Sensitivity Analysis of the MM5 Weather Model Using Automatic Differentiation

Christian H. Bischof
Math. & Comp. Sc. Div.
Argonne National Laboratory
Argonne, IL 60439
bischof@mcs.anl.gov

Gordon D. Pusch
Physics Department
Michigan State University
East Lansing, MI 48842
pusch@mcs.anl.gov

Ralf Knoesel
Math. & Comp. Sc. Div.
Argonne National Laboratory
Argonne, IL 60439
knoesel@mcs.anl.gov

Argonne Preprint MCS-P532-0895

Abstract

We present a general method for using automatic differentiation to facilitate model sensitivity analysis. Automatic differentiation techniques augment, in a completely mechanical fashion, an existing code such that it also simultaneously and efficiently computes derivatives. Our method allows the sensitivities of the code's outputs to its parameters and inputs to be determined with minimal human effort by exploiting the relationship between differentiation and formal perturbation theory.

Employing this method, we performed a sensitivity study of the MM5 code, a mesoscale weather model jointly developed by Penn State University and the National Center for Atmospheric Research, comprising roughly 40,000 lines of Fortran 77 code. Our results show that AD-computed sensitivities exhibit superior accuracy compared with divided differences approximations computed from finite-amplitude perturbations, while consuming comparable or less CPU time and less human labor. We also comment on a numerically induced precursor wave that would almost certainly have been undetectable if one used a divided difference method.

1 Introduction

In this paper, we present a general approach for automating the sensitivity analysis of a computer model using automatic differentiation. Automatic differentiation (AD) techniques augment, in a completely mechanical fashion, an existing code such that it also simultaneously and efficiently computes derivatives; hence, one may obtain the partial derivatives (also referred to as "sensitivities") of the code's outputs with respect to its parameters and inputs with minimal human effort. An AD tool mechanically generates from an existing code an augmented code that in addition computes derivatives by a systematic application of the chain rule to each elementary operation and function [18, 14]. Unlike divided difference (DD) approximations, AD-computed sensitivities are accurate to machine precision, whereas DD estimates suffer from an unavoidable tradeoff between truncation and cancellation error [12, 17].

To demonstrate the utility of our approach, we present two results from a sensitivity study in which we used the ADIFOR tool [2, 3, 4] to construct the tangent linear model (TLM) of the MM5 fifth generation mesoscale weather model. MM5 is a 3-D limited area (that is, mesoscale and regional scale) finite-difference weather model. It is capable of both hydrostatic and nonhydrostatic weather simulation and prediction, and it contains numerous submodels of various microscale and subgrid-scale meteorological processes [13, 10]. MM5 has been developed jointly by the Penn State University (PSU) Meteorology Department and the National Center for Atmospheric Research (NCAR) as a community mesoscale model and is continuously being improved by its many users at universities and government laboratories. The MM5 code consists of roughly 40,000 lines of Fortran 77 code.

The tangent linear model of a system results from linearizing that system about some specified <u>reference solution</u> or <u>base</u> [6, 11, 21]. The TLM may also be obtained by applying formal first-order perturbation theory to the system; furthermore, it can be shown that the first-order perturbation of any prognostic or diagnostic variable may be obtained by evaluating the derivative of that variable with respect to the perturbation parameter. Therefore, one can generate the TLM of a complex code with minimal human labor by parameterizing the perturbation of interest, then differentiating with respect to that parameter by using an AD tool. This approach enables one to perform sensitivity studies on a complex code quickly, easily, and accurately; by contrast, more conventional approaches to sensitivity analysis may require substantial human effort to "reverse engineer" and modify the code, yet achieve less accurate results.

The outline of this paper is as follows. In Section 2, we discuss the tangent linear model; we then show the connection of the TLM to first-order perturbation theory and to differentiation. In Section 3, we give a brief introduction to automatic differentiation and comment on its superiority to divided-difference approximations of derivatives. We then show how to easily construct the TLM by using AD to differentiate with respect to a scalar perturbation parameter. In Section 4, we describe the domain and initial conditions used for our unperturbed reference trajectory, and we discuss the physical meaning of our chosen initial perturbation. We then present a dramatic illustration of the superior accuracy that may be obtained by using AD-computed sensitivities instead of divided-difference approximations. Finally, we discuss one of the phenomena we observed in the TLM output that would almost certainly have gone undetected had we used a divided-difference approach: a numerically induced "superacoustic" precursor wave.

2 The Tangent Linear Model and Differentiation

Whether presented as a table of results from a set of exploratory runs or formally encapsulated as a sensitivity coefficient, most parametric sensitivity studies either are equivalent to or can be embedded in a problem of *estimating derivatives.* By differentiating the output of a model with respect to its parameters, one can quantify how sensitive or robust the model's predictions are relative to variations of that parameter, as well as gain insight into how to adjust parameters that are poorly known. Questions regarding the sensitivity of the model output to more abstract quantities involving many model variables can also often be rephrased in terms of derivatives, either directly or by embedding the problem of interest into a larger parameterized framework (for example, by using a continuation [19] or invariant imbedding method [22]). Our approach is an example of this latter approach: we obtain the TLM evolution of a perturbation in the initial-value data by introducing a parameter that linearly interpolates between the unperturbed and perturbed initial states. We shall show that formal perturbation theory with respect to the parameter yields the TLM and can be shown to be equivalent to evaluation of the derivative with respect to the interpolating parameter.

2.1 An Example of the TLM for Discrete Time

The tangent linear model describes the linearized evolution of errors or perturbations about the trajectory of some particular base or reference solution. Its name is derived from the fact that a solution to the TLM always falls somewhere on the instantaneous tangent to the reference solution as both of them evolve in time [11, 21]. To illustrate the TLM, we consider a discretized-time example. Assume that the vector $X(t)$ describing the state of the system at time t satisfies the simple equation

$$X(t_{k+1}) = H(X(t_k)), \quad k = 0, \ldots, f. \tag{1}$$

For the sake of notational simplicity, we shall lump into X not only all the prognostic variables, but also the diagnostic variables and model parameters, and treat X as if it were a single column vector, rather than a collection of multidimensional arrays. Note also that, while MM5 uses a leapfrog scheme wherein $X(t_{k+1})$ depends on the state of *two* preceding time slices, $X(t_k)$ and $X(t_{k-1})$, in (1) we have written that $X(t_{k+1})$ depends only on $X(t_k)$; however this should not be cause for concern. It is always possible to rewrite a second-order difference equation $\xi(t_{k+1}) = g(\xi(t_k), \xi(t_{k-1}))$ as two first-order difference equations of the form (1), by regarding the pair $(\xi(t_k), \xi(t_{k-1}))$ as a single set of variables $X(t_k)$ defining the state at t_k, and adjoining the trivial equation $\xi(t_k) = 1*\xi(t_k)+0*\xi(t_{k-1})$.

Let

$$J(t_k) := \frac{\partial X(t_{k+1})}{\partial X(t_k)} = \left.\frac{\partial H}{\partial X}\right|_{X=X(t_k)} \tag{2}$$

denote the $n \times n$ Jacobian of H with respect to the state at time step k. Let $\delta X(t)$ denote the sensitivity of the state $X(t)$ at time t with respect to a perturbation $\delta X(t_0)$ of its initial state $X(t_0)$; that is,

$$\delta X(t) = \frac{\partial X(t)}{\partial X(t_0)} * \delta X(t_0), \tag{3}$$

where "$*$" denotes vector-matrix multiplication, and $\delta X(t)$ and $\delta X(t_0)$ should be interpreted as column vectors. Then by (1), (2), (3), and the associativity of the chain-rule,

$$\begin{aligned} \delta X(t_1) &= J(t_0) * \delta X(t_0) \\ \delta X(t_2) &= J(t_1) * \delta X(t_1) = J(t_1) * J(t_0) * \delta X(t_0) \\ &\vdots \\ \delta X(t_k) &= J(t_{k-1}) * \delta X(t_{k-1}) = J(t_{k-1}) * \cdots * J(t_0) * \delta X(t_0). \end{aligned} \tag{4}$$

Note that the sensitivity is propagated by the *same* chain of Jacobians regardless of the initial perturbation $\delta X(t_0)$. That is, the sensitivity is a *linear* function of the initial perturbation; however, this function is determined entirely by the *unperturbed* trajectory, not the perturbed one. In other words, the sensitivity is really a property of the *trajectory,* rather than the perturbation.

Let $X_0(t)$ be a particular solution of (1), which we shall call the *base,* or *reference trajectory,* and let $X(t)$ be a perturbed nonlinear solution that is initially close to $X_0(t)$. The tangent linear approximation $X_{TLM}(t) \approx X(t)$ relative to the base state $X_0(t)$ is defined as

$$X_{TLM}(t) := X_0(t) + \delta X(t) = X_0(t) + \prod_{i=0}^{k-1} J(t_i) * \delta X(t_0), \quad \delta X(t_0) := X(t_0) - X_0(t_0); \tag{5}$$

that is, the TLM solution is a first-order Taylor approximation to the perturbed nonlinear solution, expanded about the base state. Because the Jacobian J generalizes the notion of the slope or tangent of a curve to an n-dimensional space, and since the TLM solution is a linear function of $\delta X(t_0)$, it follows that a solution to the TLM always falls somewhere on the *instantaneous tangent to the reference solution* as both of them evolve in time—hence the term "tangent linear model" [11, 21].

Equation (5) represents the general solution of the TLM given an arbitrary $\delta X(t_0)$. The choice of $\delta X(t_0)$ is at the disposal of the analyst; particular solutions corresponding to specific choices of $\delta X(t_0)$ may have particularly simple or useful interpretations. For example, by initializing $\delta X_i(t_0) = 1$, $\delta X_j(t_0) = 0$ for $j \neq i$, $\delta X(t_f)$ can be interpreted as the sensitivity of the final state vector $X(t_f)$ per unit change in the ith component $X_i(t_0)$ of the initial state. For a general choice $\delta X(t_0)$, the interpretation of $\delta X(t_f)$ is given by the concept of the *directional derivative*—the rate of change of $X(t_f)$ per unit change of $X(t_0)$ in the direction of $\delta X(t_0)$.

2.2 The TLM and Formal Perturbation Theory

Let us consider a perturbed initial state

$$X(t_0; \epsilon, \delta X(t_0)) := X_0(t_0) + \epsilon\, \delta X(t_0), \tag{6}$$

where we have split the perturbation into two factors—a spatially varying function δX that determines its "shape," and a scale factor ϵ parameterizing its amplitude; we use the semicolon to emphasize that ϵ and δX represent *parameters* of the function X, rather than arguments of it. Applying update-equation (1) to (6),

$$X(t_{k+1};\, \epsilon, \delta X(t_0)) := H(\, X(t_k;\, \epsilon, \delta X(t_0))\,), \tag{7}$$

we generate a parameterized family of perturbed solutions containing the unperturbed solution at $\epsilon = 0$. We then define the *first-order perturbation-theoretic sensitivity* of $X(t;\, \epsilon, \delta X(t_0))$, to be the *value of its partial derivative with respect to* ϵ*, at* $\epsilon = 0$:

$$\delta X(t;\, \delta X(t_0)) := \left. \frac{\partial\, X(t;\, \epsilon, \delta X(t_0))}{\partial\, \epsilon} \right|_{\epsilon=0}. \tag{8}$$

Note that $\delta X(t;\, \delta X(t_0))$ *does not depend on* ϵ. As we stated in section 2.1, the sensitivity is a property of the *unperturbed* trajectory, which corresponds to $\epsilon = 0$ in (6); therefore, it follows that only sensitivities that are evaluated at $\epsilon = 0$ are meaningful.

To motivate our claim that the TLM and first-order perturbation theory are equivalent, let us temporarily consider $X(t_k; \epsilon, \delta X(t_0))$ for $\epsilon \neq 0$. Taylor expanding (7) about the reference state $X(t_k; 0, \delta X(t_0))$, after using (8), we obtain

$$\begin{aligned} & X(t_{k+1}; 0, \delta X(t_0)) + \epsilon \delta X(t_{k+1}; \delta X(t_0)) + O(\epsilon^2) \\ & \quad = \quad H\left(X(t_k; 0, \delta X(t_0)) + \epsilon\, \delta X(t_k, \delta X(t_0)) + O(\epsilon^2)\right) \\ & \quad = \quad H(X(t_k; 0, \delta X(t_0))) + \epsilon \left. \frac{\partial H}{\partial X} \right|_{X=X(t_k)} * \delta X(t_k; \delta X(t_0)) + O(\epsilon^2). \end{aligned} \tag{9}$$

Separately equating like powers of ϵ in (9), we see that the zeroth-order coefficients merely reproduce (1), the evolution equation for the base state. The first-order coefficients, after use of (2), satisfy

$$\delta X(t_{k+1}; \delta X(t_0)) = \left. \frac{\partial H}{\partial X} \right|_{X=X(t_k)} * \delta X(t_k; \delta X(t_0)) = J(t_k) * \delta X(t_k; \delta X(t_0)). \tag{10}$$

One sees that equation (10) has the same form as the evolution equation (4) for the TLM, if one identifies $\delta X(t_k)$ with $\delta X(t_k; \delta X(t_0)) = \partial\, X(t_k; \epsilon, \delta X(t_0)) \,/\, \partial\, \epsilon\, |_{\epsilon=0}$. One can make the connection between formal perturbation theory and repeated differentiation even clearer by noting that the Taylor expansion of (7) with respect to ϵ is (9), which can be differentiated term by term. Differentiating (9) with respect to ϵ annihilates the "constant" term and lowers the order of all remaining terms by one. Setting $\epsilon = 0$ causes every term except the constant to vanish. Hence, differentiating (7) n times with respect to ϵ and setting $\epsilon = 0$ is equivalent to picking off the nth term of (9). This both justifies definition (8), and extends it to arbitrary order: the nth-order sensitivity of any quantity is defined as its nth partial with respect to ϵ, evaluated at $\epsilon = 0$ [16, chap. 2].

Note also that this approach is not limited to simple perturbations of the form (6). For example, one could write a more general perturbation

$$X(t_0;\, \epsilon_A, \epsilon_B, \delta X_A, \delta X_B) := X_0 + \epsilon_A\, \delta X_A + \epsilon_B\, \delta X_B, \tag{11}$$

where X_0 is the unperturbed initial state, and δX_A and δX_B are, for example, shape functions defining two "bumps" in the prognostic variables centered on two points A and B, respectively. Then, by differentiating with respect to both ϵ_A and ϵ_B, one can generate code that simultaneously computes the sensitivities to the bumps at points A and B in a single run.

3 Automatic Differentiation

Automatic differentiation techniques [18, 14] rely on the fact that every function, no matter how complicated, is executed on a computer as a sequence of elementary operations, such as addition or multiplication, and elementary functions, such as square root or log. By applying the chain rule of differential calculus,

$$\frac{\partial}{\partial t} f(g(t))\Big|_{t=t_0} = \left(\frac{\partial}{\partial s} f(s)\Big|_{s=g(t_0)} \right) * \left(\frac{\partial}{\partial t} g(t)\Big|_{t=t_0} \right), \tag{12}$$

to each stage in this sequence of elementary operations, one can compute derivatives of f exactly and in a completely mechanical fashion. For example, the short code segment

```
y = sin(x)
z = y*x + 5
```

upon being augmented to compute derivatives becomes

```
y = sin(x)
∂y = cos(x)*∂x
z = y*x + 5
∂z = y*∂x + x*∂y ,
```

where ∂var is a vector representing the derivatives of var with respect to the independent variable(s). Thus, if `x` is the scalar independent variable, then `∂x` is equal to 1 and `∂z` represents $\partial z/\partial x$. (The above example uses the *forward mode* of automatic differentiation, wherein one propagates the derivatives of intermediate variables with respect to the independent variables; by contrast, in *reverse mode* automatic differentiation one propagates the derivatives of dependent variables with respect to the intermediate variables. Reverse mode is related to the *adjoint model* in the same way that forward mode is related to the TLM [6].)

Several tools have been developed that use automatic differentiation to generate derivative codes. A compilation of currently available AD tools can be found on the World-Wide Web at the site `http://www.mcs.anl.gov/Projects/autodiff/AD_Tools` or in [5].

We employed the ADIFOR tool in our experiments. ADIFOR (Automatic DIfferentiation of FORtran) [2, 3, 4] performs automatic differentiation of programs written in Fortran 77. Given a collection of Fortran subroutines and a specification of which variables in its parameter lists or common blocks correspond to the "independent" and "dependent" variables for differentiation, ADIFOR produces portable Fortran 77 code that allows the computation of the derivatives of the dependent variables with respect to the independent ones.

Automatic differentiation yields the values of the analytical derivatives, exact to machine precision, without truncation error, and without any adjustable parameters such as step-size to be determined. By contrast, to obtain accurate results using a divided difference approximation, one must determine a step size that optimizes the "built-in" tradeoff between minimizing the truncation and the cancellation error—and even if one expends the additional computational effort to balance these two errors, a significant loss of precision will still be incurred [12, 17]. AD yields derivatives whose accuracy is uniform over the entire simulation domain, and almost always comparable to that of the function itself at all simulation times; by contrast, a DD step size that is optimal at one point or instant may not be optimal elsewhere, since the state variables are both space and time dependent. AD-enhanced code can compute all the desired derivatives in a single run, and these derivatives are available immediately; by contrast, to compute n divided differences, one needs to make n perturbed runs, plus an additional postprocessing step to compute the differences. Finally, AD-enhanced code generated by tools like ADIFOR can achieve all these advantages over divided differences while still only requiring a CPU-time comparable to the total CPU-time that would have been consumed by the set of perturbed runs [2, 3].

The previous discussion suggests that a simple mechanism for generating the TLM would be to use an AD preprocessor like ADIFOR to compute the perturbation expansion as follows:

1. Introduce new variables ϵ and δX_i.

2. Modify the code to initialize the ith component of the initial state to $X_i(t_0) + \epsilon\, \delta X_i$.

3. Differentiate the resulting code with respect to ϵ.

4. Execute the derivative-enhanced code, initializing ϵ to zero, and $\partial\epsilon$ to unity.

In accordance with (8), step 4 ensures that the sensitivities are evaluated on the unperturbed reference trajectory and represent derivatives with respect to the amplitude of the perturbation. This approach requires minimal modifications to the original code and is ideally suited for automatic differentiation.

4 Experimental Results

To demonstrate the utility of our methodology, we have analyzed the sensitivity of an MM5 forecast to its initial conditions using a "real world" data set. Below, we describe the simulation domain and data set used, and the initial perturbation assumed. We then present two results from this study: a dramatic illustration of the superior accuracy that may be obtained by using AD-computed sensitivities instead of divided-difference approximations, and the observation of a numerical precursor wave that would almost certainly have gone undetected had we used a divided difference approach.

4.1 Domain, Grid, and Initial Conditions

The simulation domain is centered on the Korean peninsula. The initial-condition data is defined on a $61 \times 61 \times 23$ grid; the grid is rectangular and uniform when viewed in stereographic projection from the North Pole, and true at 60 degrees north latitude. The mean gridpoint spacing is roughly 101 km; however, as a result of the projection, the local spacing varies significantly over the simulation domain. The simulation begins at 1994-Nov-12, 1200 UTC; this choice of initial data is representative of a typical winter day in Southeast Asia. We used a timestep of five minutes and terminated the run at 1800 UTC simulation-time, having simulated a total of six hours of weather.

4.2 Initializing the Perturbation

By parameterizing our perturbed state-variables as $X + \epsilon\, \delta X$ and following the prescription of section 3, we naturally ensure that the structures of X and δX are identical, and also that a corresponding finite-amplitude perturbation can easily be obtained by simply setting $\epsilon \neq 0$ and running the undifferentiated code. Thus, we can use a single source-code for both the finite-amplitude-perturbation executable, and as input for sensitivity enhancement by our AD tool. As stated in section 2.1, the choice of the initial-state perturbation $\delta X(t_0)$ is at the disposal of the analyst; a general choice will yield a directional derivative, whereas a particular choice may yield a sensitivity with a particularly simple or useful interpretation.

In our study, we were interested in sensitivities of the forecast generated by MM5 with respect to the initial data. Direct differentiation of the final state with respect to all of the elements of the initial state would consume a run time on the order of N times the unenhanced code, to yield an $N \times N$ sensitivity array, where N is the total number of grid points; however, not only would this represent a prohibitive expenditure of memory and CPU time, but in most practical cases, such highly detailed sensitivity information is not needed. By investing some initial consideration into determining what one's actual observables are, what sensitivities are actually of interest, and what large scale physical phenomenon might be expected to dominate the model sensitivities, one may often reap a substantial savings by a choosing a smaller but "smarter" set of directional derivatives to be computed, instead of exhaustively computing a mathematically complete set. For example, in weather simulation, the *physical* input data are not the gridded values, but the observations from the network of surface meteorological stations—and these are orders of magnitude less numerous

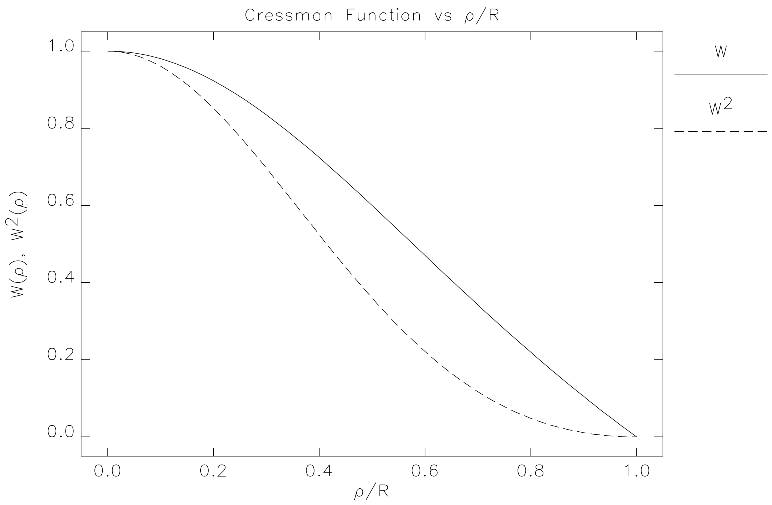

Figure 1: The Cressman function and its square.

than the computational gridpoints. Hence, it is the actual observations one should differentiate with respect to, and therefore one should choose a perturbation representative of how the observations influence the initial data on the grid.

The initial condition data for MM5 are constructed by a suite of preprocessors [15] by using a modified Cressman objective analysis scheme [9, 1]. *Objective analysis* is the term meteorologists use for any method of interpolating a set of irregularly spaced weather observations onto a regular computational grid. Most objective analysis algorithms use some form of spatial proximity-weighting to correct a "first guess" initial-data field until it agrees with the observations upon interpolating back to each measurement station (the "first guess" data might, for example, be the most recent forecast). The preprocessors also impose various consistency and balance checks on the initial data.

In performing our sensitivity study, we chose to perturb the initial temperature, since it impacts on every other prognostic variable, yet it is largely decoupled from the consistency and balance conditions imposed on the pressure and wind; hence we need only simulate the objective analysis procedure itself. This simulation is straightforward because the modified Cressman update formulas are *linear* in the observed data [15, 9, 1]; hence, the impact of perturbing a single temperature observation will be to perturb the initial conditions at each grid point in the vicinity of the selected observation in proportion to the value of its proximity-weighting factor. From the formulae given in the MM5 preprocessor manual [15], we have determined that a good approximation to this perturbation may be obtained by assigning the initial temperature to $T + \epsilon\, \delta T$, where T is the unperturbed temperature field, ϵ is the amplitude of the perturbation in degrees Kelvin, and the shape function δT is defined as

$$\delta T := \sigma\, W(\rho, R(\sigma)). \tag{13}$$

Here, $W(\rho, R)$ is the horizontal weighting function, ρ is the horizontal distance from the perturbation center, $R(\sigma)$ is the radius of influence of the measured datum, and σ is the terrain-following vertical coordinate used by MM5. We define σ as

$$\sigma := \frac{p_o(z) - p_{\rm top}}{p_{\rm s}(x, y) - p_{\rm top}}, \tag{14}$$

where $p_o(z)$ is the nominal pressure of a stationary equilibrium reference atmosphere at altitude z, $p_S(x, y)$ is the nominal pressure of the reference atmosphere at the surface point with horizontal coordinates (x, y), and p_{top} is the nominal pressure at the top of the simulation volume; σ therefore varies from unity at the surface, to zero at the top. The horizontal weighting factor $W(\rho, R)$ was chosen to be the *Cressman function,* defined as [9]

$$W(\rho, R) := \begin{cases} \dfrac{R^2 - \rho^2}{R^2 + \rho^2}, & \rho < R, \\ 0, & \rho \geq R, \end{cases} \tag{15}$$

which is a bell-shaped function of ρ/R; Figure 1 shows both W and W^2. The scaling by σ in perturbation (13) reflects the fact that a measurement at ground level should have progressively less influence on the estimated conditions at increasingly higher altitudes. The radius of influence R of the perturbation also depends linearly on σ, as $R(\sigma) = R_0\,[2 - \sigma]$; hence R ranges from R_0 at the surface to $2R_0$ at the top, so that the perturbed volume is shaped like an upside-down truncated cone. This shape reflects the fact that, while a measurement taken at ground-level should be relevant only to its immediate neighborhood at the surface, the estimated conditions at high altitude must be inferred from many surface observations scattered over a large area. We choose R_0 to be three grid point spacings in our simulation, corresponding to roughly a 300 km radius of influence.

4.3 Comparison of an AD-computed Derivative with a DD Approximation

The results in this section demonstrate the dramatic effect automatic differentiation can make in obtaining accurate and reliable derivatives. Figures 2 and 3 show the temperature sensitivity as computed at time step 12 (60 min) by AD and by DD, respectively. Figure 3 was obtained by using a second-order forward difference approximation to the desired sensitivity (we chose forward rather than central differencing because it allowed us to reuse the data from previous runs while searching for the optimum finite-amplitude perturbation size). After numerous runs, we determined by trial and error that the optimum step-sizes were near one and two degrees Kelvin; note that these perturbations are rather large compared with the mean value of the data (on the order of 0.3% and 0.6% of the data, respectively), and that they are optimal only in some average sense, since the temperature data are space dependent.

We see that the AD and DD results agree closely in appearance. However, the DD approximation suffers from substantial loss of accuracy because all but the last three decimal digits of the mantissas cancel; this manifests itself in a mottled appearance to the DD frame. Choosing a smaller amplitude perturbation exacerbates this problem. The figures clearly demonstrate the inadequacy of divided differences for computing sensitivities of complicated simulations involving many hundreds or thousands of floating-point operations at each grid point per time step.

4.4 "Superacoustic" Precursor Wave

A weak "precursor" wave was observed to propagate anisotropically and at superacoustic speeds; it was first noted through the fact that at the end of the first time step, the central 0.39 deg/deg positive peak of the temperature sensitivity was encircled by a shallow negative "moat" of maximum depth -3.2×10^{-3} deg/deg, surrounded in turn by a low positive embankment of 2.2×10^{-3} deg/deg, which fell off to extremely small values beyond a radius of about 10.5 grid points—that is, nearly six grid points larger than the 4.61 grid-point radius of the initial perturbation on this sigma-level. Since the speed of sound is only about one grid point per time step, this is at least five grid points further from the perturbation than any disturbance should have had time to travel. During subsequent steps, the precursor rapidly and anisotropically propagated away from the perturbation—initially about Mach 6, falling asymptotically to about Mach 4. It had a much smaller amplitude than the

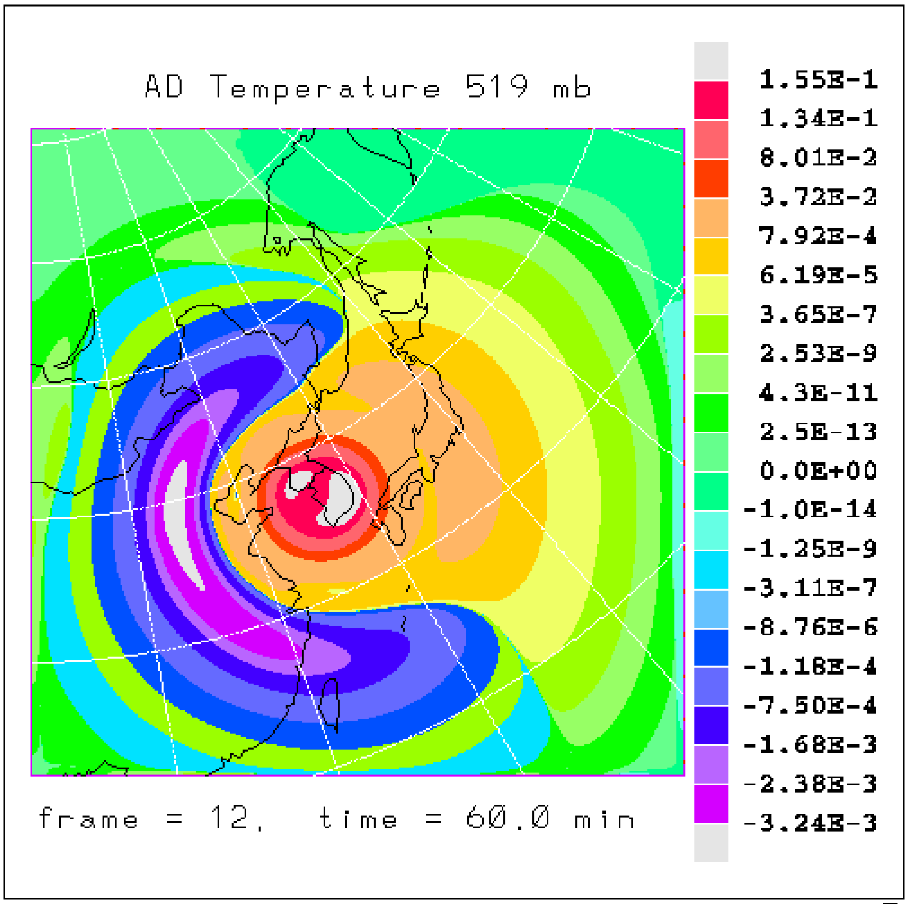

Figure 2: AD-computed 519 mb temperature sensitivity in deg/deg, at t = 60 min. Observe the anisotropic, low-amplitude precursor extending far beyond the 15 grid-point radius to which sound has had time to propagate (the simulation domain is 61×61 gridpoints), as well as the bipolar, crescent-shaped gravity waves emerging from the central peak. (Contour levels have been chosen by a histogrammic equalization method such that the intervals all occupy approximately equal areas; note the strong nonlinearity of the interval spacing.)

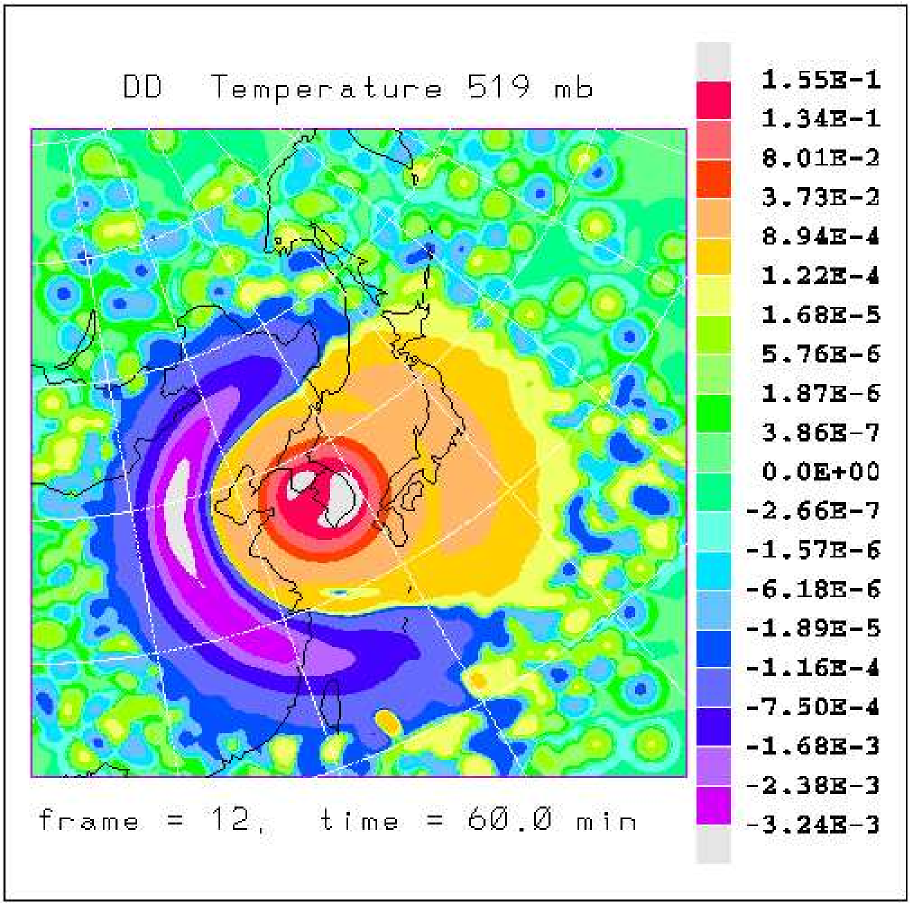

Figure 3: DD-computed 519 mb temperature sensitivity in deg/deg, at t = 60 min. Observe that, while the bipolar gravity-wave arcs are visible, the precursor is lost in the finite-precision noise. (Contour levels have also been chosen by histogrammic equalization; note the much more coarsely spaced small-amplitude contour levels used, in comparison with Figure 2.)

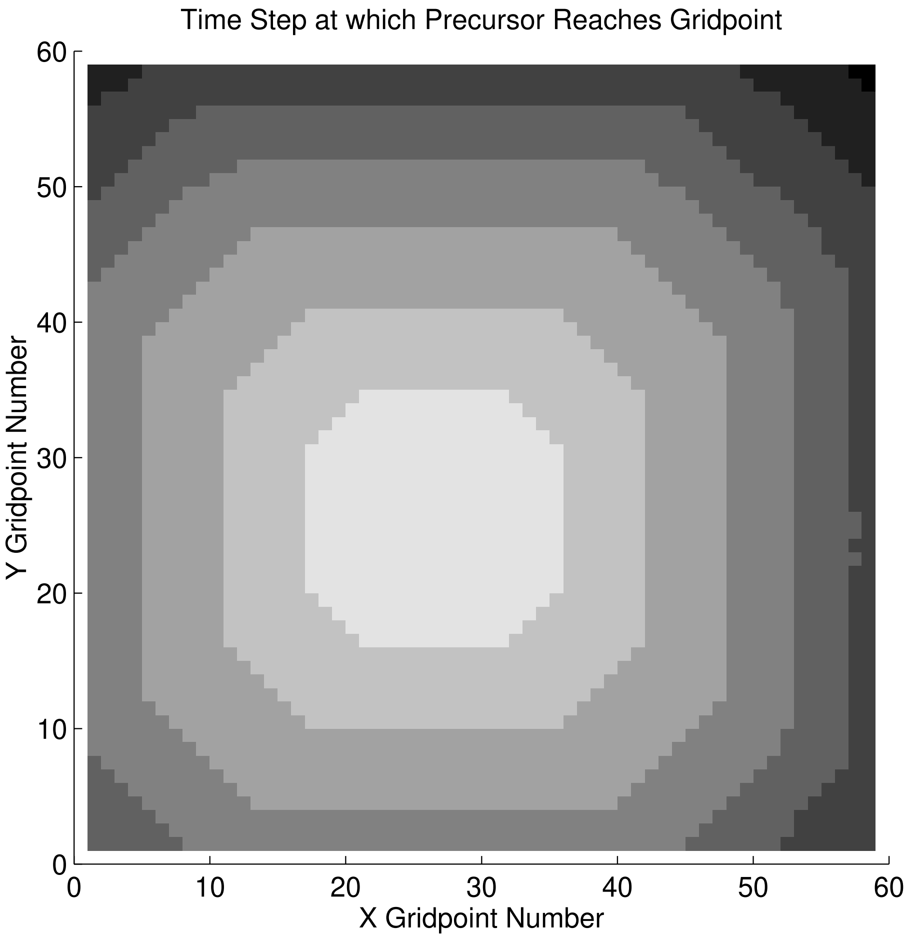

Figure 4: First time step for which the temperature-sensitivity becomes nonzero; darker greys correspond to later times.

primary wave and appeared to be strongly damped (it may perhaps behave be more accurately characterized as a "diffusion," rather than a wave). The late-time precursor may be seen in Figure 2 as the low-amplitude halo surrounding the central, physical disturbance; at the time of this image, the sensitivities would have been expected to vanish outside the roughly 15-gridpoint radius to which sound has had time to propagate.

The apparent speed of the precursor seems impossible, until one realizes that in nonhydrostatic mode, MM5 performs between two and four microsteps using the semi-implicit acoustic-wave solver `SOUND` per each "leapfrog" macrostep, and that `SOUND` couples adjacent grid points together [13, 10]. A plot showing the first time step for which a nonzero sensitivity value occurs at each grid point (Figure 4) indicates that nonzero values appear on the grid in roughly octagonal regions having a half-width and half-height time-sequence of (12, 18, 24, 29, 33, 37, 41); after time step seven, the leading edge of the precursor reaches the upper right corner of the grid, and we can no longer track it. Taking four grid points as the radius of the nonzero initial data, we get propagation increments of (8, 6, 6, 5, 4, 4, 4), suggesting that the asymptotic velocity of the precursor is four grid points per time step.

These observations suggest the hypothesis that, in attempting to account for high-frequency acoustic modes, `SOUND` introduces a spurious "superacoustic" computational mode; indeed, one expects that because of the adjacent coupling introduced by `SOUND`, the superacoustic mode should propagate at four grid points per macrostep, independent of the time step size (the larger initial apparent velocity can be understood as a geometric projection effect due to the conical shape of perturbation (13) caused by the σ-dependence of R). These features suggest that the precursor represents something more nearly akin to a "computational diffusion" than to a true wave.

One should expect that both physical and numerically induced precursors may appear in the results of *any* simulation code. Precursors are an inevitable aspect of dispersive wave propagation [8]; the imposition of a discrete space/time mesh for simulation purposes necessarily introduces additional spurious dispersion into the problem, since the discretization length and time scales impose a cutoff wavenumber and frequency above which propagation becomes impossible [7, 20]. In addition, since a grid-point is coupled to its neighbors via the "stencil" pattern of the differencing scheme, an unphysical "computational wave" will propagate across the grid at one stencil radius per time step. The first (or "Sommerfeld") precursor is physical; it characteristically has a rather high frequency on the order of the reciprocal of some characteristic time associated with the physical properties of the medium, and propagates at velocities on the order of the fastest speed the medium will support—in this case, the acoustic wave speed, which is far too slow to explain the precursor we observe. However, the semi-implicit method used in `SOUND` in effect couples the nearest *four* neighbors together, which is precisely what one needs to explain the observed precursor. We conclude that the precursor is probably a computational artifact induced by the semi-implicit microstep algorithm in `SOUND`.

The observation of this precursor provides a remarkable demonstration of the synergy of both automatic differentiation and suitable visualization techniques: Because of its small amplitude, the precursor almost certainly could not have been observed with a divided-difference approach; indeed, one can easily show that except for its innermost radii during the first few time steps, the precursor would have been completely lost to significant-digit cancellation even on a 64-bit machine. By contrast, AD-computed derivatives retain the full accuracy of the original function. Likewise, because of the very large dynamic range in these images, if we had used equally spaced contour intervals instead of histogrammic equalization, the precursor would almost certainly never have been noticed.

5 Conclusions

We have shown that, using automatic differentiation (AD) tools such as ADIFOR, we can quickly and easily generate and update sensitivity-enhanced versions of large, complicated codes, such as the tangent linear model (TLM) of MM5, a community mesoscale weather, with little expenditure of human labor. We explained the connection between the TLM and formal perturbation theory and showed how automatic differentiation can be used to assess the effect of adding additional observations in the initial temperature field of the MM5 mesoscale weather model with little computational effort. The inherent accuracy of AD-generated codes allows us to observe low-amplitude sensitivity waves such as, for example, a numerically induced supersonic precursor. These results show that automatic differentiation tools, suitably employed, enable the rapid and accurate sensitivity analysis of large computer codes, revealing computational structure that is obscured by the inherent errors in divided-difference approximations of derivatives, and previously detectable only after expending considerable human effort in developing accurate derivative codes by hand.

Acknowledgments

We thank Tom Canfield, Jimy Dudhia, Georg Grell, Kevin Manning, Lt. Col. Harold Massey, John Michalakes, Ravi Nanjundiah, Seon-ki Park, Kathy-Lee Simunich, and David Stauffer for helpful discussions regarding the MM5 code and meteorology, and Peter Campbell and Gerald Marsh for their support of this work.

This work was supported 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.

References

[1] Stanley G. Benjamin and Nelson L. Seaman. A simple scheme for objective analysis in curved flow. *Monthly Weather Review*, 113:1184–1198, 1985.

[2] Christian Bischof, Alan Carle, George Corliss, Andreas Griewank, and Paul Hovland. ADIFOR: Generating derivative codes from Fortran programs. *Scientific Programming*, 1(1):11–29, 1992.

[3] Christian Bischof, Alan Carle, Peyvand Khademi, and Andrew Mauer. The ADIFOR 2.0 system for the automatic differentiation of Fortran 77 programs, 1994. Preprint MCS-P481-1194, Mathematics and Computer Science Division, Argonne National Laboratory, and CRPC-TR94491, Center for Research on Parallel Computation, Rice University. To appear in *IEEE Computational Science & Engineering*.

[4] Christian Bischof, Alan Carle, Peyvand Khademi, Andrew Mauer, and Paul Hovland. ADIFOR 2.0 user's guide. Technical Memorandum ANL/MCS-TM-192, Mathematics and Computer Science Division, Argonne National Laboratory, 1994. CRPC Technical Report CRPC-95516-S.

[5] Christian Bischof and Fred Dilley. A compilation of automatic differentiation tools. *ACM SIGNUM Newsletter*, 30(3):2–20, 1995.

[6] Christian H. Bischof. Automatic differentiation, tangent linear models and pseudo-adjoints. In Francois-Xavier Le Dimet, editor, *High-Performance Computing in the Geosciences*, volume 462 of *Series C: Mathematical and Physical Sciences*, pages 59–80, Boston, Mass., 1995. Kluwer Academic Publishers.

[7] Leon Brillouin. *Wave propagation in periodic structures; electric filters and crystal lattices.* 2nd edition, Dover Publications, New York, 1953.

[8] Leon Brillouin. *Wave propagation and group velocity*, volume 8 of *Pure and applied physics.* Academic Press, New York, 1960.

[9] G. P. Cressman. An operational objective analysis scheme. *Monthly Weather Review*, 87:293–297, 1959.

[10] Jimy Dudhia. A nonhydrostatic version of the Penn State-NCAR mesoscale model: Validation test and simulation of an Atlantic cyclone and cold front. *Monthly Weather Review*, 121(5):1493–1513, 1993.

[11] Ronald M. Errico, Tomislava Vukicevic, and Kevin Raeder. Examination of the accuracy of a tangent linear model. *Tellus*, 45A(5):462–477, 1993.

[12] Phillip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, London, 1981.

[13] G. A. Grell, J. Dudhia, and D. R. Stauffer. A description of the fifth-generation Penn State/NCAR mesoscale weather model (MM5). Technical Report NCAR/TN-398+STR, National Center for Atmospheric Research, Boulder, Colorado, June 1994. See also `http://www.mmm.ucar.edu/mm5/mm5-home.html`.

[14] Andreas Griewank. On automatic differentiation. In *Mathematical Programming: Recent Developments and Applications*, pages 83–108, Amsterdam, 1989. Kluwer Academic Publishers.

[15] Kevin W. Manning and Philip L. Haagenson. Data ingest and objective analysis for the PSU/NCAR modeling system: Programs DATAGRID and RAWINS. Technical Report NCAR/TN-376+IA, National Center for Atmospheric Research, Boulder, Colorado, October 1992.

[16] Stephen M. Omohundro. *Geometrical Perturbation Theory in Physics*. World Scientific Publishing, Singapore, 1986.

[17] William H. Press and Saul A. Teukolsky. Numerical calculation of derivatives. *Computers in Physics*, 5(1):88–89, Jan/Feb 1991.

[18] Louis B. Rall. *Automatic Differentiation: Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, 1981.

[19] W. C. Rheinboldt. *Numerical Analysis of Parametrized Nonlinear Equations*. John Wiley and Sons, New York, 1986.

[20] Robert Vichnevetsky and John B. Bowles. *Fourier analysis of numerical approximations of hyperbolic equations*, volume 5 of *SIAM Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, 1982.

[21] Tomislava Vukicevic and Ronald M. Errico. Linearization and adjoint of parameterized moist diabatic processes. *Tellus*, 45A(5):493–510, 1993.

[22] L. T. Watson. Numerical linear algebra aspects of globally convergent homotopy methods. *SIAM Reviews*, 28:529–545, 1986.