

Final Response: Andrew Yool

Thank you for your revised manuscript and responses to referees.

In general, I am satisfied that your amendments address the issues that I flagged in my previous assessment. However, I note an issue that I should have spotted before around your manuscript's Code Availability section. Namely, while your manuscript refers to PyESPERv1.0.0, your Zenodo archive instead refers to PyESPERv1.01.01b. I suspect that this latter name may just be one that you have used internally when documenting your code. However, as it is inconsistent with your manuscript, please amend this one way or the other. You may find the easiest thing to do is to create a new Zenodo archive with the correct description, and then to use the DOI to this archive within your manuscript.

Other than this, I am satisfied that you have answered the criticisms of your referees well.

We have created a new Zenodo archive, which is updated in the code availability section 5 as follows:

PyESPERv1.0.0, affiliated files, and analyses files are available through LMD's GitHub page (<https://github.com/LarissaMDias>) and archived through Zenodo (doi: 10.5281/zenodo.15929902).

Thank you for your revised manuscript and response to your (many!) referees.

We thank you for the opportunity to greatly improve the manuscript and code.

I have reviewed these now and am generally satisfied that your revisions and expansions satisfy the majority of the issues raised by your referees. However, I note - particularly in your responses to referee #4 - that your replies appear provisional with reference to planned or anticipated revisions to the model code. While this may just be a colloquial language in your response, it does suggest that the code described by this manuscript is still undergoing changes. While it is to be expected that the code will undergo revision in the future, it is important that this manuscript describes a specific instance (and fixed version number) of the code that is satisfactory to your referees.

We apologize for lack of clarity and have finished all code revisions that were required for this version.

To which end, I would appreciate it if you could clarify your responses to your referees as necessary so that it is absolutely clear where work has been concluded on this version. Where future (even imminent) change is expected, this should be noted, but your manuscript should absolutely be clear on where the final accepted version of your code ends and where "future development" is anticipated. The current "still working on this" tone of some of your responses is not acceptable where a very specific model version is being described and formalised.

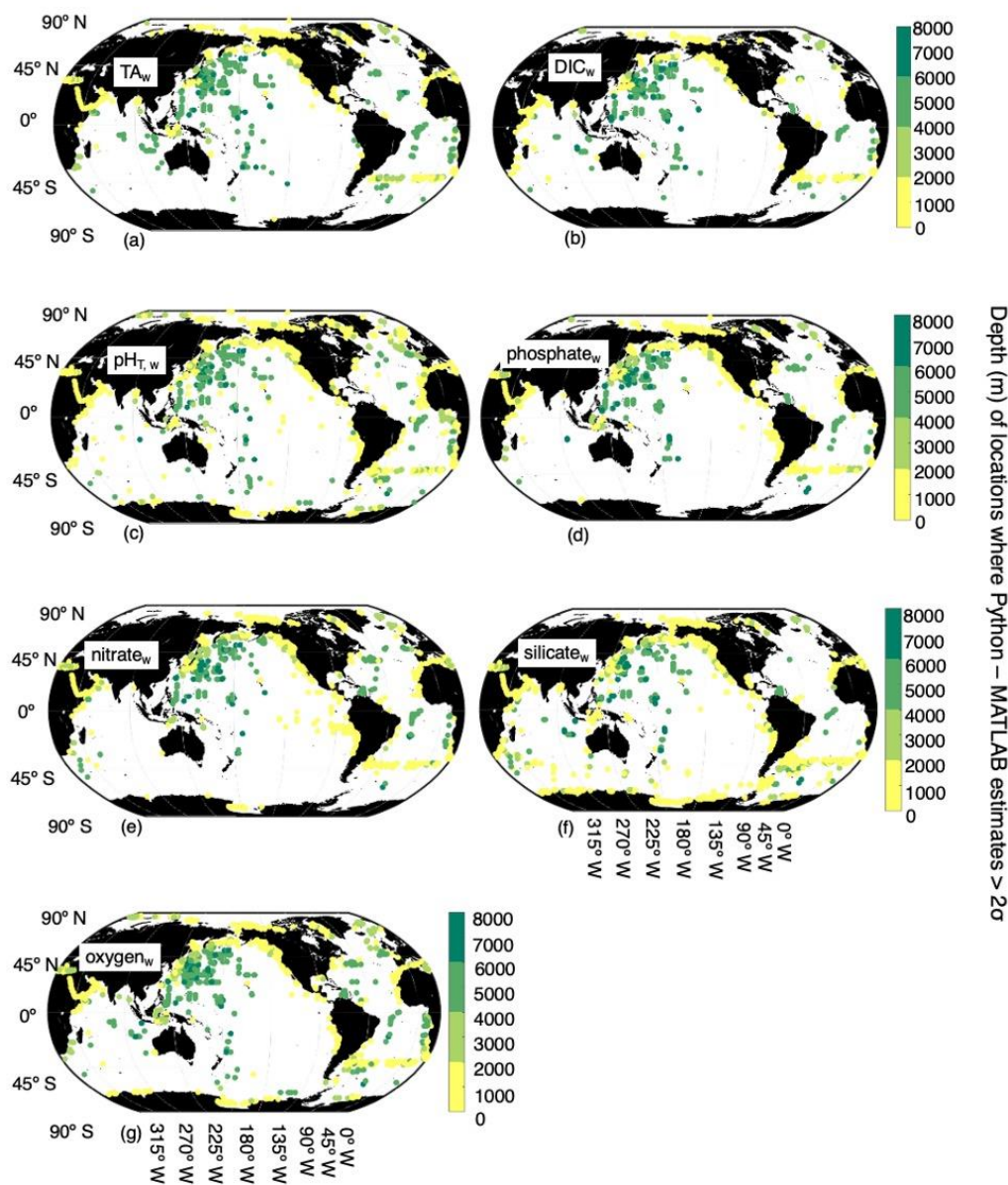
We have clarified the responses to Reviewer #4 and attached those to the end of this document. Specifically, we have addressed all immediate issues and have finished with the current code version.

As such, I am returning your manuscript and response to you for what I expect will be a final iteration. The positivity of your referees means I do not anticipate the need to return your manuscript to them. If you require any additional time to satisfy this request, please do not hesitate to get in contact with me - I appreciate that manuscript revision is in competition with your other research activities (and the upcoming summer leave period).

We thank you for your consideration.

* One further point: Figure B3 uses a rainbow palette that is incompatible with our requirements around colour blindness. Could you please revise this to one of the other palettes that you use that is consistent with this requirement?

We apologize for forgetting to do the colorblind correction of this figure and have updated it to adhere to the guidelines and match Fig. 4. Please see L605 and the improved figure B3 below.



Additional (Edited) Responses to Reviewer #4 (MH):

Review of “PyESPERv1.01.01: A Python implementation of empirical seawater property estimation routines (ESPERs)” by L.M. Dias and B.R. Carter

28th May 2025

Note: Also see Git commits and notes from Matthew email

Overview

[1] This manuscript presents a Python implementation of an existing MATLAB tool to estimate values for various marine carbonate system and nutrient parameters in

seawater. There is no new development of the tool here. The aim is a direct translation. This is a valuable goal, because unlike MATLAB, Python is free and open source. But it does mean that, in my opinion, the quality of the code itself is equally as important as (or even more important than) the manuscript when assessing this submission. The manuscript is primarily describing the new code, so the new code must be complete before publication. I do not think that the code in its current form is complete, usable and publishable, but I think it is possible for this to be achieved within the scope of revisions to this submission (manuscript and code).

We thank you for the constructive comment and agree that the code can be greatly improved quickly with a few minor fixes prior to publication, especially with your advice below and on GitHub.

[2] My sense from looking through the other reviews is that they are mostly focused on the manuscript rather than the code, so this review deliberately focuses mostly on the code rather than the manuscript.

We appreciate the coding expertise.

[3] Although my comments may read as being rather critical, they are all intended to be constructive and I am overall really positive about this manuscript and (more importantly) the code. I'm very happy to see it appear in Python and it's certainly something that I could see myself using in the future. Thank you to the authors for their efforts!

We thank you for the constructive comment.

MATLAB-Python differences

[4] The authors acknowledge that Python code does not produce exactly the same results as the MATLAB. They argue that this is mostly due to differences in how Delaunay triangulation and extrapolation are implemented by the external packages used to do these steps. This argument is plausible but it is not yet convincing. Is there some way the authors can prove that this is the cause of the differences, or at least demonstrate it more quantitatively? See also [8] below.

We have adjusted the explanation as follows and added an appendix (D) to the manuscript that provides an improved and more detailed quantitative explanation and comparison of this issue.

L327-332. Spatial patterns in distribution of outliers shown in Fig. 4 appear to reflect locations where more edge-of-grid biogeochemical measurements were collected (e.g., near coasts and in deep waters). Hence, these locations aligned well with places where coefficients were extrapolated in MATLAB for use in PyESPER_LIRs, compared to interpolations with far away “dummy points” within MATLAB ESPER_LIRs (see Sect. 2.1.1, “Locally interpolated regressions”; Figs. 3, 4, and 5; for w Fig. B2 and B3). Within regions where MATLAB and Python were interpolating similarly, far outliers were uncommon (Figs. 3, 4, 5, B2, and B3).

[5] Continuing on the above, I am worried about the word “most” (“this difference in implementation is the source of most disagreements”; line 87). This implies that there remain some differences that cannot be explained in this way, which presumably points to bugs in the code? See also [9] below.

This choice of wording was indeed misleading, as we believe the interpolation differences beyond machine precision to be entirely due to interpolation differences. See the corrected wording below (simply omitting the word “most”):

L107-109. The three-dimensional interpolation algorithm is implemented differently in MATLAB and Python, and although both calculations are valid, this difference in implementation is the source of disagreements we find and later quantify between ESPER and PyESPER.

[6] The test dataset does include some additional cruises that were not part of the training set but it is not really independent. The additional cruises will have been assessed for consistency with the existing GLODAP product and potentially had their values adjusted to match better.

This is true. However, the GLODAP dataset was also used to validate ESPERs, which was our rationale for choosing this dataset. In future updates, we plan to also validate both ESPERs and PyESPERs against other datasets and potentially model results in an independent analysis.

[7] If I understood Section 3.1.1 correctly (especially lines 260-264), the ‘extrapolation’ areas generally had bigger differences than the ‘interpolation’ areas. This is puzzling. My understanding from Section 2.1.1 (lines 110-128) was that the Python implementation does not extrapolate itself, but rather reads a from saved output for the extrapolation regions generated by the MATLAB implementation. If that’s right, then surely these regions should agree very well with each other, because Python is just copying MATLAB directly rather than doing the calculations internally? Perhaps I have misunderstood the explanations – in which case the corresponding text should be made clearer.

This is a good point and a complicated issue, but it is important to note that Python is not copying MATLAB directly. We have added more information about what the two ESPER versions are doing below and in Appendix D.

MATLAB ESPER_LIR: The grid is expanded vastly (to very large numbers) in order to avoid extrapolation.

Python PyESPER_LIR: The above method resulted in extremely different values due to different triangulation methods in Python. Instead, we extrapolated the grid within MATLAB and used this larger, extrapolated grid to interpolate within Python. After extensive testing of many methods, this was the closest agreement method possible.

Please note that in updates we hope to find interpolation methods that match precisely between MATLAB and Python.

[8] From Figure 2, some of the differences are really rather large (e.g. up to 200 $\mu\text{mol/kg}$ in DIC, 0.5 in pH). Without further evidence I find it hard to understand how or accept that such a large difference could really be due to differences in how Delaunay triangles are calculated. A clearer explanation of this would be appreciated.

Please see the above comments and addition of Appendix D and the table of differences below for a randomly created variable with values between 1-10. Because neural networks agreed to within machine precision, and we have noted these differences between interpolation for the two languages, we can conclude that indeed the interpolation methods introduced the differences.

Table D1: Comparison of differences between MATLAB interpolations and extrapolations and Python results (all interpolations).

	MATLAB Interpolation - Python Interpolation	MATLAB Extrapolation - Python Interpolation
Mean	0.0004	-0.6693
Standard Deviation	0.9559	5.2088
Max	2.2582	13.3083
Min	-2.4593	-15.6633

[9] Were this being released as a data product, then the issues above would be less important, because of the validation against the GLODAP dataset for example. However, this is a tool intended for users to calculate things with untested sets of input conditions. If some part of the differences between implementations are due to bugs in the code, they cannot be written off just because they're fairly small in these tests, because they could easily have a much bigger effect with a different set of inputs. In order to have confidence in the results, any unexpected behaviour or differences between implementations above the level of computer precision must be really thoroughly understood.

This is a valid point, which we believe we have addressed through addition of Appendix D.

Code quality

[10] I was able to get the example code to run but it still required some troubleshooting and corrections to the code beyond the instructions given in the README. These were mostly related to defining and concatenating file paths (which can more robustly and conveniently be done with `os.path.join` rather than by manually manipulating strings). I have made a pull request (PR) to the GitHub repo which contains these and some other (see [11]) fixes (<https://github.com/LarissaMDias/PyESPER/pull/1>).

Thank you for the useful comments. We have accepted and merged all aspects of this pull request.

[11] Parts of the code are very difficult to follow. This makes me worry more about points [5] and [9] above. The most critical issues are:

We agree that (as marine chemists) we have no formal training in coding and the code may be sloppy. We thank you for your careful edits! We have divided code into modules, edited all modules for clarity, and used ruff autolinter/filter (as recommended below) to check for additional errors or fixes and to make the code easier to follow.

- The functions needed are in a Jupyter notebook, so they can't be imported and used in other workflows.

We now have .py modules available. We have also completely eliminated the JupyterNotebooks from the repository.

- There are two notebooks both with copies of these functions – there should only be one “source of truth”.

We now have only one copy of each function.

- Variables are defined, renamed and copied without clear reasons why, making it easy to lose track of which version of a variable should be used for the next step of the calculation.

We have closely edited code and variables within, for a more streamlined code in the final version. Additional explanations have been added for explanation of variable changes within modules.

- The deprecated seawater package is used instead of its well-maintained successor gsw.

This is done for consistency with the current MATLAB version but will be changed to the gsw package for future ESPER updates in both MATLAB and Python. When we used the gsw package within this version of PyESPER, it did not align with results from the current MATLAB version, which uses the deprecated seawater package. We have added an additional message within the PyESPER code (within the “errors.py” module) when sw is used clarifying this to users (please see below).

Error message within errors.py module of PyESPER: “Please note that, for consistency with MATLAB ESPERv1, the now-deprecated sw package is used. This will be replaced with gsw in future updates.”

- It's virtually never necessary to explicitly use global variables in Python and best practice to avoid doing so.

Thank you, we have removed all unnecessary global variables.

- Numerical data appear to be processed into strings at some points?

The iterations.py module required string formatting where indexing within arrays according to string labeling (lines 160-168 of this module, see below):

```
for v in range(0, len(salinity)):
    prodnptile[v][prodnptile[v] == "0"] = "nan"
    prodnptile[v][prodnptile[v] == "1"] = salinity[v]
    prodnptile[v][prodnptile[v] == "2"] = temperature_processed[v]
    prodnptile[v][prodnptile[v] == "3"] = phosphate_processed[v]
    prodnptile[v][prodnptile[v] == "4"] = nitrate_processed[v]
    prodnptile[v][prodnptile[v] == "5"] = silicate_processed[v]
    prodnptile[v][prodnptile[v] == "6"] = oxygen_processed[v]
    product_processed.append(prodnptile)
```

Some more minor points that would improve things:

- Variables are converted between dicts and pandas DataFrames, and lists and numpy arrays, often without any clear reason. Both for code clarity and computational speed, numerical data should be kept as numpy arrays throughout, and dicts promoted to DataFrames only when essential.

Thank you for these tips. We have eliminated pandas DataFrames (replacing with dictionaries) from the package and use numpy whenever numerical data is used.

- Some packages are imported and not used (e.g., decimal).

We have eliminated these entirely.

- Some variables are defined and never used.

We have eliminated variables that were defined but not used.

- Sometimes multiple packages are used where one would be more efficient (e.g., using math and statistics for some calculations that should all be done with numpy).

Thank you for the advice. We have discontinued math and statistics packages for our calculations in the final version.

- The code could be run through a linter / auto-styler (e.g. RuK, Black) to make it more readable and help locate some of the issues noted above.

This is a good idea, and we have used ruff linter / auto-styler for this purpose.

The PR I made to the GitHub repo (see [10]) also contains fixes for some, but not all, of the points above, and I'd be happy to discuss with the authors further on how to tackle any of these issues if that might be useful.

We thank you and have made efforts to address all of these issues.

[12] Following from [10], the authors note that the Python code runs significantly slower than the MATLAB. I suspect the frequent reliance on looping calculations through lists, which is known to be very slow in Python, rather than vectorising calculations across numpy arrays, may be largely responsible for this. Operations on pandas DataFrames can also be a lot slower than the equivalent with a dict or numpy array.

We have rewritten this section and table; most of our issues stemmed from using JupyterNotebooks. However, we have implemented your above comments for even greater speed. We have also updated Sect. 3.2 and Table 3 to account for changes in calculation for the final version.

[13] For this to be really considered “available” in Python it needs at the very least to be packaged properly and installable from the GitHub repo with pip. Functions in Jupyter notebooks are not useful for integrating into other workflows. Given my comments in [1], that this manuscript is really about the code, I think that should be a bare minimum for publication.

We thank you, this package is now installable with pip, and instructions for installing this are available on the README.

[14] Uploading to PyPI and conda-forge would be very useful additional steps, although not critical for publishing this manuscript.

We agree and will also plan on doing this in the very near future.

Minor comments

[15] Figure 2: the y-axis scales have very unusual intervals, which does make it harder to interpret the figures.

We have changed the y-axis scales of Figure 2 to be much more readable, and whole-number intervals when possible.

[16] Line 261-262: presumably “these locations” refers to the “exceptions” from the previous sentence rather than the “most ocean regions”, but this is not clear.

We have altered the language to “these exceptionally different locations”

[17] The version number 1.01.01 is quite unusual. Of course it's the authors' prerogative

to use whatever system they like, but I would suggest considering switching to the very widely used semantic versioning (<https://semver.org>) to make it easier to interpret.

If we understood correctly, all version numbers for this initial release should (and have been) altered to 1.0.0.

[18] For the examples, you could consider using <https://github.com/mvdh7/glodap> to import the GLODAP dataset (this automatically downloads the files if the user doesn't have them). I included an example script in my PR (see [10]) which shows how this could be implemented.

We thank you for the information and have included this method in our examples, rather than prior downloaded datasets.