

Generating tangent linear and adjoint versions of NASA/GMAO's Fortran-90 global weather forecast model

Ralf Giering¹, Thomas Kaminski², Ricardo Todling³, Ronald Errico⁴, Ronald Gelaro⁵, Nathan Winslow⁶

ABSTRACT The NASA finite volume General Circulation Model (fvGCM) is a three-dimensional Navier-Stokes solver that is being used for quasi-operational weather forecasting at NASA-GSFC. By means of TAF, efficient tangent linear and adjoint versions are generated from the Fortran-90 source code of fvGCM's dynamical core. fvGCM's parallelisation capabilities based on OpenMP and MPI have been transferred to the tangent linear and adjoint codes. For OpenMP, TAF automatically inserts corresponding OpenMP directives in the derivative code. For MPI, TAF generates interfaces to hand written tangent linear and adjoint wrapper routines. TAF also generates a scheme that allows the tangent linear and adjoint models to linearise around an external trajectory of the model state. The generation procedure is set up in an automated way, allowing quick updates of the derivative codes after modifications of the fvGCM.

Keywords: automatic differentiation, tangent linear model, adjoint model, source-to-source translation, Fortran-90

The NASA finite volume General Circulation Model (fvGCM) [7, 6, 8] is a three-dimensional Navier-Stokes solver. The GCM has been developed at NASA's Data Assimilation Office (DAO, now Global Modeling and Assimilation Office, GMAO) for quasi-operational weather forecasting. The model has various configurations. For the current study, we use the b55 production configuration, which runs on a regular 144×91 horizontal grid and 55 vertical layers. The time step is 30 minutes, and typical integration

¹FastOpt, Hamburg, Germany, (Ralf.Giering@FastOpt.com)

²FastOpt, Hamburg, Germany, (Thomas.Kaminski@FastOpt.com)

³NASA/GSFC, Greenbelt, MD, USA, (todling@gmao.gsfc.nasa.gov)

⁴NASA/GSFC, Greenbelt, MD, USA, (rerrico@gmao.gsfc.nasa.gov)

⁵NASA/GSFC, Greenbelt, MD, USA, (rgelaro@gmao.gsfc.nasa.gov)

⁶NASA/GSFC, Greenbelt, MD, USA, (winslow@gmao.gsfc.nasa.gov)

periods vary between 6 and 48 hours depending on applications. The state of the model comprises three-dimensional fields of 5 prognostic variables, namely two horizontal wind components, pressure difference, potential temperature, and moisture.

To run in the GMAO's retrospective Data Assimilation System (GEOS, [10]) tangent linear (TLM) and adjoint (ADM) versions of fvGCM's dynamical core are needed. Further applications such as sensitivity analysis, stability (singular vector) analysis, or chemical data assimilation also require a TLM and ADM. TAF [2, 4] is a source-to-source transformation AD-tool for programs written in Fortran 77-95, i.e. TAF generates a TLM or ADM from the source code of a given model. As the above applications differ in their sets of dependent and independent variables, TAF generates a TLM/ADM pair for each of the above applications and in addition two pairs for finite difference tests (rough and detailed).

fvGCM is implemented in Fortran 90 and contains about 87000 lines of code excluding comments. It makes use of Fortran-90 features such as *free source form*, *modules*, *derived types* and *allocatable arrays*. Our standard approach to render a given code TAF compliant consists in combining modifications to the model code with TAF enhancements. For instance, at two places, fvGCM's control flow structure has been simplified. Generation of an efficient store/read scheme for providing required variables [2] has been triggered by 41 TAF init directives and 75 TAF store directives. The ADM can be generated with and without a checkpointing scheme [3, 4]. To support TAF's data dependence analysis, 11 TAF loop directives have been used to mark parallel loops. In total 204 TAF flow directives have been inserted to trigger generation of specified calling sequences [3]. For instance, TAF flow directives allow to use the Fast Fourier Transformation (FFT) and its inverse in the TLM and ADM respectively, which is more efficient than using a generated derivative code [9, 4]. In some subroutines, variables were allocated and/or initialised during the very first call. This introduces a data flow dependence between the first and later calls which forces TAF to generate proper but inefficient recomputations. In order to avoid these recomputations we have moved the allocations and initialisations into extra module procedures which are not differentiated. As a result of these initial modifications, the generation procedure for the derivative code is now fully automated. This is important to allow quick updates of the derivative code to modifications in the underlying model code. Unfortunately there is a bug in the SGI-compiler on the production machine (Origin 2000) which requires switching off the compiler optimisation for 2 files and reduced (-O2 instead of -O3) the optimisation level for 6 more files.

For a typical production machine such as the SGI Origin 2000, the model is parallelised using a combination of OpenMP over vertical levels and MPI (MPI-1 [5] or MPI-2) over latitude bands. TAF has been enhanced to transfer the model's OpenMP compatibility to the generated TLM and ADM. Each of the model's OpenMP directives is transformed to a correspond-

ing directive in the TLM and ADM. Carle and Fagan[1] address handling of MPI-1 in TLM's of Fortran 77 codes. TAF has also been enhanced to handle many MPI calls when generating TLMs. For fvGCM the only one missing is *MPI_Allreduce* with *MPI_MAX* as reduction operation. As MPI versions of both the TLM and ADM are needed, we chose to handle MPI via a different approach. The model's MPI calls are restricted to a set of 25 wrapper routines that arrange communication across the borders of latitude bands. Adjoints of all wrappers have been hand coded. In the TLM, most of the model wrappers can be reused, only 1 TLM wrapper had to be hand coded. As the actual MPI library calls are carried out within the wrappers, this approach is working independently of the MPI implementation (MPI-1 or MPI-2). Generation of the calling sequences for TLM and ADM versions of the wrappers is triggered by TAF flow directives. This allows to fully automate the process of derivative generation, which results in a TLM and ADM for each of the 4 combinations OpenMP on/off and MPI on/off.

In dynamic meteorology it is typical to run the TLM/ADM on a coarser resolution than the forecast model. Also a set of processes called physics (parametrisation of clouds and rain) is usually not included in the derivative code. To compensate for this approximation, one linearises along a trajectory of the state of the full high resolution model. Technically this is achieved by periodically reading in this state from an external trajectory during the (coarse) TLM/ADM integration and over-writing the state of the model the TLM/ADM belongs to. For an AD-tool over-writing makes the final model state appear independent from the initial model state, as the data flow is interrupted. Straight forward use of AD would result in an erroneous TLM and ADM. To solve this problem, we exploit TAF's flexibility in setting up a store/read scheme for providing required variables. A combination of TAF init, store, and flow directives triggers generation of the desired TLM and ADM. It is important to keep in mind that the resulting TLM and ADM are only proper linearisations of the processed model, if this very model has been used to compute the external trajectory.

The performance of the sequential TLM and ADM versions has been tested on a Linux PC (P4 3GHz Processor, 2 GByte memory, Intel Fortran Compiler 8.0) and on a SGI Origin 2000. It is common to quantify the CPU time of the derivative code in multiples of the CPU time of a function evaluation (model integration). Note that both the TLM and ADM do not only evaluate the derivative but also the function.

On Linux-PC because of memory limitations we could only run a smaller configuration (a18, 72x46x18 grid). The run time is 1.1s for the function, the relative run time of the TLM (ADM) is 1.5 (7.0).

On SGI we run the two different parallelisation strategies. The function run time without parallelisation is about 130s. Using OpenMP 1.1 the speedup increases with the number of processors (Fig. 1). While the TLM speedup compares well with that of the function, the ADM speedup lags

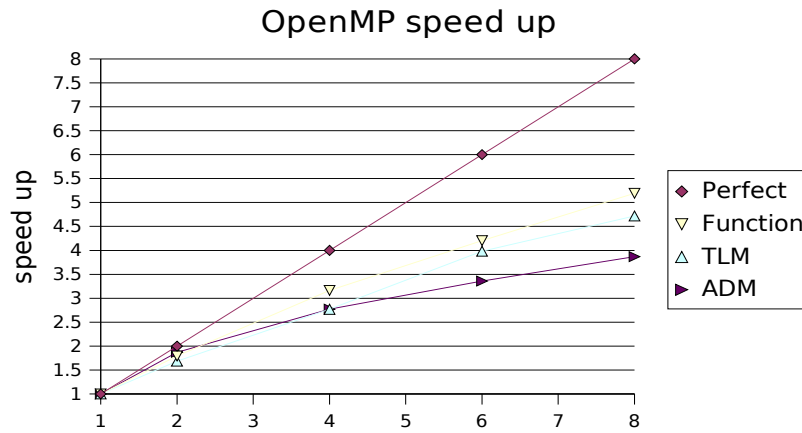


FIGURE 1. Speedup of OpenMP with number of threads

behind. We think that this is due to many critical sections which have to be generated because OpenMP 1.1 does not support array reductions. We expect that OpenMP 2.0 code will have a better performance. We are currently examining that. In the MPI case (Fig. 1) the speedup of the function code is much better and the TLM and ADM archive the same speedup.

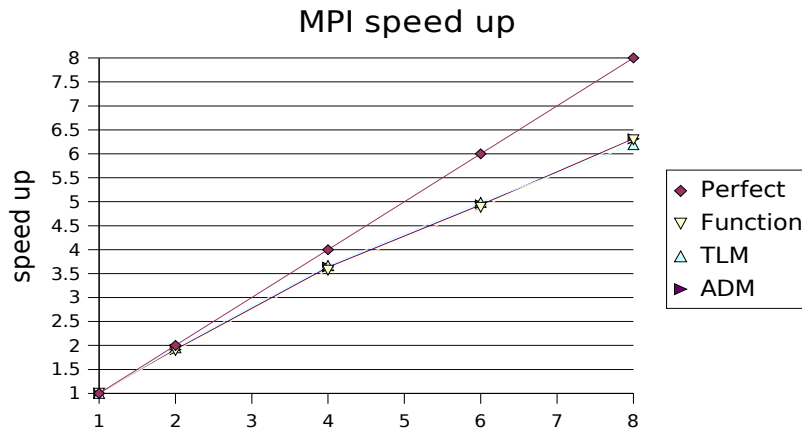


FIGURE 2. Speedup of MPI with number of processors

In summary, the relative performance of the TLM on the SGI is between 1.3 and 1.5, depending on the number of threads/processors and the parallelisation strategy. Compiling with reduced optimisation (see above) the

relative performance of the ADM is between 12 (1 CPU) and 21 (8 CPUs OpenMP). Compiling with full optimisation this factor decreases to a value between 6 and 11, but here the derivatives are inaccurate by a few percent.

Currently the derivative generation procedure is being adapted to and integrated in fvGCM's file hierarchy and setup procedure. The fully automated generation approach allows adaptations of the derivative code to modifications in fvGCM. Currently, extensions to the dynamical core such as a simplified physics model (an important damping process) and the corresponding TLM and ADM are being developed/generated.

1 REFERENCES

- [1] Alan Carle and Mike Fagan. Automatically differentiating MPI-1 datatypes: The complete story. In George Corliss, Christèle Faure, Andreas Griewank, Laurent Hascoët, and Uwe Naumann, editors, *Automatic Differentiation: From Simulation to Optimization*, Computer and Information Science, chapter 25, pages 215–222. Springer, New York, 2001.
- [2] R. Giering and T. Kaminski. Recipes for Adjoint Code Construction. *ACM Trans. Math. Software*, 24(4):437–474, 1998.
- [3] R. Giering and T. Kaminski. On the performance of derivative code generated by TAMC. *submitted to Optimization Methods and Software*, 2002.
- [4] R. Giering, T. Kaminski, and T. Slawig. Applying TAF to a Navier-Stokes solver that simulates an Euler flow around an airfoil. *to appear in Future Generation Computer Systems*, 2004.
- [5] William D. Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI – Portable Parallel Programming with the Message Passing Interface*. MIT Press, Cambridge, 1994.
- [6] S.-J. Lin. A finite volume integration method for computing pressure gradient force in general vertical coordinates. *Quart. J. Roy. Meteor. Soc.*, 123:1749–1762, 1997.
- [7] S.-J. Lin and R. B. Rood. Multidimensional flux-form semi-lagrangian transport scheme. *Mon. Wea. Rev.*, 124(9):2046–2070, 1996.
- [8] S.-J. Lin and R. B. Rood. An explicit flux-form semi-lagrangian shallow-water model on the sphere. *Quart. J. Roy. Meteor. Soc.*, 123:2477–2498, 1997.
- [9] Oliver Talagrand. The use of adjoint equations in numerical modelling of the atmospheric circulation. In Andreas Griewank and George F.

Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 169–180. SIAM, Philadelphia, Penn., 1991.

- [10] Y.Q. Zhu, R. Todling, J. Guo, S.E. Cohn, I.M. Navon, and Y. Yang. The geos-3 retrospective data assimilation system: The 6-hour lag case. *Monthly Weather Review*, 131(9):2129–2150, 2003.