Review of the manuscript “Deploying Machine Learning components coupled to Earth System Models with OASIS3-MCT (v1) and Eophis (v1.1)”

In this manuscript, authors device Eophis, wrapping the python-side OASIS3-MCT, as the engine that manages the time stepping. Using NEMO as the earth system model (ESM) that couples to two machine learning (ML) based algorithms, authors assessed the performance of the resulting product. The analysis is further divided into synchronous and asynchronous execution strategies. Authors demonstrated that asynchronous strategy is preferred. The manuscript focuses on the idea and the technical side of the tool. Although it has a theme, there is not a central scientific question to work on in the manuscript.

This tool follows ongoing trend in which hybrid modeling becomes more common. The main contribution of this work is the coupling strategy that puts high-level language (python) as the engine, rather than the lower-level one (Fortran/C/C++). Theoretical or idealized studies can take advantage of this framework to develop and test their own simple model that couples to an existing model (NEMO, MITgcm, MOM, …).

Still, there are places in the manuscript that can potentially be refined. I recommend minor revision, and list my comments below.

Minor Comment

In the scalability discussion, I wonder if adding some cost analysis can enhance the understanding of the resulting curve seen in Figure 7? I provide what I meant as below, but it is up to authors to decide if such an extension is desirable.

Suppose the execution time of Fortran model is τ_F and the python model is τ_P, the cost-of-time needed to advance the coupled model for one time step τ using the synchronous strategy is

$$\tau_s = \tau_F + \tau_P$$

where negligible overhead is assumed. The cost of time for asynchronous strategy is

$$\tau_a = \max(\tau_F, \tau_P)$$

If overhead times are generally negligible (supported by author’s Figure 8), this means $\tau_a < \tau_s$. If the cost of python execution time is perfectly scalable

$$\tau_P = \frac{\tau_P^*}{N_P}$$

where N_P is the amount of CPU used for python, and τ_P^* is cost of time when $N_P = 1$. We can see

$$\tau_s = \tau_F + \frac{\tau_P^*}{N_P} = \tau_F \left(1 + \frac{\tau_P^*}{\tau_F} N_P^{-1}\right)$$

$$\tau_a = \max\left(\tau_F, \frac{\tau_P^*}{N_P}\right) = \tau_F \max\left(1, \frac{\tau_P^*}{\tau_F} N_P^{-1}\right)$$

Now, since the simulated year per day (SYPD) equals execution time ΔT divided by cost-of-time, we see

$$SYPD_s = \frac{\Delta T}{\tau_F} \cdot \left(1 + \frac{\tau_P^*}{\tau_F} \cdot 10^{-x}\right)^{-1}$$

$$SYPD_a = \frac{\Delta T}{\tau_F} \cdot \left[\max\left(1, \frac{\tau_P^*}{\tau_F} \cdot 10^{-x}\right)\right]^{-1}$$

Where x is the log-log coordinate as shown in Figure 7 such that $10^x = N_P$. The resulting function can explain the S-shape curve in Figure 7. For synchronous strategy, this curve is S-shaped, and approaches a maximum when $x \to \infty$. For asynchronous strategy, because $\tau_P^* > \tau_F$, when x is small, it behaves like exponential 10^x, when $x > \log(\tau_P^*/\tau_F)$, it is capped by constant 1. Finally, if $\tau_P^* \gg \tau_F$, then $SYPD_s \approx SYPD_a$ at $x = 0$. This analysis seems to match what is observed in Figure 7.

Other comments

1. Line 80: “Another category”: I suggest “Second category” to enhance clarity.
2. Lines 221-222, Figures 5 and 6: I think “grids line” and “exchs line” are a bit awkward. Maybe adding line numbers to figures such that can directly refer as “code lines 6 and 7”.
3. Line 268-269: “the 2D cases are doubled”: needs clarification. What in 2D cases are doubled?
4. For section 5.2 Future perspectives, I wonder if authors can provide some views on the recent effort to revive the usage of model hierarchy by Shaw et al (2025) which seems relevant?
5. Figure 5: Is “ESM, = eophis.register_tunnels(…)” a typo? It looks like the comma on the left hand side is a bug. In python it should be “ESM, _ = …” or “ESM = …”.
6. Figure 1
    - Since in the later example you are going to us variables u, v, maybe you can replace a, b with u, v to help the readers?
    - I would suggest consistently use OASIS3-MCT as in the text throughout the manuscript, rather than “OASIS3” as if this is a different idea.
    - I also think add text to differentiate two “OASIS API” boxes. Such as “OASIS3-MCT API (Python)” and “OASIS3-MCT API (C-Python adapter)”. Current Figure 1 confuses me easily.

Reference

- Shaw, T. A., & Stevens, B. (2025). The other climate crisis. *Nature*, *639*(8056), 877-887.