152 × 136 grid point 1800 UTC 10 Apr 1979 SESAME case.

than on the IBM and SGI as evident by the one-processor runtimes of 248 533, 164 368, and 163 840 s for the Compaq, IBM, and SGI, respectively. Beyond four processors the problem is dominated by computational cost rather than I/O, so the Compaq has the advantage. However, once internal I/O is used, the speedups for all three of the machines behave similarly. Between 16 and 64 processors the runs on the Compaq and IBM exhibit speedup factors of approximately 3. It should be noted that since the SGI is a shared-memory machine it would theoretically be possible to run with internal I/O on when running on one or four processors. However, in reality, this would not be practical since having so much memory devoted to so few processors would mostly likely adversely impact other users of the computer. Running the problem using 64 processors on the Compaq ES-45, a set of optimal initial conditions were computed in approximately 20 min of wall-clock time.

5. Conclusions

Our goals in this effort were twofold. First, we wanted to create a new version of the MM5 TLM and adjoint that were more computationally efficient and maintainable. New versions of the MM5 TLM and adjoint compatible with version 3.4 of MM5 have been developed. Use of the TAMC code compiler greatly reduced the time to generate the new TLM and adjoint and, as noted in the migration from MM5v3.3 to MM5v3.4, substantially reduced the time needed to upgrade TLM and adjoint as the forecast model is updated. Following the approach successfully used for the MM5 NLM, the ability of running on multiple processors simultaneously has been added to the new codes to allow a faster time to solution when one has access to multiple processors. Runs on an IBM SP P3, Compaq ES-45, and SGI Origin 3800 show reasonable scaling efficiency when more processors are added to a fixed problem size. Of course, computer systems with different architectures will have different results. While in an operational setting domain size will be different, additional observational sources will need to be handled, and more iterations may be necessary, the wall-clock time execution of the MM5v3 4DVAR on the Compaq ES-45 using 64 processors certainly allows for the possibility of using 4DVAR to generate analyses in operational settings.

The second goal was to demonstrate the successful application of the automatic code generator for a TLM and adjoint. The application here was for MM5 but this approach should be equally applicable to other limited area models such as the Weather Research and Forecast (WRF) model. We believe that with the advent of improved adjoint compilers, improved software engineering, and the use of FORTRAN-90 employed in WRF that development of the distributed-memory version of the WRF adjoint should be as easy if not even easier than described here. In fact the WRF adjoint is currently being constructed at NCAR using the Transformation of Algorithms in FORTRAN (TAF) automatic adjoint compiler. TAF is the commercial successor to TAMC. It does remain to be seen if TAF could successfully handle calls to the parallelization libraries such as MPI and OpenMP. The TAF developers state that it can. While in the case of the MM5 TLM and adjoint some manual intervention was necessary due to the structure of the MM5 NLM code, it is possible to automatically generate TLM and adjoint codes as part of the forecast model compilation process (Marotzke et al. 1999). It would be wise for future model developers to keep in mind the requirements of the automatic adjoint compilers when coding new models so as to eliminate the necessity of maintaining separate TLM and adjoint codes.

When the WRF adjoint is complete it would be expected that most users would eventually migrate toward it and away from MM5. However, it will still be a few years before the WRF adjoint can approach the current capability of the MM5v3 adjoint. Until that time the new MM5v3 TLM and adjoint with demonstrated computational efficiency should allow researchers to pursue the general scientific issues involved with the use of the adjoint in 4DVAR systems, which should benefit future users of adjoints from other nonlinear models.

Work continues on the new MM5 4DVAR system. An incremental driver has been developed for it. The

incremental driver can reduce the time to solution in the procedure outlined originally by Courtier et al. (1994), where simplified physics or reduced resolution are used in an inner loop of iterations. The tropical cyclone bogus data assimilation scheme used by Zou et al. (2001) with the original MM5 4DVAR system has been ported to the new system. Work is ongoing to incorporate a moisture microphysics scheme as a precursor of an effort to develop observation operators for satellite cloud imagers.

People interested in the MM5 TLM and adjoint code described above are encouraged to contact the corresponding author for instructions on obtaining it.

Acknowledgments. This work was supported by the Department of Defense (DoD) High Performance Computing (HPC) Modernization Office's Common High Performance Computing Software Support Initiative. Computer time was provided by the DoD Major Shared Resource Centers of the Aeronautical Systems Center at Wright-Patterson AFB, Ohio, and the Army Research Laboratory at the Aberdeen Proving Ground, Maryland, and the DoD Distributed Resource Center at the Maui High Performance Computing Center, Kihei, Hawaii. Software development work was provided by Mark Cerniglia of Atmospheric and Environmental Research, Inc. The authors thank Al Bourgeois of NCAR for his help with generating the observation operators and the gradient solver used in this study.

APPENDIX

Examples of Problems TAMC Had Generating TLM and Adjoint Code

*a. An example of incorrect NLM dependency analysis: **BDYRST***

In this example, the dependency analysis of TAMC is incorrect because TAMC considers dependencies of entire arrays, not individual elements. The NLM code for this subroutine contains if statements that depend on the assignments to TA in the first two do loops (over j and i):

```
do nb1 = 1, lb
nb = nb1
is = nb1
js = nb1
ie = iend + 1 − nb1
je = jend + 1 − nb1
do k = 1, kl
do j = nb + 1, je − 1
ta(is, j, k) = tsb(j, k, nb) + dtbc*tten(is, j, k)
ta(ie, j, k) = tnb(j, k, nb) + dtbc*tten(ie, j, k)
enddo
do i = nb, ie
ta(i, js, k) = twb(i, k, nb) + dtbc*tten(i, js, k)
ta(i, je, k) = teb(i, k, nb) + dtbc*tten(i, je, k)
enddo
do j = nb + 1, je − 1
if(ta(is, j, k).lt.epsi)ta(is, j, k) = epsi
if(ta(ie, j, k).lt.epsi)ta(ie, j, k) = epsi
enddo
do i = nb, ie
if(ta(i, js, k).lt.epsi)ta(i, js, k) = epsi
if(ta(i, je, k).lt.epsi)ta(i, je, k) = epsi
enddo
enddo
enddo
```

If the "-pure" option was selected (in which NLM code is only generated if needed for the TLM), the TLM code generated by TAMC omitted the first recalculation of the TA variable. Without this option, the following correct code was generated (the lines omitted in the -pure version are in bold):

```
do nb1 = 1, lb
nb = nb1
is = nb1
js = nb1
ie = iend + 1 − nb1
je = jend + 1 − nb1
do k = 1, kl
do j = nb + 1, je − 1
g_ta(is, j, k) = g_tsb(j, k, nb) + g_tten(is, j, k) * dtbc
ta(is, j, k) = tsb(j, k, nb) + dtbc*tten(is, j, k)
g_ta(ie, j, k) = g_tnb(j, k, nb) + g_tten(ie, j, k)*dtbc
ta(ie, j, k) = tnb(j, k, nb) + dtbc*tten(ie, j, k)
end do
do i = nb, ie
g_ta(i, js, k) = g_tten(i, js, k)*dtbc + g_twb(i, k, nb)
ta(i, js, k) = twb(i, k, nb) + dtbc*tten(i, js, k)
g_ta(i, je, k) = g_teb(i, k, nb) + g_tten(i, je, k)*dtbc
ta(i, je, k) = teb(i, k, nb) + dtbc*tten(i, je, k)
end do
do j = nb + 1, je − 1
if (ta(is, j, k) .lt. epsi) then
g_ta(is, j, k) = 0.
ta(is, j, k) = epsi
endif
if (ta(ie, j, k) .lt. epsi) then
g_ta(ie, j, k) = 0.
ta(ie, j, k) = epsi
endif
end do
do i = nb, ie
```

```
if (ta(i, js, k) .lt. epsi) then
g_ta(i, js, k) = 0.
ta(i, js, k) = epsi
endif
if (ta(i, je, k) .lt. epsi) then
g_ta(i, je, k) = 0.
ta(i, je, k) = epsi
endif
end do
end do
end do
```

Why using the "-pure" option resulted in TAMC correctly including the second recalculation of TA but not the first is an interesting question. However, tracking down the cause of this would require experimentation and, more importantly, interaction with the developers of TAMC (since only the executable is available to users). This was not pursued, since an easy workaround (using the "-nopure") option existed.

b. *An example of incorrect NLM recomputation:* ***CUPARA2***

In this example, the errors in the NLM code were generated by go to statements characteristic of older FORTRAN codes, as found in the Kuo cumulus parameterization scheme of the MM5. The NLM contained the following code segments, in which the vertical loop limits for the cloud computations are derived by a computation of cloud base and height. TAMC encountered two problems in this case: 1) it generated unnecessary TLM code of the cloud base and top (these are more properly considered the NLM trajectory about which to linearize) and 2) crucial go to statements (in bold below) were missing, which led to incorrect loop limits:

```
c$$$ The preceding code computes the lifting
c$$$ condensation level sigma (SIGLCL) and the
c$$$ cloud and environmental values of
c$$$ saturation equivalent potential
c$$$ temperature (EQTM and SEQT)
   DO 220 K = 1, KL
   IF(A(K).GE.SIGLCL)GOTO 230
220 CONTINUE
230 CONTINUE
   kbase = k
   if (kbase .gt. kl) then
   kbase = kl
   endif
   do kk = 1, kbase
   k = kbase + 1 − kk
   DEQT = SEQT(K)-EQTM
   IF(DEQT.GT.DLT)GOTO 270
   end do
270 continue
   ktop = k
```

These errors were corrected manually.

c. *An example of inefficient NLM recomputations:* ***TRANSM***

In many of the MM5 physics subroutines all computations are performed independently for each vertical column, but are contained inside a loop over a horizontal grid index (i) for computational efficiency. In some cases the adjoint code generated by TAMC reversed the order of this i loop, and either inserted nested loops over i, or used local storage to recompute or restore the NLM for all previous horizontal grid points. For example, in the following NLM code atmospheric radiative transmissivities are computed from, among others, column precipitable water (PRW) computed previously:

```
ilx = iyy − 1
do i = 1, ilx
prw(i, j) = amin1(prw(i, j), 4.99)
prw(i, j) = amax1(prw(i, j), 0.01)
end do
do i = 1, ilx
c$$$ The following computes the output from prw
(code not shown)
end do
```

In the above, i and j are both horizontal grid indices. The TAMC-generated code contains storage and recomputations for all horizontal grid points (unnecessary statements are in bold):

```
do ip2 = 1, mjx
do ip1 = 1, mix
prwh(ip1, ip2) = prw(ip1, ip2)
end do
end do
ilx = iyy − 1
do i = 1, ilx
prw(i, j) = amin1(prw(i, j), 4.99)
prw(i, j) = amax1(prw(i, j), 0.01)
end do
do i = 1, ilx
c$$$ The following computes the
c$$$ adjoint of prw
(code not shown)
end do
do ip2 = 1, mjx
do ip1 = 1, mix
prw(ip1, ip2) = prwh(ip1, ip2)
```

```
end do
end do
do i = 1, ilx
prw(i, j) = amin1(prw(i, j), 4.99)
a_prw(i, j) = a_prw(i, j)*(0.5 + sign(0.5, prw(i, j) −
  0.01))
do ip2 = 1, mjx
do ip1 = 1, mix
prw(ip1, ip2) = prwh(ip1, ip2)
end do
end do
a_prw(i, j) = a_prw(i, j)*(0.5 + sign(0.5, 4.99 −
  prw(i, j)))
end do
```

This code was manually simplified as follows:

```
do ip1 = 1, mix
prwh(ip1) = prw(ip1, j)
end do
ilx = iyy − 1
do i = 1, ilx
prw(i, j) = amin1(prw(i, j), 4.99)
prw(i, j) = amax1(prw(i, j), 0.01)
end do
do i = 1, ilx
c$$$ The following computes computes the
c$$$ adjoint of prw
(code not shown)
end do
do i = 1, ilx
c$$$ Only restore prw for current i:
prw(i, j) = prwh(i)
prw(i, j) = amin1(prw(i, j), 4.99)
a_prw(i, j) = a_prw(i, j)*(0.5 + sign(0.5, prw(i, j) −
  0.01))
c$$$ Only restore prw for current i:
prw(i, j) = prwh(i)
a_prw(i, j) = a_prw(i, j)*(0.5 + sign(0.5, 4.99-prw(i,
  j)))
end do
```

REFERENCES

Alberty, R. L., D. W. Burgess, and T. T. Fujita, 1980: Severe weather events of 10 April 1979. *Bull. Amer. Meteor. Soc.,* **61,** 1033–1034.

Andersson, E., and Coauthors, 2005: Assimilation and modeling of the atmospheric hydrological cycle in the ECMWF forecast system. *Bull. Amer. Meteor. Soc.,* **86,** 387–402.

Anthes, R. A., and T. T. Warner, 1978: Development of hydrodynamic models suitable for air pollution and other mesometeorological studies. *Mon. Wea. Rev.,* **106,** 1045–1078.

Arakawa, A., and V. R. Lamb, 1977: Computational design of the basic dynamical process of the UCLA general circulation model. *Methods in Computational Physics,* Vol. 17, Academic Press, 173–265.

Asselin, R., 1972: Frequency filter for time integrations. *Mon. Wea. Rev.,* **100,** 487–490.

Blackadar, A. K., 1979: High resolution models of the planetary boundary layer. *Advances in Environmental Science and Engineering,* Vol. 1, J. R. Pfallin, and E. N. Ziegler, Eds., Gordon and Breach, 50–85.

Courtier, P., J.-N. Thépaut, and A. Hollingsworth, 1994: A strategy for operational implementation of 4DVAR, using an incremental approach. *Quart. J. Roy. Meteor. Soc.,* **120,** 1367–1378.

Deardorff, J. W., 1972: Parameterization of the planetary boundary layer for use in general circulation models. *Mon. Wea. Rev.,* **100,** 93–106.

Dudhia, J., 1989: Numerical study of convection observed during the Winter Monsoon Experiment using a mesoscale two-dimensional model. *J. Atmos. Sci.,* **46,** 3077–3107.

Errico, R. M., and K. D. Raeder, 1999: An examination of the accuracy of the linearization of a mesoscale model with moist physics. *Quart. J. Roy. Meteor. Soc.,* **125,** 169–195.

Gauthier, P., and J.-N. Thépaut, 2001: Impact of the digital filter as a weak constraint in the preoperational 4DVAR assimilation system of Météo-France. *Mon. Wea. Rev.,* **129,** 2089–2102.

Gérard, É., and R. Saunders, 1999: Four-dimensional variational assimilation of Special Sensor Microwave/Imager total column water vapour in the ECMWF model. *Quart. J. Roy. Meteor. Soc.,* **125,** 3077–3101.

Giering, R., 1999: Tangent Linear and Adjoint Model Compiler. Users manual 1.4, 64 pp. [Available online at http://www.autodiff.com/tamc/document.html.]

——, and T. Kaminski, 1998: Recipes for adjoint code construction. *Assoc. Comput. Mach. Trans. Math. Software,* **24,** 437–474.

Grell, G. A., 1993: Prognostic evaluation of assumptions used by cumulus parameterizations. *Mon. Wea. Rev.,* **121,** 764–787.

——, J. Dudhia, and D. R. Stauffer, 1995: A description of the fifth-generation Penn State/NCAR Mesoscale Model (MM5). NCAR Tech. Note NCAR/TN-398-STR, 122 pp. [Available from UCAR Communications, P.O. Box 3000, Boulder, CO 80307.]

Ide, K., P. Courtier, M. Ghil, and A. C. Lorenc, 1997: Unified notation for data assimilation: Operational, sequential and variational. *J. Meteor. Soc. Japan,* **75B,** 181–189.

Kaminski, T., R. Giering, and M. Heimann, 1996: Sensitivity of the seasonal cycle of CO_2 at remote monitoring stations with respect to seasonal surface exchange fluxes determined with the adjoint of an atmospheric transport models. *Phys. Chem. Earth,* **21,** 457–462.

Kleist, D. T., and M. C. Morgan, 2005: Interpretation of the structure and evolution of adjoint-derived forecast sensitivity gradients. *Mon. Wea. Rev.,* **133,** 466–484.

Klemp, J. B., and R. B. Wilhelmson, 1978: The simulation of three-dimensional convective storm dynamics. *J. Atmos. Sci.,* **35,** 1070–1096.

Kuo, H. L., 1974: Further studies of the influence of cumulus convection on larger-scale flow. *J. Atmos. Sci.,* **31,** 1232–1240.

Kuo, Y.-H., X. Zou, and Y.-R. Guo, 1996: Variational assimilation of precipitable water using a nonhydrostatic mesoscale adjoint model. Part I: Moisture retrieval and sensitivity experiments. *Mon. Wea. Rev.,* **124,** 122–147.

Le Dimet, F. X., and O. Talagrand, 1986: Variational algorithms

for analysis and assimilation of meteorological observations: Theoretical aspects. *Tellus,* **38A,** 97–110.

Liu, D. C., and J. Nocedal, 1989: On the limited memory BFGS method for large scale optimization. *Math. Programm.,* **45,** 503–528.

Lorenc, A. C., 2003: The potential of the ensemble Kalman filter for NWP—A comparison with 4DVAR. *Quart. J. Roy. Meteor. Soc.,* **129,** 3183–3203.

Mahfouf, J.-F., 1999: Influence of physical processes on the tangent-linear approximation. *Tellus,* **51A,** 147–166.

Marotzke, J. R., R. Giering, K. Q. Zang, D. Stammer, C. Hill, and T. Lee, 1999: Construction of the adjoint of the MIT ocean general circulation model and application to Atlantic heat transport sensitivity. *J. Geophys. Res.,* **104,** 29 529–29 547.

Mass, C. F., and Y.-H. Kuo, 1998: Regional real-time numerical weather prediction: Current status and future potential. *Bull. Amer. Meteor. Soc.,* **79,** 253–263.

Michalakes, J., 1997a: FLIC: A translator for same-source parallel implementation of regular grid applications. Mathematics and Computer Science Division Tech. Rep. ANL/MCS-TM-223, Argonne National Laboratory, Argonne, IL, 11 pp.

——, 1997b: RSL: A parallel runtime system library for regional atmospheric models with nesting. *Structured Adaptive Mesh Refinement (SAMR) Grid Methods,* S. Baden et al., Eds., IMA Volumes in Mathematics and its Applications, Vol. 117, Springer, 59–74.

——, 2000: The same source parallel MM5. *Sci. Programm.,* **8,** 5–12.

Rabier, F., H. Järvinen, E. Klinker, J.-F. Mahfouf, and A. Simmons, 2000: The ECMWF operational implementation of four-dimensional variational assimilation. I: Experimental results with simplified physics. *Quart. J. Roy. Meteor. Soc.,* **126A,** 1143–1170.

Sun, J., 2005: Initialization and numerical forecast of a supercell storm observed during STEPS. *Mon. Wea. Rev.,* **133,** 793–813.

Veersé, F., and J.-N. Thépaut, 1998: Multiple-truncation incremental approach for four-dimensional variational data assimilation. *Quart. J. Roy. Meteor. Soc.,* **124,** 1889–1908.

Vukicevic, T., T. Greenwald, M. Zupanski, D. Zupanski, T. Vonder Haar, and A. S. Jones, 2004: Mesoscale cloud state estimation from visible and infrared satellite radiances. *Mon. Wea. Rev.,* **132,** 3066–3077.

Ware, R., R. Carpenter, J. Güldner, J. Liljegren, T. Nehrkorn, F. Solheim, and F. Vandenberghe, 2003: A multichannel radiometric profiler of temperature, humidity, and cloud liquid. *Radio Sci.,* **38,** 8079, doi:10.1029/2002RS002856.

Wee, T.-K., and Y.-H. Kuo, 2004: Impact of a digital filter as a weak constraint in MM5 4DVAR: An observing system simulation experiment. *Mon. Wea. Rev.,* **132,** 543–559.

Zou, X., F. Vandenberge, M. Pondeca, and Y.-H. Kuo, 1997: Introduction to adjoint techniques and the MM5 adjoint modeling system. NCAR Tech. Note NCAR/TN-435-STR, 107 pp. [Available from UCAR Communications, P.O. Box 3000, Boulder, CO 80307.]

——, Q. Xiao, A. E. Lipton, and G. D. Modica, 2001: A numerical study of the effect of GOES sounder cloud-cleared brightness temperatures on the prediction of Hurricane Felix. *J. Appl. Meteor.,* **40,** 34–55.

Zupanski, M., and D. Zupanski, 2002: Four-dimensional variational data assimilation for the blizzard of 2000. *Mon. Wea. Rev.,* **130,** 1967–1988.