MESMER v1.0.0: Consolidating the Modular Earth System Model Emulator into a Sustainable Research Software Package

Final Author Response

Response to Referee Comments 1

Comment 0

This paper presents the modular climate model emulator (MESMER v1), its components, and strategy for integration, and serves as detailed documentation in addition to the online documentation for users. The authors also demonstrate that this rewritten emulator now meets sustainable research software standards. The output of the emulator includes annual and monthly mean temperature and several climate extreme indicators, such as maximum annual temperature. The paper provides a thorough description of the emulator, improvements made as well as performance and sustainability assessment. However, prior to publication, the following comments and questions should be addressed:

Response

Thank you for the positive evaluation of our study and the helpful comments. We address the comments made below and explain the modifications we made to the study.

Comment 1 - Motivation of the study

The motivation of this study appears to focus mainly on making the emulator accessible and providing documentation on how to use it (as inferred from the introduction: Lines 30-31). However,

users may also want to be able to read and understand the model source code, extend and modify its functionality, and contribute to maintenance (if possible, e.g., by reporting bugs) in order to improve the overall software sustainability of the emulator. Therefore, I recommend revising the motivation around lines 30–31 to more fully expand on these aspects.

Response

Thank you for this comment, we agree the motivation in the paragraph is too narrow. We added the following in line 31:
The emulators' source code should be easily available, understandable and extensible in order to open possibilities for new applications and further democratize climate information.

Comment 2 - State of the old MESMER software

What is currently not clear to me is the state of the old (legacy) MESMER software. You only hint at this with statements such as organic growth of the code base (Section 1: Introduction, lines 36–38) and development of software by multiple researchers without a focus on software design (Section 3: Design of MESMER v1, lines 98–99). Was documentation available for the old software? Was the old software easily accessible by users? Is the code of the legacy MESMER difficult to read and understand? What is the state of code comments in the old code? Were the separate components of old MESMER all integrated as shown in this paper? Did the old software have a well-defined software architecture? I expected to see a section that answers and expands on these points.
This would also be beneficial during your assessment (Section 3.6: Assessment of the redesigned code base), as it would help to understand how much the rewritten code improves upon the old one. Additionally, did the state of the old code require rewriting everything from scratch (or only parts of it)? This also highlights the importance of making software sustainable, as complete rewriting requires significantly more time and effort. This is an issue that is now slowly gaining attention in academia.

Response

We now added a subsection “State of the legacy Software” to section 2, which should answer these questions. We did not have to rewrite MESMER from scratch. Mostly, we reused the basic statistical functionalities, but split them up into modular bits, built new APIs for them, wrote wrappers for the new data-structure, and unified common functionalities between the different components. For the manuscript however, we prefer to emphasise the key features of the new software for potential users, and somewhat less on the shortcomings of the legacy code or the process of the rewriting itself.

Previously, the individual MESMER components listed above were published in separate repositories, or, in the case of MESMER-X, not accessible for the public at all. Overall, there was no overarching software architecture. While all components were written in Python, the structure varied across components. Some components were written in an object-oriented style, others were a collection of scripts that had to be executed in the right order, and others were collections of function definitions and execution code in single Jupyter Notebooks. Across all four components, there were about 12'000 lines of code in 60 files.

External documentation for MESMER-M, MESMER-M-TP and MESMER-X was limited to the model description papers (Nath et al., 2022; Schöngart et al., 2024, Quillcaille et al, 2002, 2023), whereas MESMER already had a documentation webpage with an API reference. Around 60 % of public function and class definitions were documented with docstrings in the legacy code. The code quality of the legacy components was mixed, scoring pylint scores (Goodger and Rossum, 2011) between 0 and 8.03, were 0 is the smallest (worst) score possible and 10 the highest (best).

The comprehensibility of the software was further hampered by a rather in-transparent data structure consisting of nested dictionaries, where data was stored as plain numpy arrays (Harris et al., 2020). To navigate the data one had to know the dictionary structure, the appropriate keys, and data dimensions. Lastly, MESMER-M, MESMER-M-TP and MESMER-X had no testing suite, ensuring correctness and stability of the Code. As of the release of MESMER v0.8.3 in Hauser et al, 2021, MESMER had a test coverage of around 78 %.

Comment 3 - Performance Assessment

Performance assessment (Section 5) was mainly focused on runtime. It would be helpful for the paper to also include some visualized plots showing the reproduction of old results or even improvements over previous results. For example, in Section 4.3, Switch to a cyclo-stationary AR(1) process for monthly emulations, you state that the cyclo-stationary AR(1) represents variability better than the AR(1) process. However, the corresponding plot is not shown.

Response

We added the plot showing the comparison variability estimations in the Appendix:

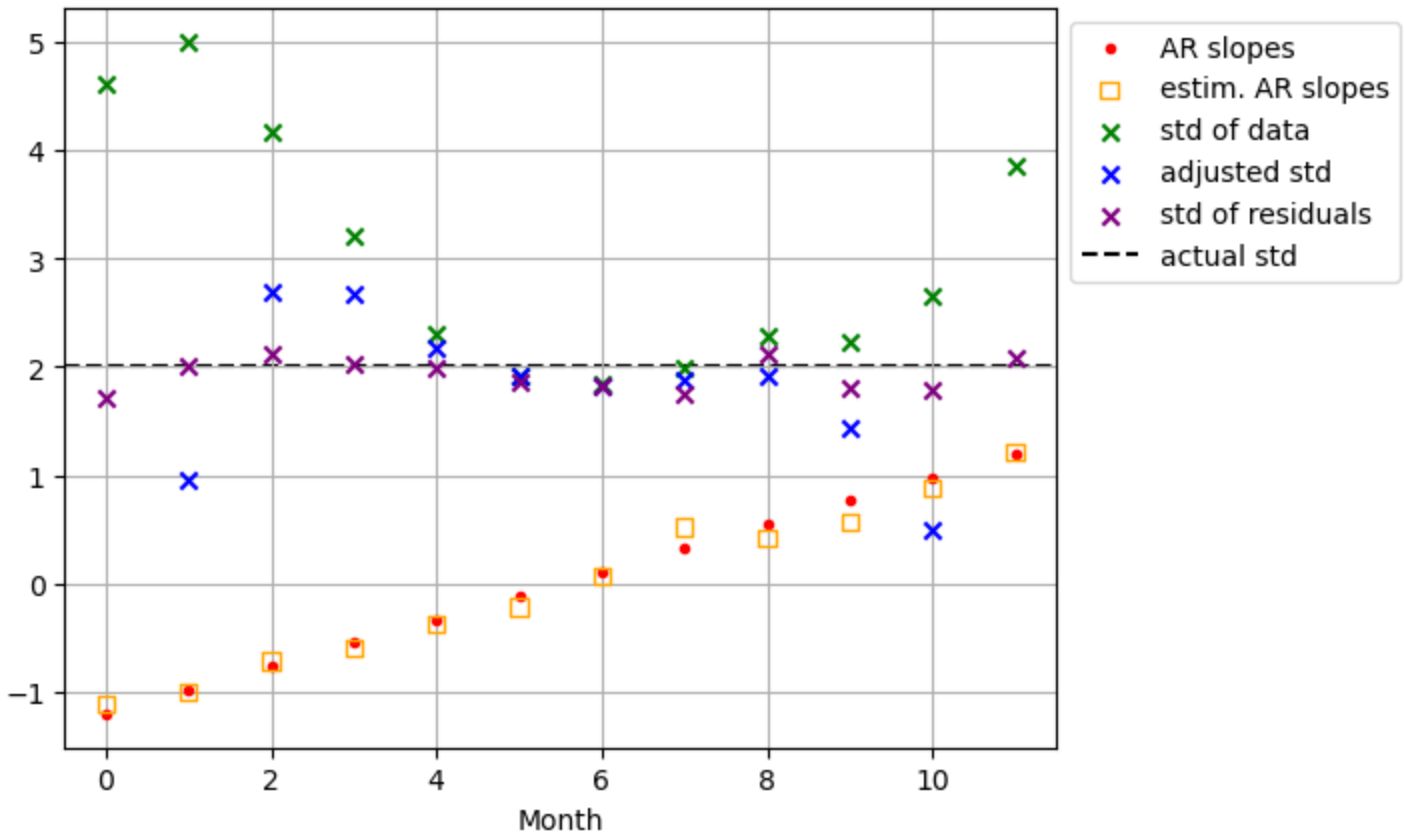

Fig A1: Comparison of methods to estimate the driving noise variance of data generated by a cyclo-stationary AR(1) process. Data was generated for 12 months using AR slopes indicated by red dots and a standard deviation (std) of 2 (actual std - gray stippled line). Yellow squares mark the estimated AR slopes retrieved by MESMER's fitting algorithm. Green crosses indicate the standard deviation estimated from the raw monthly data, blue crosses the one estimated by adjusting the standard deviation with the retrieved AR parameters, and purple crosses the one estimated on the data residuals of the AR process.

Specific Comments

Section 1: Introduction

1. Line 15: Please provide citation(s)

Section 3: Design of MESMER v1

2. Line 102: Please check the citation. Should it be Anzt et al., 2021?
3. Lines 112-113: Is mesmer.proba the same as mesmer.distrib in Figure 2? If so, could you please make it consistent?
4. Lines118-119: For data handling functionality, I don't seem to see mesmer.anomaly.calc_anomaly in Fig. 2. (mesmer),
5. Lines 121-124: Making this workflow for calibration and emulation creation readily available would already benefit users and improve reproducibility. Is this workflow already available?
6. Lines 154-156: Please provide citation.
7. Line 206: Should this be 238 instead of 269 as in Table 2?
8. Line 276-278: " … We were able to fix these issues by switching to more precise routines", This line is unclear to me. Which routines were used to address the numerical stability problem?

Section 5: Performance assessment

9. There are several instances where runtimes are mentioned as referring to Table 3, but Table 3 is not actually referenced. For example, in Lines 367–368 and 390–391. Please revise these references to make them clear
10. Line 355: Should this be 3 minutes rather than 4 minutes?
11. Line 400: Should this be 30 seconds rather than 1 minute?

Response

1. We added the citation Tebaldi, C., Selin, N., Ferrari, R., and Flierl, G.: Emulators of Climate Model Output, Annual Review of Environment and Resources, 50, 709–737, https://doi.org/10.1146/annurev-environ-012125-085838, 2025.
2. Yes, thank you, we corrected all mentions of the wrong citation.
3. Yes, thank you, we corrected all instances of mesmer.proba to mesmer.distrib in the text (the figure was correct and remains unchanged).
4. It is at the very bottom, in the middle in Figure 2.
5. This workflow is demonstrated in the MESMER tutorials. We refrained from adding high level functions that execute these workflows in one go, to keep flexibility and modularity of the software high. We added this text in line 124: *This workflow is demonstrated in the tutorials delivered with MESMER and on the documentation webpage.*

6. We added the citation Rew, R. and Davis, G.: NetCDF: an interface for scientific data access, IEEE Computer Graphics and Applications, 10, 76–82, https://doi.org/10.1109/38.56302, 1990.
7. Thanks for spotting this, we adjusted the number in the text.
8. We added the following text:
   We were able to fix these issues by switching to more precise numpy routines in the computation of the transformed data (namely numpy.expm1(x) and numpy.log1p(x) that yield higher precision for small x than numpy.exp(x)-1 and numpy.log(1+x), cf. https://numpy.org/doc/stable/reference/generated/numpy.expm1.html, last accessed: 12.3.2026).
9. We added the missing references in the text.
10. Yes, thank you, we adjusted the number in the text.
11. Yes, thank you, we adjusted the number in the text.

Other questions

Is there a containerized solution for the new MESMER, such as a Docker container, to enable running it on different computers and facilitate reproducibility?

Response

No, we do not provide a Docker container, but we do test on multiple operating systems and python versions and provide detailed documentation on our dependencies.

Note to the referees and editors

We will release the MESMER v1.0.0 upon final acceptance of the manuscript. We uploaded an updated version of the analysis scripts to https://doi.org/10.5281/zenodo.19203528. Some of the numbers in Table 2 changed slightly, due to ongoing developments in the MESMER Code base, see https://mesmer-emulator.readthedocs.io/en/latest/changelog.html#v1-0-0-unreleased and screenshot of latexdiff of Table 2 in the manuscript below. The corresponding information in the text has been updated as well.

Target	Metric	Value
Modularity	Average number of lines of code per object	~~38~~ 37
	Number of lines of code per file (avg [min, max])	~~238~~ 244 [7, 1219]
Code quality	Pylint score	8.74
Data structure	Numpy arrays to xarray data structure	Completed
Documentation	% of public API documented	~~100~~99 %
	% of private API documented	53 %
Testing	% of lines covered by tests	99 %