Reply on CC1

Hui Wan (on behalf of all authors)

Thank you, Sean, for the very insightful comments, questions, and suggestions. As a long-time user of CESM and E3SM, you already provided answers to most of the questions when making your comments. Nevertheless, our responses are provided below.

Comment: I find this manuscript to be well-written overall, and to describe a feature that seems very desirable as a long-time user of CESM and E3SM. There are a few comments/questions I had though:

1. I found this behavior described in the paper to be odd:

"If the metric and the QoI are both 3D but have different numbers of vertical layers (e.g., the metric is the air temperature defined at layer midpoints while the QoI is the net longwave radiative flux defined at layer interfaces), then masking will be skipped, meaning this specific QoI will be captured for output as if no conditional sampling had happened."

My first reaction to this was that the code should raise an error in this case rather than proceed without conditional sampling, since it's surely a user error if a QoI is listed that is incompatible with the metric dimensions. It's only later that the manuscript mentions that the QoIs have to be the same for all conditions, i.e. there may be multiple conditions with different metrics and the list of QoIs cannot be tailored to each condition separately. I assume that this is the reason why the code has to allow a QoI to be specified in this way rather than raising an error, but when reading the paper in order, it's not clear why this behavior was chosen.

Response: You are very right that this oddity resulted from the simpler design we chose for version 1 of CondiDiag, namely there may be multiple conditions with different metrics of different dimensions and the list of QoIs cannot yet be tailored to each condition separately. This explanation is added to the revised manuscript at the place you quoted. Thank you.

In the conclusions section of the original manuscript, we said "the current tool monitors the same set of QoIs and checkpoints under all conditions ... we will assess the trade-off between more flexibility and the potential risk of causing confusion for model developers and users." In the past few months, we have already seen multiple new use cases in our own numerics work that this simpler design can lead to a large number of unnessary variables in the history file. So, we probably will change this in version 2 of CondiDiag.

Comment: 2. In order to better understand how portable the code is, is it true that the conditional_diag module does not use EAM-specific data structures, but that the other three modules do to some extent?

Response: I would say all four modules use some EAM-specific data structures and software functionalities. Some of them would be straightforward to port and some would need a rewrite:

- The `conditional_diag` module has the weakest dependency on EAM/CAM: its meta-data handling part (i.e., parsing the user's choices of QoIs, metrics, etc.) is independent of EAM's data structures. The module also (at least currently) contains a few subroutines that allocate memory for the derived-type arrays used for storing the QoIs, metrics, etc., which assume a column-chunk structure for the horizontal grid.
- The `conditional_diag_main` module assumes all QoIs and condition metrics can be retrieved or re-calculated from EAM/CAM-specific data structures like `state`, `pbuf`, `cam_in` and `cam_out`, so this

module cannot be used in other GCMs without adaptation. On the other hand, I would say the more important part of the module is the general algorithms used for retrieving field values, deriving increments, calculating vertical integrals, and performing conditional sampling, etc. If we understand how another GCM's "`state`" and "`pbuf`" etc. are structured, the adaptation is likely to be relatively straightforward.

- The `conditional_diag_output_utils` module uses EAM/CAM's `addfld` and `add_default` subroutines which define variables/fields in the history output files. The part that handles history variable names to distinguish different sampling conditions, checkpoints, etc., can be simply taken to another GCM.
- The `conditional_diag_restart` module is completely EAM/CAM-specific as the subroutines therein read and write restart files. When porting to another AGCM, this module would need to be rewritten following how restart-related I/O is done in the new host model.

Comment: Also, since the vertical coordinate must be known in order to perform averaging, is the vertical dimension always the last array dimension in the code?

Response: I suppose you mean the vertical integral? In the tiny subroutine `mass_wtd_vert_intg` which calculates the integral, it is assumed that the vertical dimension is the second dimension of the input array (the first dimension is assumed to be column). But I think this can be easily adapted, as Fortran's `sum` function has an optional input argument for specifying which dimension to sum over. For EAM, we currently have

```
! Vertical sum divided by gravit

arrayout(1:ncol,1) = sum( tmp(1:ncol,:), 2 )/gravit
```

If a new host GCM uses, say, rank-3 arrays with dimensions (z, y, x), then we could change the code snippet to the following:

```
! Vertical sum divided by gravit

arrayout(1,1:ny,1:nx) = sum( tmp(:,1:nx,1:ny), 1 )/gravit
```

Comment: 3. For Tables B1 and B2, it may be good to mention that these order of checkpoints in the table is the same as the actual order of the checkpoints in the code (assuming that this is the case).

Response: This is indeed the case, and the comment is added to the table captions in the revised manuscript. Thanks for the suggestion.