

Development of Adjoint for a Complex Atmospheric Model, the ARPS, using TAF

Ying Xiao¹, Ming Xue^{1,2} and Jidong Gao¹

Center for Analysis and Prediction of Storms¹ and School of Meteorology²
University of Oklahoma, Norman, 73019, USA

1 Introduction

Adjoint models have been used in meteorology for applications such as four-dimensional variational analysis (4DVAR) and sensitivity studies for over two decades although operational implementations of 4DVAR data assimilation did not start until the recent years. Compared with the models used in other fields, operational weather prediction models are much more complex and the problem sizes tend to be much larger too. Thus the application of the associated adjoint models is often hindered by overwhelming programming tasks and high computational cost. The European Center for Medium Range Weather Forecast (ECMWF) is the first center to implement an operational 4DVAR system (Rabier et al., 2000) and the development of the adjoint code took many man-years of investment. The adjoint codes for mesoscale research models have also been developed in recent years, mostly by hand, and used for data assimilation and sensitivity studies (e.g., Errico and Vukicevic 1992; Zou et al 1997; Wang et al 1995; Gao et al 1998; Ruggiero et al., 2002). Most existing adjoint models in meteorology only contain limited and often simplified physics processes, however. For the Advanced Regional Prediction System (ARPS; Xue et al. 2000, 2001), a comprehensive regional atmospheric prediction model, an adjoint was developed by hand, with limited physics, in the mid to late 1990s (Wang et al 1995, Gao et al 1998). Since then, the ARPS model has undergone significant changes. In this paper, we report on our recent work on developing an adjoint code for the full physics ARPS with the help of automatic adjoint generation tools, initially TAMC and more recently TAF.

In the rest of the paper, we denote the ARPS nonlinear model as ARPS NLM and the tangent linear and adjoint models as ARPS TLM and ARPS ADJ, respectively.

2 Transformation of Algorithms in FORTRAN (TAF)

TAF is a source-to-source automatic differentiation (AD) tool for FORTRAN 90/95 codes from Fas-

tOpt (<http://www.FastOpt.com>). For a given nonlinear model, it generates tangent linear (TLM) and adjoint model (ADJ). It is the commercial successor to the Tangent Linear and Adjoint Model Compiler (TAMC) by Giering and Kaminski (1998), which was used to develop the MM5-based 4DVAR system (Ruggiero et al., 2002). Compared with TAMC, TAF is much more robust, much faster and better maintained. At the early stage of our development, we used TAMC but it was not able to generate the tangent linear or adjoint model from the top driver level of ARPS. The model had to be broken up into sublayers and the dependencies across the layers analyzed manually. This greatly degrades the benefit of using AD tools. Similar problems also existed when we first applied TAF in mid-2002, and by working with the TAF developers, we are now able to generate the TLM and ADJ of the entire ARPS model with TAF, though all directly generated codes are not necessarily correct.

Since the AD tools are not completely reliable resulting from inability to correctly handle certain styles (though perfect legal) of code and the presence of bugs in the tools, we had to make changes to the nonlinear model and to report wrong behaviors of TAF to the developer, which often results in bug fixes. Some of the changes will be incorporated into the official version of ARPS NLM so that TAF can be applied to future versions of the NLM with much reduced efforts.

In order to ease the maintenance of the adjoint codes, necessary changes are made to the NLM code instead of the generated ADJ code whenever possible. Direct modification to the ARPS TLM and ADJ codes is discouraged and is used as the last resort to address mainly the performance issues.

3. Code Generation and Testing

3.1. Code Generation Issues

The ARPS NLM was first written with Fortran-77 and was converted to Fortran-90 with the help of an automatic conversion tool. Still, some older features such as SAVE, GO TO statements remain. Even though TAF can handle some of these structures, codes generated with these structures often have problems. For the SAVE statements, we replace them with com-

mon block variables since the SAVE statement causes incorrect recomputation of intermediate NLM quantities and complicates the testing of the generated codes.

Some large NLM subroutines are divided into smaller subroutines. Usually, when the subroutine contains nonlinear calculations, the size of the corresponding adjoint subroutine is increased, sometimes significantly. Too long subroutines often cause compilation to fail at high optimization levels. The main time integration loop of the NLM is also divided into two levels to incorporate the two-level checkpointing schemes supported by TAF, which is designed to reduce the number of time levels that have to be stored in the main memory while at the same time avoid excessive disk I/O. Basically, the NLM state is saved every certain number of time steps, in checkpoint files, and the model states inbetween the checkpoint points are reconstructed by an additional NLM integration from the checkpoint times. Compared to saving the entire NLM trajectory in memory, the cost of doing so is one extra NLM time integration.

Float pointer exception problems have been encountered with the NLM and ADJ codes even though the NLM model works properly. This is because the valid domains of some functions are different from their derivatives. For example, the derivative of $x^{1/2}$ is $\frac{1}{2}x^{-1/2}$ and there will be a float pointer exception if the derivate is executed with $x=0$. Since *sqrt* function is widely used in ARPS NLM, we replace the *sqrt* function by *safesqrt* as follows,

```
safesqrt(x):
  if ( $x < a\text{-small-value}$ ) then return sqrt(a-small-value)
  else return sqrt(x)
```

3.2. Code Testing

Let $F(X)$, $g_F(X, g_X)$ and $adF(X, adX)$ denote the NLM, TLM and ADJ models, respectively, where g_X and adX are the arguments of TLM and ADJ functions (subroutines). The correctness of the ARPS TLM and NLM model are tested with three standard procedures:

a) Linearity Test of the TLM:

$$g_F(X, c \cdot g_X) = c \cdot g_F(X, g_X), c \in R^{++} \quad (1)$$

b) Comparison with finite difference of NLM

$$\frac{[F(X + \delta X) - F(X)]}{g_F(X, \delta X)} = 1 \quad \lim \delta X \rightarrow 0 \quad (2)$$

c) Consistence between TLM and ADJ model:

Let $g_Y = g_F(X, g_X)$, $adY = adF(X, g_Y)$. Then check whether

$$\langle g_Y, g_Y \rangle = \langle adY, g_Y \rangle. \quad (3)$$

If the toplevel subroutine of the generated code fails the test, testing will be performed from the top to lower level subroutines until the source of the error is identified. The testing may also be applied to a block of codes to detect the error if the problem is found to lie between two consecutive subroutine levels.

Because almost all of the problems encountered by TAF resulted from recomputation of the NLM quantities, we add to the above three standard tests a checkpoint test for verifying the correctness of recomputation for some subroutines. In the test, we first invoke TLM and record all NLM quantities passed to the subroutine being tested and run the ADJ model to compare the NLM values passed to the corresponding adjoint subroutine. During the development of ARPS ADJ, we carried out this testing only for adjoint subroutines that fail to pass the three standard tests.

Since the test procedure is standard and it is tedious and error prone to write the test code and driver by hand, we developed an automatic test code generator, called TATCG (Tangent and Adjoin Test Code Generator), to facilitate the testing. TATCG can generate test code and driver for all four tests described above. It may also perform code transformation if the test is to be applied to a code block instead of a subroutine. New subroutines that wrap the code blocks will be generated in addition to the test codes.

4. Known Problems of TAF

The overall performance of TAF is very impressive and the use of it greatly sped up the development of ARPS ADJ model. The robustness of TAF also improved with versions during the period we used it. However, there remain some problems with the generated codes. These problems either resulted from TAF bugs or the undecidability of static algorithm analysis used by TAF. In this section, we will show some TAF errors encountered during the development of ARPS ADJ model. Some of these problems may have been solved by the time that you read this paper, however.

As with its predecessor TAMC, most of the TAF problems are related to the recomputation of the intermediate NLM values and flow dependency analysis of arrays.

4.1. Incorrect Handling of the SAVE Statement

If a variable is declared with SAVE attribute, its value will be sustained until the termination of the program, which may introduce intricate data flow depend-

encies. The current version of TAF cannot handle this properly. The following is an example.

```

SUBROUTINE test(x, y)
do i = 1, 10
    CALL demo(x, y)
enddo
END SUBROUTINE test

SUBROUTINE demo(x,y)
LOGICAL :: firstcall
SAVE firstcall
data firstcall/.true./
IF (firstcall) THEN
    y = x**2
    firstcall = .false.
ELSE
    y = x**3
END IF
END SUBROUTINE demo

```

The TAF generated code is as follows (only the toplevel subroutine).

```

subroutine adtest( x, adx, y, ady )
!...omited....
!-----
! FUNCTION AND TAPE COMPUTATIONS
!-----
do i = 1, 10
    call demo( x,y )
end do
!-----
! ADJOINT COMPUTATIONS
!-----
do i = 10, 1, -1
    call addemo( x,adx,ady )
end do
end subroutine adtest

```

It is easy to see that the last invocation of the adjoint subroutine *addemo* in the loop is not correct because noting is done to recover the values *firstcall*.

As we mentioned earlier, we solved this problem by replacing all SAVE statements with common statements. This rule will also be applied to new codes of ARPS NLM.

4.2. Array Dependency

TAF cannot distinguish the data dependency of individual elements and those of entire arrays. Both TAF and TAMC have this limitation and a general solution is harder to find in this case.

```

SUBROUTINE test (f,x)
REAL::f(100),x(100)
x(1) = x(1)
x(1)=1.0
DO I=1,2
    f (i)=x(i)**2
ENDDO
END SUBROUTINE test

```

If the bold line in the above code is commented out, TAF will assume no data dependency between the value *f* and *x* passed to the subroutine as if the array *x* are set as constant. This problem can be solved by add the bold line in the above code segment.

5. ARPS ADJ Model and Performance

Not counting the I/O libraries, user interface and other helper functions, the ARPS NLM source code is about 40,000 lines in FORTRAN 90 excluding comments and blank lines. It contains a set of physics parameterizations, i.e., subgrid-scale and planetary boundary layer turbulence parameterization, cloud microphysics (CM), cumulus parameterization (CP), surface layer parameterization (SLP), soil model (SM), longwave and shortwave radiation (LSR). Except for the CP and SLR, and some seldomly used subcomponents, the ARPS ADJ model includes all of the major components of the APRS NLM. According to our knowledge, this is one of the most complicate adjoint codes in meteorology. The adjoint codes of CU and LSR have also been generated but have not been debugged.

The correctness and efficiency of the ARPS ADJ model are tested with the data from a supercell thunderstorm simulation, which included externally boundary conditions and full physics. We also carry out some simple adjoint sensitivity experiments for which we have a good idea on the correct solution. The tests were carried out on IBM Regatta P690 computer with Power4 1.1 GHz CPU. All the experiments are performed at double precision although single precision generally works too.

5.1 Correctness test

In this testing, we implemented the three standard schemes in section 3. If the test output is closed to 1, the adjoint codes are considered correct.

Tested components:

CM (Kessler warm rain microphysics), SLP and SM
 Integration time: 6600 seconds to 7620 seconds
 Integration time steps: 20

TLM Model Computation time: 2 times NLM model
 ADJ Model Computation time: 11 times NLM model

Output for test a): 1.0000000000000000
 Output for test b): 1.00000196449945489
 Output for test c): 0.9999999999999994116

5.2 Sensitivity analysis

Numerically modeled simulations are used in the investigation of complex physical systems. These mod-

els typically contain many parameters and/or initial condition values. The results may or may not be very sensitive to the variations in these values. Adjoint model has been widely used as a powerful tool for the sensitivity analysis (Ronald 1997). We carry out an experiment to evaluate the performance of ARPS ADJ model in a real application. Unlike the first experiment which contains complex physics processes, we turn off the CM, SLP, and SM in the adjoint model. Only very simple equations that include primarily advection and diffusion processes are activated such that the correct sensitivity results can be estimated analytically.

In the experiment, a small perturbation is applied to the water vapor mixing ratio q_v located at $x = y = 100$ km (the target location) at $t=1800$ s (target time) from which the adjoint model starts integration backwards. The sensitivity of q_v at the above point to q_v at $t=64$ s (sensitivity time) is shown in Fig. 1. Since the low-level winds at this level is from southwest to northeast, a circular sensitivity pattern located upstream of the target location is expected and further early is the time of sensitivity field, the larger should be this pattern, and the smaller is the magnitude because of the presence of diffusion. Physically, a change at grid point at the sensitivity time of the amount given by the sensitivity field (the output of adjoint model) should cause a unit change in q_v at the target location. In general, the trajectory and magnitude of the values given by the ADJ model match the physics process very well. In this simplified setting, the perturbation of the physics parameter propagates southeast to northwest at 10 ms^{-1} . The sensitivities of the model output to other parameters have also been examined, and all results examined so far appear correct.

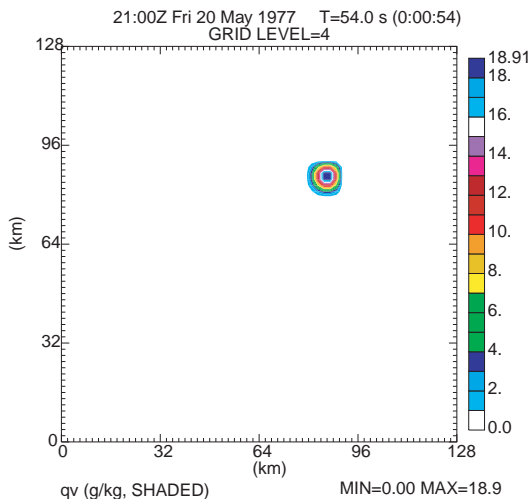


Fig.1. Sensitivity of water vapor q_v at location $x=y=100$ km at $t=1800$ s to q_v field at time $t=64$ s.

6. Future Plan

We plan to continue to develop the adjoint model for the CP and LSR components. The codes for these two components are complex and contain many more problematic constructs than in other parts the ARPS model. Substantial code changes may be need for TAF work effectively.

The current ARPS ADJ model is a serial model and is far from being inefficient. This issue can be addressed by optimizing the ADJ codes and through parallelization. The former can be accomplished by carefully making compromise between recomputation and storage of the intermediate NLM quantities, removing redundant storage and recomputation and through direct modification of the generated codes.

We will apply the adjoint code to sensitivity studies of real cases, and develop an 4DVAR system based on the adjoint.

Acknowledge: This work was supported by NSF ATM-0129892 and an grant from FAA.

References

- FastOpt, Manual of Transformation of Algorithm in FORTRAN.
- Giering, R. and T.Kaminski, Recipes for adjoint code construction, ACM Trans. Math. Software 24 (4), 1998
- 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.*, 126(A), 1143-1170
- Ronald, M. Errico, What is and Adjoint Model?, Bulletin of the American Meteorological Society, 1997
- Ruggiero, F., D. Norquist, T. Nehrkorn, G.D. Modica, M. Cerniglia, J.G. Michalakes, and Z. Zou, 2002: MM5V3-Based 4DVAR: current status and future plans, 12th PSU/NCAR mesoscale model users' workshop, NCAR, June 24 - 25, 2002
- Xue, M., K. K. Droegemeier, and V. Wong, 2000: The Advanced Regional Prediction System (ARPS) - A multiscale nonhydrostatic atmospheric simulation and prediction tool. Part I: Model dynamics and verification. *Meteor. Atmos. Physics*, **75**, 161-193.
- Xue, M., K. K. Droegemeier, V. Wong, A. Shapiro, K. Brewster, F. Carr, D. Weber, Y. Liu, and D. Wang, 2001: The Advanced Regional Prediction System (ARPS) - A multi-scale nonhydrostatic atmospheric simulation and prediction tool. Part II: Model physics and applications. *Meteor. Atmos. Phys.*, **76**, 143-166.

- Zou, X., F. Vandenberghe, M. Pondaca, and Y.-H. Kuo, 1997: Introduction to adjoint techniques and the MM5 adjoint modeling system. NCAR Technical Note NCAR/TN-435-STR, 110 pp.
- Wang, Z., K. K. Droegemeier, M. Xue, and S. K. Park, 1995: Sensitivity analysis of a 3-D compressible storm-scale model to input parameters. *Proc., Int. Symp. on Assim. Obs. Meteor. Oceanography.*, World Meteor. Org., Tokyo, Japan, 437-442.
- Gao, J., M. Xue, Z. Wang, and K. K. Droegemeier, 1998: The initial condition and explicit prediction of convection using ARPS adjoint and other retrievals methods with WSR-88D data. *12th Conf. Num. Wea. Pred.*, Amer. Meteor. Soc., Phoenix AZ, 176-178.
- Errico, R. M. and R. Vukicevic, 1992: Sensitivity analysis using an adjoint of the PSU-NCAR mesoscale model. *Mon. Wea. Rev.*, **120**, 1644-1660.