Dear reviewers

First of all, thank you very much for reviewing our manuscript. Per your comments (in black font), we have revised our manuscript accordingly (in red font) and made point-to-point responses (in blue font) to all the comments and concerns. Below are our detailed responses.

Reviewer#1

1 General comments:

- Overall, from a computational perspective, I see little evidence or data supporting your conclusions. I would suggest to add more information about your analysis and verification methods. (Please see my Specific Comments).

Reply: We very appreciate the helpful suggestions, although most of the manuscript are discussions of code optimization techniques.

- You mention that your code is already part of WRFPLUS (dyn_em). Can you please add any link or URL for a pull request or discussion of the code? This is valuable information for future work and other scientists.

Reply: Sorry to fail to download the complete codes. We move the complete code to a new site (https://github.com/juanjliu/WRFDA-OPTIMAZATION with a simple readme as following:

Step 1: Download the underlying versionV3.7

Users can download the WRFPLUS source code from http://www2.mmm.ucar.edu/wrf/users/wrfda/download/wrfplus.html , then Unzip and untar the WRFPLUS file.

Step 2: Download the optimization code

User can download the optimization code from this web: https://github.com/juanjliu/WRFDA-OPTIMAZATION.git

Step 3: Unzip and untar the WRFDA-OPTIMAZATION file

cd WRFPLUSV3 (from Step 1)

Using optimization codes to Update codes in the underlying version

Step 4: Installing WRFPLUS for 4D-Var Run

cd WRFPLUSV3

./configure wrfplus

./compile em_real >& compile.out

More detail information can be found from “README.md” on https://github.com/juanjliu/WRFDA-OPTIMAZATION.git.

- English: Can you please make a general review, readability could be improved and some phrases and expressions are hard to understand.

Reply: The constructive comments are highly appreciated. In fact, we have done two polishing services. To improve the readability, we further modify some phrases and expressions in the manuscript (marked in red font).

2 Specific Comments:

- P2, l16: "Apparently, different ways have different costs in runtime or memory." Can you please be more specific (examples, including data about impact on memory usage).

Reply: Thanks for your suggestion. However, this is a big problem to answer since there are innumerable ways for adjoint implementations. For this reason, we can present several documents and discussions in this manuscript.

And here, we present some ways as examples:

Since the adjoint accumulation is always performed in a reverse way ($\nabla^{\mathrm{T}}\mathrm{F}_s \rightarrow \nabla^{\mathrm{T}}\mathrm{F}_1$ See Fig 1 in the manuscript), we face the problem: how can the value of a required variable (generally called basic states $\mathrm{T}_1, \mathrm{T}_2, \cdots, \mathrm{T}_s$) for the current adjoint calculation be obtained. Generally, there are some ways to do including directly restoring and directly recalculating.

For directly restoring, one can use Output/Input (OI) Memory or OI Disk (See http://www2.mmm.ucar.edu/wrf/users/wrfda/Docs/user_guide_V3.7/users_guide_chap6.htm#_Description_of_Namelist_2).

OI Memory: one can successively push the required data $\mathrm{T}_1, \mathrm{T}_2, \cdots, \mathrm{T}_s$ into a huge stack L in the last run of the underlying functions and naturally pop them out in the reverse adjoint accumulations one by one. Then OI Memory will need huge costs in memory.

OI Disk: The required data $\mathrm{T}_1, \mathrm{T}_2, \cdots, \mathrm{T}_s$ was written into disk one by one. It has little memory-consuming, but has expensive time-consuming so the WRFDA would not support the OI Disk option.

For directly recalculating, one can recalculating it from the start of the underlying functions, or calculating it from intermediate results. The former has more expensive time-consuming than the latter.

- P2, l21: You make a leap from WRF to WRFPLUS that can make it confusing for readers that are not familiar with the version (package) differences. Can you please add some text in here (1/2 lines) talking about their differences thus, increasing the readability?

Reply: Sorry for confusion here. The Weather Research and Forecasting (WRF) model is a numerical weather prediction and atmospheric simulation system designed for both research and operational applications. The adjoint model and the tangent linear model (called WRFPLUS) based on WRF are developed for the four-dimensional variational data assimilation (4DVar), cloud analysis, forecast sensitivity to observations, and chemistry data assimilation et al. (Zhang et al. 2013).

For clarification, we revised manuscript.

Reference

Zhang, X., Huang, X. Y. and Pan, N.: Development of the Upgraded Tangent Linear and Adjoint of the Weather Research and Forecasting (WRF) Model. J. Atmos. Ocean. Tech., 30(6), 1180-1188, doi: 10.1175/jtech-d-12-00213.1, 2013.

- P3, l5: "This implementation requires comparatively less memory consumption but much more computational cost, although there are always productive ways available for reducing the latter in practice." Can you please be more specific on about 'less memory' and 'more computational cost'? Can you please give an example of other ways of resources usage optimization?

Reply: "Different with the saving-all strategy as TAPENADE, this implementation of a 2-level checkpointing structure requires comparatively less memory consumption but much more computational cost, although there are always productive ways available for reducing the latter as shown in next discussions."

- P3, l31: "As a typical strategy of the reverse accumulations..." Can you please add a reference in here?

Reply: Done. Thanks.

- P6, l25: "allocating/ deallocating them outside of the running cycles of the procedures." Can you please specify the cost of this? What it is the benefit of this for the overall run (intuitively this will improve performance on your part of the code but move the cost somewhere else).

Reply: In Fortran 90/95 programs, the dynamic array allocation within tight loops can really slow down the execution of my code.

i.e., several medium-sized arrays are allocated inside a loop, like:

```
Do i=1,1000
  Allocate(tmp(20))
  tmp(1:20)=1d0
  call foo(tmp)
  deallocate(tmp)
enddo
```

But in WRFPLUS, "it"is really of a very small cost for the overall run "in some cases, we can significantly reduce "it" by allocating/deallocating them outside of the running cycles of them", we will change the statements to make it clear. Thanks.

- P7, l23: "Through careful IO analysis". During all the paper you did not indicate any technique (e.g. tracing, or data dumping) that lead you to all these conclusions more than reading the code and finding IO calls. Can you please elaborate this more and provide more insights/evidence of this analysis and its conclusions?

Reply: IO analysis is a relative notation that is employed to describe the input/output behavior of a variable within a program object defined by a segment of program lines. By recording the IO knowledge of argument parameters in each procedure, it can calculate the final IO relationship of any argument parameter through deep recursive dependence analysis. This process can be represented as an iteration $A_{k+1} = A_k \oplus {A_k}^2$,where A_0 is the initial dependence matrix.

Because IO technique has been discussed in other documents, so we add two references as "Through careful IO analysis and the "To be recorded" analysis techniques for reverse mode (Cheng et al. 2009a, Hascoet et al. 2005)" in the revised manuscript. Thanks.

Reference:

Cheng Q., H.B. Zhang and Y.H. Zhao: Differentiation Transforming System, Nature Sciences Progress, 19(3): 397-406, doi: 10.1016/j.pnsc.2008.07.012 , 2009a.

Hascoet, L., U. Naumann and Pascual, V.: "To be recorded" analysis in reverse-mode automatic differentiation. Future Gener. Comp. Sy., 21(8). 1401-1417, doi: 10.1016/j.future.2004.11.009 ,2005.

- P8, l7: "on a cluster system with 250 nodes, each of which has 20 processors" Can you please be more specific (node type, processor type, memory, network interconnection...)?

Reply: Thanks for your suggestion. We have added it in the revised manuscript.