2. Rebuttal

The second point-by-point response to the reviews including a list of all relevant changes made in the manuscript *"FINAM - is not a model (v1.0): a new Python-based model coupling framework"*.

Sebastian Müller Martin Lange Thomas Fischer
Sara König Matthias Kelbling Jeisson Javier Leal Rojas
Stephan Thober

February 25, 2025

Chapter 1

Reviewer: Anonymous referee #3

1.1 General Comments:

Original comment:

It is an honor to review such a technically oriented research paper. All the papers I have reviewed before focused on scientific issues, and technical problems were usually solved by researchers themselves while addressing scientific questions. However, this paper attempts to solve many common problems that researchers often encounter and provides an interesting solution. I am not professionally engaged in developing coding tools, so I may not fully understand some aspects, but I believe this work is meaningful and can provide convenience for researchers.

Response:

Thank you for your positive feedback and for recognizing the value of our approach. Our main goal is in fact to simplify and automate aspects of model coupling that are often cumbersome in research workflows. We appreciate that you find this work meaningful and convenient for researchers and hope that our explanations and examples clarify the technical details for a broader audience.

1.2 Specific Comments:

Original comment:

1. In my own research, I often deal with model coupling, such as using the output of one model as the input for subsequent models. However, many models are encapsulated, and we cannot access their source code. Such models should not be couplable in the FINAM system, which limits the application scope of FINAM.

Response:

We acknowledge that fully encapsulated models can be challenging to couple if no mechanisms exist to expose their inputs and outputs. FINAM's approach

relies on 'wrapping' models with a Python-based interface, where at least a portion of the model's functionality (e.g. initialization, stepping, data exchange) is accessible. If the model is entirely closed source or offers no means of extracting or injecting data programmatically, it cannot be directly coupled within FINAM. However, as some users employ scripting interfaces or configuration-based I/O even for closed source applications (e.g., controlling runs via command-line arguments, batch scripts or APIs), these can sometimes be wrapped with a 'black-box' style component in FINAM. In other words, if there is any programmatic handle - although minimal - we can integrate that into FINAM with suitable wrappers. We already mentioned file-based couplings as the first approach to combine models in the Introduction.

Manuscript changes:

We added a clarifying paragraph (Section 2.2): "If no Python bindings of the model exist, but it can be run as a black box for a single time-step, there is also the possibility to create a component that prepares the required input files for each time-step, calls the model, and reads the output files to provide the data in the FINAM composition. But be aware that this approach may introduce performance bottlenecks since it is basically a file based coupling."

Original comment:

2. Through the example in Figure 1, I indeed grasped the purpose of the FINAM tool. In similar computations, it is often necessary to read data, output intermediate variables, read them again for further calculation, and then output the results. This is a very cumbersome process. The FINAM tool has been encapsulated into a Python package, eliminating a lot of intermediate work and directly outputting results. I wonder if this process can also be edited using Python's parallel computing syntax to achieve multithreaded computation?

Response:

FINAM is currently designed for serial execution and does not natively support parallelization through multithreading or MPI as already mentioned. However, one could think about integrating Python-based parallel computing approaches (e.g., using `multiprocessing`, `joblib`, `dask`, etc.) around FINAM if certain components are naturally parallelizable. In that scenario, FINAM would still manage the data flow between components, while parallel execution of selected tasks could be handled at the Python level or by the native models themselves. We note that true multithreaded speedups in Python can be limited by the Global Interpreter Lock (GIL), unless one uses libraries that release the GIL (e.g., NumPy with native code sections). Full MPI-based parallelization is currently out of scope, but we plan to investigate how FINAM can interface with parallel libraries in future work as mentioned in the discussion.

Manuscript changes:

We added the following paragraph in the Discussion: "There are also Python-based parallelization approaches (multiprocessing[1], joblib[2], dask[3], etc.) that could be used in the future to run independent parts of the composition in parallel."

1. https://docs.python.org/3/library/multiprocessing.html
2. https://joblib.readthedocs.io
3. https://www.dask.org/

Original comment:

3. Although it is difficult to directly couple encapsulated models with other models, it seems possible to use syntax to drive the model for computation, output results, and then couple the result file with other models. If this can be operated within the FINAM framework, it could also reduce simulation time to some extent.

Response:

Yes, a "file-based" or "black-box" approach can be used for models that do not expose sufficient source code or direct I/O routines. In this scenario, a FINAM component wrapper orchestrates the external model run by generating the necessary input files, executing the model, and reading its output files back into FINAM for subsequent coupling steps. While this still involves intermediate file I/O, it can be streamlined by a single controlling workflow in Python, potentially saving user time and reducing the complexity of manual data handling. However, the simulation time benefit depends heavily on how frequently data needs to be exchanged. For very frequent exchanges, file-based I/O may become a bottleneck. FINAM aims to be flexible enough to accommodate this style of coupling, but its primary design still favors in-memory data transfer for accessible source code or APIs.

Manuscript changes:

We expanded Section 2.2 as described above.

Original comment:

4. In the field of hydrology, surrogate or alternative models have been very popular recently. A complex model can be learned using regression algorithms to understand its inputs and outputs, and then the constructed regression model can be used as an alternative model. It appears that coupling alternative models directly with FINAM would be much easier.

Response:

We agree that surrogate modeling is becoming increasingly prominent, particularly in hydrology and other domains where computationally heavy models can be approximated by faster regressions or machine learning algorithms. FINAM's Python-centric design is well-suited to integrating such surrogate models, as these models often come in the form of Python packages or can be accessed via scikit-learn, TensorFlow, PyTorch, etc. Wrapping a surrogate model in FINAM typically involves creating a wrapper that runs the regression or neural network for each time step, and the rest of FINAM handles data exchange with other components. This can be done in the exact same way as we described the PET component. We see this as a major advantage for users who wish to experiment with hybrid or alternative modeling approaches without altering the underlying code of the original more complex model. Since the model structure is not relevant for FINAM we do not want to add another paragraph for specific types of models, since any time-step-based model is suitable to be coupled with FINAM.

Manuscript changes:

In the Summary, we now acknowledge that surrogate models are candidates for couplings.

Original comment:

5. It is also currently popular to couple machine learning models with traditional models. Could machine learning be considered for future development?

Response:

Yes, machine learning (ML) integration is one possible use case for FINAM. Since FINAM is built on Python, it is relatively straightforward to couple ML models and traditional numerical models within the same workflow since they only need to be wrapped in a component. As the internal model structure (like process-based or ML-based) is irrelevant for FINAM, we would like to not bloat the text further with specific sorts of models or give the impression FINAM has a special relation to machine learning.

Manuscript changes:

In the Summary we now acknowledge that ML models are candidates for couplings.

Chapter 2

Reviewer: Nils-Arne Dreier

2.1 General Comments:

Original comment:

The article "FINAM - is not a model (v1.0): a new Python-based model coupling framework" describes a novel coupling framework whose main focus is to simplify the coupling configuration, with Python as the fundamental language for composing model components. The authors provide a detailed explanation of the concepts of FINAM and explain the decisions made during the development process. Furthermore, they provide examples of how to use FINAM.

This article is fitting within the journal's context and is of high quality, hence, my recommendation for its publication with minor revisions.

Response:

Thank you for your positive feedback and your recommendation for publication. We are pleased that you find our approach and detailed explanations aligned with the scope of the journal.

2.2 Specific Comments:

2.2.1 Introduction:

L5:

Original comment:

"such as YAC, ESMF, or OASIS" I wondered if there was a specific reason behind the order of the stated "coupling solutions". If not, it might be advisable to list them alphabetically. The same applies to their mentions later in the text.

Response:

Thank you for pointing this out. There is no particular reason for the existing order. We will adjust the text to list them alphabetically for consistency and clarity. Where these coupling solutions are cited again, we will maintain the alphabetical order.

Manuscript changes:

In the Abstract and other relevant sections, we changed the list to: "ESMF, OASIS or YAC." In other places like the Introduction we kept the order (OASIS, YAC and ESMF) since this is the logical order to introduce FINAM (first plain couplers, then frameworks).

2.2.2 Section 2:

Original comment:

By the conclusion of this section, I became curious about any other adapters and utilities that FINAM provides to formulate coupled experiments. While I found this information in the online documentation, it would be helpful if the authors could include a brief summary of the key adapters and utilities towards the end of this section to offer a deeper understanding of the potent adapter concept.

Response:

We appreciate this suggestion. We realize that providing a short overview of existing adapters and utilities within the manuscript clarifies the potential of FINAM for new readers without requiring them to consult external documentation. We will briefly summarize the available adapters (e.g., for file-based I/O, time shift/delay, regridding) and mention how new adapters can be implemented.

Manuscript changes:

We added a subsection "2.3 Key Adapters and Utilities" that lists commonly used adapters (e.g., DelayFixed, Regrid, FileIO) and briefly describes their functionalities, referencing the online documentation for more detailed usage examples.

L151:

Original comment:

"When units are not equivalent, like L/m2 and mm, but compatible, like K and °C, they are converted automatically." I'm a bit confused with the units here:

- *I don't understand why L/m^2 (volume/area = length) and mm (length) are not equivalent?*
- *The authors might need to explicitly define what they mean by "equivalent" and "compatible".*

Response:

Thank you for highlighting this confusion. The original wording was indeed ambiguous. In our implementation, we differentiate between units that require a conversion (e.g., °C to K), and those that are essentially the same (e.g., L/m^2 and mm).

Manuscript changes:

We updated the sentence to: "Data with compatible units such as K and °C will be automatically converted. Equivalent units such as L/m^2 and mm will not cause a conversion."

2.2.3 Section 3.1:

Original comment:

From my understanding, integrating the "DelayFixed" adapter changes equation (2) so that $sm(t-1)$ instead of $sm(t)$ is used. This should be explicitly highlighted.

I was also wondering whether the system of equations could be solved without modifying the actual timestepping scheme, e.g. by using a fixed-point iteration or Newton solver. This would require recomputing timesteps of a component or compute differentials. I understand that this goes beyond the scope of this paper and could be considered in future work.

Response:

We agree that the "DelayFixed" adapter effectively shifts the time dependency so that the components use the state of the previous time step instead of the current one. This is essential for modeling lagged dependencies or simulating bidirectional couplings where circular references would otherwise appear.

Regarding the possibility of solving the system via fixed-point iterations, Newton solvers, or more advanced numerical schemes, we think this would imply a huge impact on model developers since models would need to be able to save their state and reset if needed. As our focus is on models that were developed for stand-alone usage, we think this to be a major requirement that would raise the bar to high for many users. FINAM currently focuses on explicit time stepping at the component level, but an extension to iterative or implicit coupling methods might be valuable for problems requiring tighter convergence or strongly coupled feedbacks but would need an investigation how to incorporate that with the current focus.

Manuscript changes:

We added a sentence in Section 3.1 clarifying that: *"Integrating the DelayFixed adapter replaces $sm(t)$ with $sm(t-1)$ in the coupled equation, thus delaying the effect of changes by one timestep."*
