The Authors want to take this opportunity to thank the reviewer for their time and their help in improving the manuscript.

Reviewer's 1 Comments

1. *On page 3, line 72 you speculate that this software could be a valuable tool for upcoming CMIP7 experiments. However, in the rest of the text, I get the impression that VISION is designed to work with hourly output from UM/UKESM with specific output file names. Furthermore, to my knowledge the most frequent global fields in CMIP are 3-hourly. Does the tool operate on CMIP6 CMORized output? If not, could you elaborate in the discussion what would be needed for VISION to claim a role in the CMIP7 assessment cycle?*

The VISION tools are currently capable of reading CMIP6 CMORized output using cf-python libraries. Whilst the UM/UKESM has been used as a test model and the current version of the tool is designed to work with UM output, the next version of the VISION toolkit (currently under development) will be model independent and is currently being tested with output from other models.

As part of the NERC-TWINE VISION project (which is partly funding the further development of the VISION toolkit), we aim to run training sessions and produce training material to support modellers who wish to integrate VISION into their model's workflow. If the CMIP community recognises the value of this tool and is willing to engage in the required training, some CMIP7 models could include VISION into their workflow and produce temporary hourly output which could be converted into CMORized co-located data for archiving.

In order to clarify this point, the text on page 3, line 72 will be changed from:

"The code is being developed to read any CF compliant model data and so could provide a valuable tool for supporting expanded diagnostics in upcoming CMIP7 experiments."

to:

"Whilst the UM/UKESM has been used as a test model and the current version of the tool is designed to work with UM output, the next version of the VISION toolkit (currently under development) will be model independent. Since VISION is designed to work with CF compliant data, including CMIP CMORized output, it could prove a valuable tool for supporting expanded diagnostics in upcoming CMIP7 experiments."

2. *In section 2.3 you briefly discuss the output of the VISION tool. Is the output CF-compliant? Please mention if so.*

The current version of the tool uses CIS libraries to write output in NetCDF format. Whilst this output can be easily read and processed, it is not fully CF compliant. The next version of the VISION toolkit (currently under development) is using cf-python libraries to write model output which will be fully CF compliant.

3. *Table 2: you test different I/O libraries for NetCDF on performance. Surprisingly, the cf-python library is faster when reading a pp file with 36 fields than a single field. Could you elaborate on this?*

The JASMIN facility, on which tests described in Table 2 were performed, is a shared computing facility and the load on the system can vary at different times throughout the day. To avoid bias in the comparison of different libraries reading the same type of input file, all tests in the same column in Table 2 were performed simultaneously on the system. However, tests in different columns in Table 2 were performed at different times and this might result in perceived inconsistencies such as the one highlighted by the reviewer. Given that another reviewer has questioned the use of old version of the iris library, we have repeated these tests with the most recent versions of the three libraries and we have performed the tests on a local computer cluster to further minimise the impact of variable load on the timings in Table 2. This has resulted in more consistent numbers when looking at the performance of a Python library with multiple file formats.

> *In a broader perspective, I'm not sure whether a numpy.print is a good indicator for the I/O performance of the actual VISION workload, especially for distributed lazy I/O libraries under consideration. Maybe extracting values along a trajectory would provide a better indication of the performance? Please address this concern in the text.*

Initial timing tests using the VISION toolkit, identified reading of the model data as the single, most time-consuming step compared to reading of the observational data, extracting the values along a trajectory and writing the output. Therefore, we decided to focus our investigations on the time required to read the model data using different Python libraries, rather than the time required to read model and observational data and extract the value along a trajectory. Since all the libraries used in the comparison store the data as a numpy array, using a numpy.print statement to access the variable data array was the simplest possible way to avoid lazy loading of the data.

To clarify this, the following sentence has been added to the manuscript on page 5, line 134:

“Initial timing tests using the VISION toolkit, identified reading of the model data as the single, most time-consuming step compared to reading of the observational data, extracting the values along a trajectory and writing the output. Therefore, the time required to read model data using different Python libraries was investigated.”

4. *Section 4: the examples only involve ozone concentration. Since the tool is presented as a general-purpose interpolator/collocator, one would expect multiple variables to be plotted for illustration.*

In this paper, ozone was chosen as an example because it is a widely measured chemical quantity with a large volume of data going further back in time. However, ISO_simulator can easily produce any modelled chemical and dynamical variable along specified tracks. To showcase this further, Fig 4 in the manuscript has been edited and the new figure contains two new panels with UKESM carbon monoxide and temperature, as well as UKESM ozone, along the FAAM flight track.