EAT v0.9.6: a 1D testbed for physical-biogeochemical data assimilation in natural waters

Jorn Bruggeman, Karsten Bolding, Lars Nerger, Anna Teruzzi, Simone Spada, Jozef Skákala, and Stefano Ciavatta

RC1: 'Comment on gmd-2023-238', Anonymous Referee #1, 03 Jan 2024

Citation: https://doi.org/10.5194/gmd-2023-238-RC1

The authors present a data assimilation (DA) test bed based on a 1-dimensional physical-biogeochemical water column model. The manuscript contains 3 example applications for the DA framework, which highlights its versatility regarding different model configurations and DA techniques. It could be a useful tool for beginners to learn about DA, or for practitioners to test modifications to existing DA systems. The manuscript is well written and easy to follow. I tested one of the example applications on a computer, and it ran with just a few small issues. In the manuscript I would like to see a bit more accompanying information about the test cases, in particular, related to how easy it would be to modify some of the implementation aspects.

We would like to thank the referee for their careful review and constructive comments. Our point-by-point replies are provided hereafter in green.

General comments

Overall, the three test cases that are presented in the manuscript, and also included as example applications in the downloadable software, are very instructive and helpful to a potential user of EAT. Each of the test cases appears to highlight an issue or weakness of the chosen modelling/DA approach and made me think of possible extensions of the test cases to further investigate or mitigate those issues. Here, it would be useful to include more information in the manuscript to describe how much user input or work would be required to extend the test cases.

We are very grateful to the referee for having run the test cases. Moreover, we appreciate the opportunity to describe some of the advanced functionality that EAT has to offer. As suggested, we will for each test case include a section that describes how it may be extended. Further details are given below.

Case in point, in test case 1 (Section 3.1) it would be interesting to examine the inclusion of more sources of uncertainty, beside the 3 or 5 parameters that are included in the ensemble generation. I am not suggesting the authors make any changes to the test case, but it would be helpful to describe the amount of change required to include other parameters here (Fig 2 appears to suggest it requires just a small change in one of the python files) or introduce changes to the initial conditions. This additional information could be included in a final paragraph of the section.

We fully agree and will add the following paragraphs to the first application (note that this new paragraph also includes information about subsurface chlorophyll, depth-dependent observations, and custom observation operators, all referred to in later referee comments):

"This application can be extended in several ways. For example, additional sources of uncertainty can be introduced when constructing the ensemble. The current setup includes a primitive parametrization

of meteorological uncertainty through scaling of the surface wind components; more realistic experiments might source an ensemble of different meteorological model realizations (i.e., separate meteorological forcing files) and distribute those over the EAT ensemble members. EAT facilitates this by allowing its ensemble generators to set YAML parameters (e.g., the location of meteorological forcing in gotm.yaml) to member-specific file paths, similar to how the biogeochemical configuration (`fabm/yaml_file`) is treated in Fig. 2. Another option is to introduce uncertainty in biogeochemical parameters other than phytoplankton maximum growth rate. This is easy to realize: all biogeochemical parameters are set in fabm.yaml and any of these can be varied across the ensemble by adding a single line in the ensemble generation script as in Fig. 2. Finally, it is possible to vary physical and biogeochemical initial conditions across the ensemble, as shown in the last section of Fig. 2.

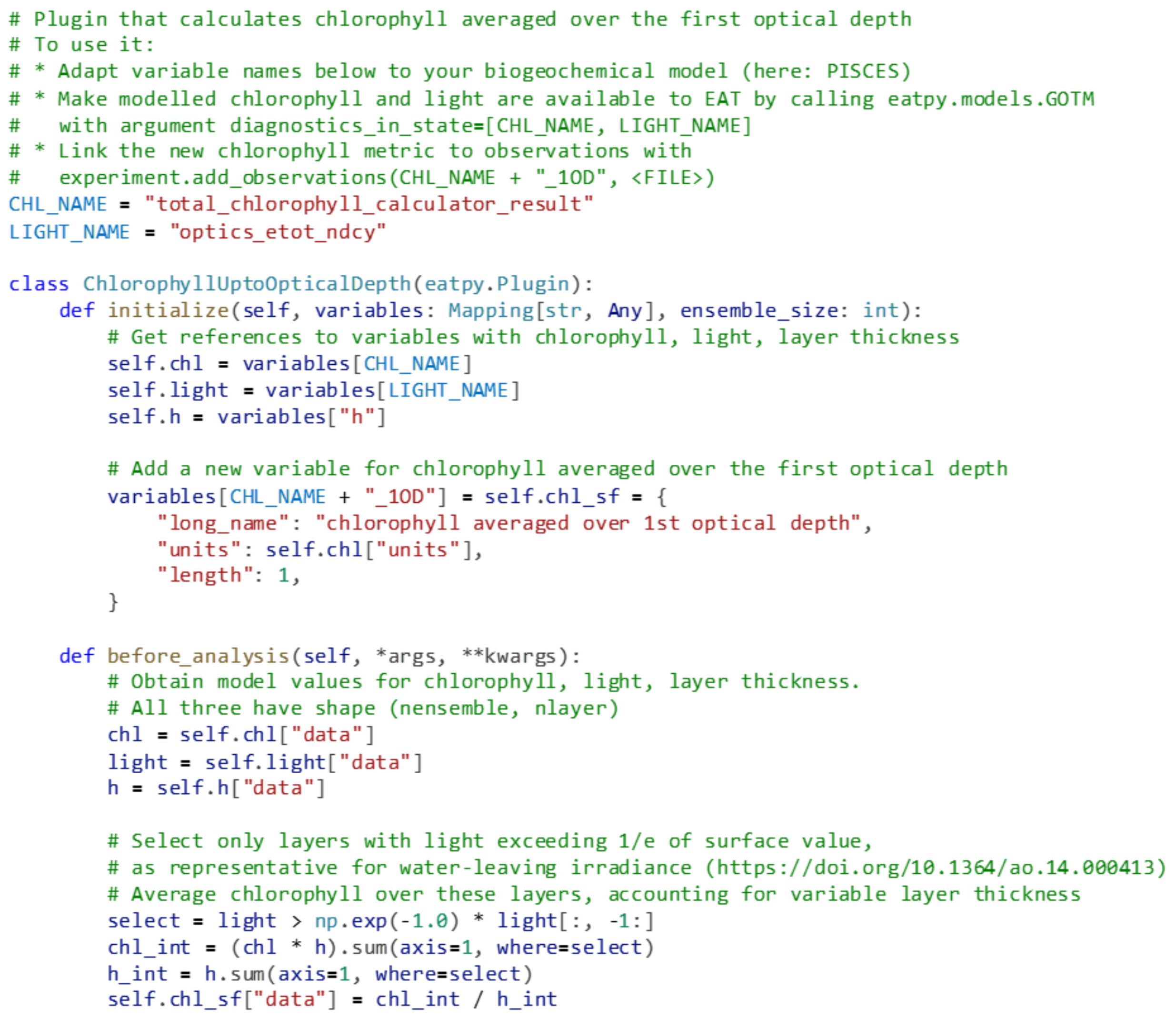
Another possible extension is to assimilate observations that describe not just the water surface, but also deeper layers. Notably, the inclusion of depth-explicit biogeochemical observations, e.g., from ship-based casts, automatic profilers or Argo floats, might help determine whether the decrease in subsurface chlorophyll in Fig. 6 is realistic or an artifact of surface-only chlorophyll assimilation. Inclusion of depth-explicit observations in EAT is straightforward, as it merely requires adding a column with depth information to the observation file and dropping the depth index (-1) in the linked model variable (Fig. 3). Another option would be to account for the fact that remotely sensed chlorophyll observations may be representative for a greater depth range than just the very first model layer, which here is just 10 cm thick (Gordon and McCluney, 1975). For instance, Fig. 4 demonstrates how EAT's plugin architecture can be used to define a custom observation operator that calculates the average chlorophyll concentration over the first optical depth in the model, which varies both over time and between ensemble members (as these differ in chlorophyll-derived light attenuation). By linking remotely sensed chlorophyll to this custom chlorophyll metric, biogeochemistry at depth will be better constrained."

Figure 2 will be updated to illustrate the described functionality:

```
# Vary the physical and biogeochemical configuration across the N ensemble members.
# We apply log-normally distributed scale factors to:
# * wind speeds (x and y components)
# * background mixing (minimum turbulent kinetic energy)
# * maximum growth rates of the two phytoplankton types
# Each member will have different configuration files for physics (gotm.yaml)
# and biogeochemistry (fabm.yaml). The path to the latter is set in the former.
# Configuration files for each member are written when the "with" clause exits.
rng = np.random.default_rng()
gotm = eatpy.models.gotm.YAMLEnsemble("gotm.yaml", N)
fabm = eatpy.models.gotm.YAMLEnsemble("fabm.yaml", N)
with gotm, fabm:
    gotm["surface/u10/scale_factor"] = rng.lognormal(mean=0.0, sigma=0.2, size=N)
    gotm["surface/v10/scale_factor"] = rng.lognormal(mean=0.0, sigma=0.2, size=N)
    gotm["turbulence/turb_param/k_min"] *= rng.lognormal(mean=0.0, sigma=0.2, size=N)
    gotm["fabm/yaml_file"] = fabm.file_paths
    fabm["instances/phy/parameters/mumax0"] *= rng.lognormal(mean=0.0, sigma=0.2, size=N)
    fabm["instances/dia/parameters/mumax0"] *= rng.lognormal(mean=0.0, sigma=0.2, size=N)

# Vary the initial state across the N ensemble members, using log-normally distributed
# scale factors drawn for each member and variable.
# Restart files for each member are written when the "with" clause exits.
restart = eatpy.models.gotm.RestartEnsemble("restart.nc", N)
with restart:
    for name, values in restart.template.items():
        shape = (N,) + (1,) * values.ndim
        scale_factor = rng.lognormal(mean=0.0, sigma=0.2, size=shape)
        restart[name] = values * scale_factor
```

Fig 4 mentioned above will be newly added:



```
# Plugin that calculates chlorophyll averaged over the first optical depth
# To use it:
# * Adapt variable names below to your biogeochemical model (here: PISCES)
# * Make modelled chlorophyll and light are available to EAT by calling eatpy.models.GOTM
#   with argument diagnostics_in_state=[CHL_NAME, LIGHT_NAME]
# * Link the new chlorophyll metric to observations with
#   experiment.add_observations(CHL_NAME + "_1OD", <FILE>)
CHL_NAME = "total_chlorophyll_calculator_result"
LIGHT_NAME = "optics_etot_ndcy"

class ChlorophyllUptoOpticalDepth(eatpy.Plugin):
    def initialize(self, variables: Mapping[str, Any], ensemble_size: int):
        # Get references to variables with chlorophyll, light, layer thickness
        self.chl = variables[CHL_NAME]
        self.light = variables[LIGHT_NAME]
        self.h = variables["h"]

        # Add a new variable for chlorophyll averaged over the first optical depth
        variables[CHL_NAME + "_1OD"] = self.chl_sf = {
            "long_name": "chlorophyll averaged over 1st optical depth",
            "units": self.chl["units"],
            "length": 1,
        }

    def before_analysis(self, *args, **kwargs):
        # Obtain model values for chlorophyll, light, layer thickness.
        # All three have shape (nensemble, nlayer)
        chl = self.chl["data"]
        light = self.light["data"]
        h = self.h["data"]

        # Select only layers with light exceeding 1/e of surface value,
        # as representative for water-leaving irradiance (https://doi.org/10.1364/ao.14.000413)
        # Average chlorophyll over these layers, accounting for variable layer thickness
        select = light > np.exp(-1.0) * light[:, -1:]
        chl_int = (chl * h).sum(axis=1, where=select)
        h_int = h.sum(axis=1, where=select)
        self.chl_sf["data"] = chl_int / h_int
```

Similarly, in test case 2, the biogeochemical covariance is limited to the phytoplankton variables. Mention how easy it would be to expand it, for example, to include nitrate.

We will add the following paragraph at the end of the variational application:

"This application can be easily extended. To better constrain the behavior of the model at depth, it would be possible to include chlorophyll profiles as part of the assimilation, or to use a custom observation operator that considers a greater depth range when calculating the model equivalent of remotely sensed chlorophyll. Both options are described in more detail for the previous application. Further, the background error covariances used in the variational approach can be reformulated to extend the assimilation to variables other than chlorophyll (e.g., profiles of nitrate or oxygen). This is straightforward to implement, as the background error covariances in EAT are already controlled by a user-defined plugin when using variational assimilation. A combination of plugins can be used to fully specify how biogeochemical variables are affected by the assimilation: to control which model variables are of interest for assimilation (the Select plugin in Fig 3), to read and apply predefined vertical error

covariances, and potentially to implement custom schemes that adjust filter-proposed updates or extend them to others variables, e.g., by applying preservation rules that force ratios between selected model variables to remain the same before and after DA."

Then there is a (perhaps worrying) decline in subsurface chlorophyll brought by the assimilation of surface chlorophyll. Here, EAT could be a nice test bed to evaluate modified observation operators, would this be an easy thing to implement? For example, could a chlorophyll observation be considered the sum of the top 4 grid cells? Or -- more difficult -- could the observation operator be dynamically determined based on optical depth? Again, a small paragraph suggesting changes to the test case, and a description of the effort that would be involved, would be of use for many future users of EAT.

We expect that the referee refers to the decline in subsurface chlorophyll in Fig. 6, that is, in the first test case (North Sea + PISCES). We agree that it would be worthwhile to evaluate whether this pattern is robust if chlorophyll observations are taken to be representative for a deeper part of the water column, in particular as the top layer in the model is thin (10 cm). EAT lends itself well for such experiments, as new observation operators are easy to implement through plugins. To demonstrate this, we will add a code snippet (new Fig. 4 shown above) that implements exactly the type of observation operator that the referee proposes: it dynamically determines the first optical depth from the daily mean shortwave radiation, and then averages chlorophyll over all layers above this depth.

In test case 3, finally, the authors suggest that perhaps more than one parameter should be included in the estimation. How difficult would it be to include more? Beginners like me do not know, but would be interested in learning more, especially if the change required is small (modification of a YAML file perhaps).

We propose the add the following at the end of the parameter estimation application:

"This application could be extended to estimate biogeochemical parameters beyond the maximum growth rate of diatoms. From a technical point of view, this is straightforward: the list of biogeochemical parameters that are to be estimated is configured in the run script through argument `fabm_parameters_in_state` provided to `eatpy.models.GOTM` (for the application's original run script, see Data availability section). The (scientific) challenge is to decide which specific parameters to target, as some biogeochemical models have several hundreds of parameters. One potential strategy is to first perform sensitivity analysis to determine which biogeochemical parameter have the greatest impact on model results; tools for doing this with GOTM-FABM are readily available (Ciavatta et al., 2022; Andersen et al., 2021). Another important consideration for selecting parameters is to assess which biogeochemical parameters are most likely to exhibit temporal variability. That assessment should highlight parameters that are constant in the original model, but clearly aggregated over multiple processes or functional types, and thus likely to vary temporally in reality as the relative importance of their constituent processes or functional types changes. Finally, it is worth noting that in EAT, parameters that are estimated during assimilation remain the same over the entire water column, even though they become variable in time. This is also common in 3D data assimilation (Doron et al., 2013). However, if parameter variability is assumed to stem from shifts in biological species composition, it is worth noting that the current approach cannot account for e.g. separate "light" and "shade" communities (Sournia, 1982), which would require parameter values to vary in depth. This is beyond the scope of EAT; the relevant parameters would need to be added to the depth-explicit model state within the biogeochemical model code."

Beyond the test cases, it might also be of interest to some to include a description of some more sophisticated features of EAT -- or a brief description of what can be implemented: For example, is there an ability to use pre-computed ensemble members for quick DA experiments without having to re-run the model? The authors further mention hybrid variational-ensemble schemes, are these included in EAT already, what kind of coding abilities are required to add a new DA technique?

We appreciate the opportunity to further highlight the possibility to use EAT for more sophisticated DA experiments. We expect that the newly added discussion of possible extensions for each application partially addresses this. In addition, we propose to modify the third paragraph of the discussion (starting at line 500 in the submitted manuscript version) as follows (novel text in italics):

"With respect to the data assimilation methodology, EAT provides a wide choice of methods, covering both variational data assimilation and sequential estimation methods that include different ensemble Kalman filter variants. This allows users to assess the impact of different assimilation methods under identical modelling conditions. Moreover, the effect of different configurations options or representations of the covariance matrix in parameterized variational methods can be easily examined. **Furthermore, EAT supports hybrid approaches that combine variational and ensemble assimilation. In this case, the user plugin for control variable transforms (an example can be found in the variational application; see Data availability section) gets access to the ensemble state, which allows the user to combine the background-error covariance matrix with the ensemble-based one in a variety of ways (Bannister, 2017).** Lastly, EAT allows to perform twin experiments in which synthetic observations are assimilated and the true state, which is to be estimated, is known. For this, a simple model run with EAT can produce the true state. Synthetic observations are subsequently generated by sampling this true state and adding perturbations. Starting from a different initial state, one can then assimilate these observations. This approach allows one to study what the application of data assimilation can achieve in an optimal case. Likewise, it can give an indication of the ability of certain observations to constrain the model state or parameters. **In general, EAT is flexible: user plugins are given full control over observations, forecasts, and analyses (with the ability to override proposed state updates). Ensemble members can differ in both state and configuration, and ensemble states can be saved and reused through support for restart files. These features can be combined in any number of ways to design new data assimilation experiments. Thus, the applications described here are representative of EAT functionality, but not exhaustive.**

As nice as it is to have a fast DA system, it is limited to a 1D (vertical water column) model. Such a model is useful, but it will not be able to serve as a test bed for many operational systems which use 3D models. As use cases for EAT, the authors mention "practical aspects such as the spatial correlation structure and regionalized setting of estimated parameters" (line 54), even though I would consider these two applications as bad examples for the use of 1D models. Examining the effect of spatial correlations of satellite chlorophyll error, for example, would require a 3D DA setup; regionalized parameter estimates would require a number of 1D models, at least one for each region (many if the region boundaries are not set). Here, it would be good if the authors acknowledged some of the limitations of 1D models in the introduction already.

While the paragraph in question highlights ongoing developments in DA, rather than use cases of 1D systems such as EAT, we appreciate that the reader may come away with the impression that EAT can address both aspects mentioned – spatial correlations structure and regionalization of estimated parameters. We agree that this is not the case when it comes to spatial correlation structure and

therefore will drop this phrase. As for the "regionalized setting of estimated parameters", there is precedent for using EAT for this purpose using exactly the method that the referee describes (multiple 1D setups). Therefore, we propose to leave this part of the sentence in place. We will expand the Discussion to discuss horizontal covariances and the work done with EAT on regionalized parameter estimates:

"Nevertheless, 1D models behave differently from 3D models in some respects. Their physics tends not to exhibit the (bounded) chaotic behavior associated with 3D models (Carrassi et al., 2018), and therefore, they do not show the same sensitivity to initial conditions. For instance, a 1D water column model set up for shallow sites is often fully mixed in winter, with the water temperature converging to the temperature of the overlaying air. At that moment, any initial variations in water temperature across any ensemble disappear. If ensemble members differ only in water temperature, its spread then collapses entirely, causing ensemble methods to fail. 1D data assimilation therefore depends on additional methods for generating ensemble spread, e.g., by perturbing forcing or parameters of physical and biogeochemical processes. EAT includes flexible ensemble perturbation logic specifically for this purpose. While this is crucial for 1D applications, it can also be helpful to explore alternative perturbations strategies that are under consideration for 3D application. **Another reason why 1D data assimilation systems such as EAT cannot be fully consistent with 3D is the additional requirement of the horizontal spatial correlation structure in 3D, and through that structure, the impact of geographically distant observations on the local model state. This is still best investigated in 3D, though we note that remote observations might crudely be represented in 1D by incrementing the observation uncertainty with an estimate of the spatial covariance, potentially derived from 3D simulation results. A related aspect that is difficult to represent in 1D is the regionalized setting of estimated parameters (Brankart et al., 2012), though some aspects of this may be investigated through the use of multiple 1D DA setups across large regions and subsequent analysis of spatial patterns in their results (Skákala et al., 2023).** Finally, 1D models have limitations independent of data assimilation. As they assume horizontal gradients are negligible, they cannot represent conditions in areas dominated by horizontal features, e.g., high-energy horizontal currents (e.g., the Gulf Stream) or convection currents. Fortunately, these areas cover a minor fraction of the open and coastal oceans. Moreover, GOTM includes mechanisms to prescribe horizontal gradients, though these cannot respond to assimilation."

Finally, I wanted to bring up the name of the tool: It is probably too late to change EAT's name, but EAT expands to "Ensemble Assimilation Tool" in my mind (it is easy to miss the "and" in "Ensemble and Assimilation Tool") even though EAT supports variational DA as well. Furthermore, it would have been nice to include "Aquatic" or "Marine" in the abbreviation, but that is just a suggestion if a name change is still on the table.

We appreciate that the name "Ensemble and Assimilation Tool" does suffer from the minor drawbacks mentioned, but we also concur with the referee's judgment that it is too late to change this, as EAT has numerous users already (> 100 people have used it during workshops). They have written scripts that reference the "eatpy" Python package (see e.g. Fig 2 and 3), and who have "eatpy" installed as conda package. Changing the name would break their scripts and prevent them from updating the application with "conda update". We prefer to avoid this. We will however ensure that we consistently highlight its ability to do variational DA as well as ensemble-based DA.

Testing one of the example applications

I decided to run test case 2 "Biogeochemical state estimation with variational assimilation" on a Linux machine. Overall, it worked well and there were only a few minor hiccups. I downloaded the zip file that was referenced in the paper, unzipped it and followed the instructions in the `README.md` file. Installing EAT using conda was straightforward and worked right away. The free run worked flawlessly as well, just the DA experiment is missing the output directory:

```

$ mpiexec -n 1 python runVar.py : -n 1 eat-gotm
INFO:root:Model simulated period: 2019-01-01 00:00:00 - 2019-12-31 18:00:00
Traceback (most recent call last):

  File "/home/user/eat-applications/Variational/BFMvar/runVar.py", line 34, in <module>
experiment.add_plugin(eatpy.plugins.output.NetCDF(outfile))
                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/.conda/envs/eat/lib/python3.12/site-packages/eatpy/plugins/output.py", line 14, in
__init__
    self.nc = netCDF4.Dataset(path, "w")

            ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "src/netCDF4/_netCDF4.pyx", line 2469, in netCDF4._netCDF4.Dataset.__init__
  File "src/netCDF4/_netCDF4.pyx", line 2028, in netCDF4._netCDF4._ensure_nc_success
PermissionError: [Errno 13] Permission denied: 'OUTNC/DAout.nc'
```

A quick `mkdir -p OUTNC` solved the issue, and I would recommend adding this directory to the zip (perhaps with a small placeholder file `OUTNC/README.md` which just mentions that this directory will be used for netCDF output).

Finally, I ran into some issues with the figure creation, too: `launcher.sh` was not executable and after making it (`chmod u+x launcher.sh`), I received odd error messages because the file has dos line breaks.

```
./launcher.sh: line 2: $'\r': command not found
./launcher.sh: line 10: $'\r': command not found
./launcher.sh: line 14: $'\r': command not found
[...]
```

This issue might not be easily reproducible, but the script even created an odd `^M^M` directory. After changing the file format to UNIX, it all worked and created the plots. I am not sure what to do about this particular issue (will UNIX line endings cause issues on Windows machines?) but including a proper shebang `#!/bin/bash` in `launcher.sh` at least stopped the script right away without creating directories etc., so I would at least recommend that. Overall, I was impressed by how well it worked, but I did not yet attempt to make any modifications to the experiment.

We apologize for the inconvenience and much appreciate that the referee has taken the time to work around the issues encountered.

All of the above issues have been addressed in a new release of the applications (v2.1 on Zenodo; https://doi.org/10.5281/zenodo.10463234). Notably, all three applications now use Jupyter Notebooks, which work on all supported platforms (Windows, Mac, Linux).

Specific comments:

L 17: It is unclear what "This" is referring to, I would suggest using "DA".

We will change the text accordingly.

L 78: Here (and maybe already in the abstract), spell out the abbreviations used or include a reference to section 2.

We will change the text accordingly.

L 159: While I appreciate the discussion of the different coupling schemes and implementation details, as a new user of EAT, I would be more interested in what I need to do to run DA for my model. Case in point:

> "Online coupling is achieved by inserting function calls to PDAF. This can be done by augmenting the model source code itself, which then enables simulation of an ensemble of model states in a single execution of the model. Alternatively, for models already capable of ensemble simulations, it can be implemented in dedicated DA code after an ensemble of model results is received. The latter is the approach adopted by EAT. PDAF-specific additions to the code are usually four functions, all of which are placed outside of the actual numerical core of the model. Overall, the online coupling approach reduces the amount of data that needs to be written to files and allows efficient data assimilation, in particular when the forecast phase between two assimilation steps is short compared to the start-up time of a model, as is common for GOTM-FABM."

After reading it, I have learned some fundamentals about PDAF, but I am not sure what kind of effort is required for a typical user. Do I need to implement four functions or is more work required, are these Python or Fortran functions? I would suggest rewriting this paragraph with an emphasis on the EAT implementation and EAT-specific instructions. For example, instead of starting "There are different strategies to couple a model with PDAF. The offline coupling uses ...", one could use "While PDAF supports both offline and online coupling, EAT uses online coupling to connect the model to the DA framework..."

We will rephrase the final part of the PDAF section to focus more specifically on its application and use within EAT:

"While PDAF supports both offline and online coupling (Nerger and Hiller, 2013; Nerger, 2020), EAT uses online coupling to connect the model to the DA framework: the model state is updated as part of the data assimilation step (analysis) while the simulation remains running. EAT stores the ensemble state internally in an array, which is synchronized with the active GOTM-FABM processes before and after the DA update. PDAF exposes numerous configuration options, which include the type of data assimilation filter to use as well as various filters-specific settings. EAT enables the user to set these configuration options in a Python run script. Internally, these options are then forwarded to PDAF functions."

L 186: Is the <RUNSCRIPT> here akin to what is shown in Fig. 3? Maybe add a short explanation to the required input.

Yes, Fig 3 is an example of such a run script. We will explicitly refer to Fig 3 and include an explanation of every input:

"Here, <RUNSCRIPT> is the name of the Python script that defines the data assimilation experiment (Fig. 3), <NENSEMBLE> is the number of ensemble members, and <EXTRA_ARGS> are additional arguments to pass to the model, e.g., --separate_gotm_yaml to indicate different ensemble members use different configurations, or --separate_restart_file to indicate different members use different initial states."

Fig 2: For readability, set the mean values explicitly, even though these are identical to the default.

We will change the code snippet accordingly (see updated Fig 2 above).

Fig 3: I think it would be more useful to add some of the information from the caption to the code (in the form of comments).

We will change the code snippet in Fig 3 accordingly:

```python
import eatpy

# Make the model diagnostic for total chlorophyll available by adding it to the model state.
# This enables us to assimilate chlorophyll observations.
experiment = eatpy.models.GOTM(diagnostics_in_state=["total_chlorophyll_calculator_result"])

# Set up ensemble data assimilation using the Error Subspace Transform Kalman Filter
# (Nerger et al., 2012; https://doi.org/10.1175/MWR-D-11-00102.1)
filter = eatpy.PDAF(eatpy.pdaf.FilterType.ESTKF)

# Identify biogeochemical state variables by checking for an underscore in their name.
# (FABM variable names contain at least one underscore; GOTM physical variable names do not)
bgc_variables = [v for v in experiment.variables if "_" in v]

# Restrict the filter to operating on temperature, salinity and all biogeochemical state variables.
# Notably, other physical variables such as water velocities and turbulent quantities are
# thus not affected by assimilation.
experiment.add_plugin(eatpy.plugins.select.Select(include=["temp", "salt"] + bgc_variables))

# Log-transform all biogeochemical variables. Any associated observations have already been
# log-transformed in preprocessing (therefore, transform_obs=False)
experiment.add_plugin(eatpy.plugins.transform.Log(*bgc_variables, transform_obs=False, minimum=1e-12))

# Link remotely sensed surface temperature and chlorophyll observations to their model
# equivalents. In both cases, the value in the top layer of the model is used.
# This is the *last* model layer in GOTM, specified by index -1.
experiment.add_observations("temp[-1]", "cci_sst.dat")
experiment.add_observations("total_chlorophyll_calculator_result[-1]", "cci_chl.dat")

# Run the experiment
experiment.run(filter)
```

Fig 2 and 3: Mention in the caption that this is Python code.

We will change the captions accordingly.

L 258: Are these text files in CSV (comma-separated values) format? Why not use YAML here as well, or a well-structured custom format to avoid/better catch user error?

Observations must indeed be provided in non-YAML format, specifically, as tab-separated values (TSV). The reason for this is that such files are more suitable for opening/reading as *stream*: EAT does not load the entire observation file in memory but read one line at a time as the simulation progresses. This speeds up initialization, reduces memory consumption, and potentially allows other processes to append to the observation file while the DA experiment is running. TSV is more convenient than YAML for stream reading, as Python's most-used YAML modules (PyYAML, ruamel.yaml) do not expose stream reading as part of their public API. While the benefits of stream reading are modest (in 1D, observation files are usually small, so file read time and memory consumption are minor issues; live addition of observations is uncommon), we feel they are sufficient to prefer TSV over YAML. Moreover, GOTM also uses TSV and TSV-like formats for forcing (e.g. meteorological time series), which means this format would always feature in any EAT experiment. We will summarize this line of reasoning by adding the following:

"[with each line describing observation time, observed depth (only for depth-explicit observations), observed value, and its standard deviation] **in tab-separated value (TSV) format. This format was chosen over structured formats such as YAML because it enables EAT to read in new observations on-demand while the simulation progresses, instead of having to parse each observation file in its entirety upon start-up."**

Fig 5 and following: "(a)" labels are missing from the figures.

We will add the missing labels to the figures.

Fig 5: Science question, only related to the DA result: Are the thin layers of subsurface cooling an effect of including only a few mixing-related sources of uncertainty in the ensemble creation?

The subsurface cooling appears to be the result of a DA induced change in thermocline depth. As described, the predominant result of data assimilation is a reduction in sea surface temperature and deeper mixing, in particular in early summer. However, there are periods in early autumn where this pattern flips: the DA runs then have a higher surface temperature. This is a critical period during which the surface mixing layer gradually extends deeper into the water column due to surface cooling. In the DA run, the higher surface temperature delays this deepening of the mixing layer. As a result, there is a thin water mass that already lies within the surface mixing layer in the free run, but still below that mixing layer when DA is active. As water masses below the mixing layer [thermocline] are relatively cold, this gives rise to the subsurface cooling signal in autumn.

We will summarize this as:

"This pattern generally persists into autumn, although occasionally, warmer surface temperatures in the DA experiment cause a decrease in mixing, and accordingly a shoaling in thermocline depth that manifests as subsurface cooling."

L 322: Is this the result of one "4D" data assimilation cycle with "asynchronous" assimilation, or were multiple cycles performed? Please add this information to the manuscript.

We will explicitly describe this with the following:

"This application uses ensemble-based sequential data assimilation with the Error Subspace Transform Kalman filter (Nerger et al., 2012), using an ensemble of 20 members.**"**

Additionally, we will describe assimilation methods more explicitly in table 1 by using "sequential ensemble-based (ESTKF)" and "variational (3D-Var)"

L 342: Does the assimilation "see" the subsurface chlorophyll maximum, or does the observation operator just work on the top layer of the model?

It operates on the top layer only. We will add the following when describing the observations that are assimilated:

"Both types of observations were mapped to model equivalents in the very top layer of the modelled water column, which is 10 cm thick."

L 352: "their different components" I would not describe carbon as a "component" of phytoplankton, and would suggest changing it to "their elemental composition" or cell quotas if these are considered.

We will rephrase this as:

The BFM model describes the marine lower trophic web through the spatial and temporal evolution of 51 state variables. **BFM uses variable stoichiometry and explicitly represents cycles of carbon, nitrogen, phosphorus, and silicon. Accordingly, it explicitly tracks the fluxes of these elements between its nutrient pools (nitrate, phosphate and silicate) and living functional types (phytoplankton, zooplankton and bacteria)** (Vichi et al., 2020).

L 414: "we further use the "diat-MSP" abbreviation": Initially, I wasn't quite sure what was meant, I'd suggest being more explicit, for example by using "in the following we refer to this parameter as "diat-MSP"".

We will change the text as proposed.

Additional references

Andersen, T. K., Bolding, K., Nielsen, A., Bruggeman, J., Jeppesen, E., & Trolle, D. (2021). How morphology shapes the parameter sensitivity of lake ecosystem models. *Environmental Modelling and Software*, *136*(December 2020), 104945. https://doi.org/10.1016/j.envsoft.2020.104945

Bannister, R. N. (2017). A review of operational methods of variational and ensemble-variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, *143*(703), 607–633. https://doi.org/10.1002/qj.2982

Gordon, H. R., & McCluney, W. R. (1975). Estimation of the Depth of Sunlight Penetration in the Sea for Remote Sensing. *Applied Optics*, *14*(2), 413–416. https://doi.org/10.1364/ao.14.000413

Skákala J, Wakamatsu T, Bertino L, Teruzzi A, Lazzari P, Alvarez E, Cossarini G, Spada S, Nerger L, Vliegen S, Brankart JM, & Brasseur P. (2023). *SEAMLESS Target indicator quality in CMEMS MFCs (D6.1)*. https://doi.org/10.5281/zenodo.10522305

Sournia, A. (1982). Is there a shade flora in the marine plankton? *Journal of Plankton Research*, *4*(2), 391–399. https://doi.org/10.1093/plankt/4.2.391