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

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).

[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.

[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!

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.

[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.

[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.

[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.

[8] From Figure 2, some of the differences are really rather large (e.g. up to 200 µmol/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.

[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.

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).

[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:

- The functions needed are in a Jupyter notebook, so they can't be imported and used in other workflows.
- There are two notebooks both with copies of these functions – there should only be one "source of truth".
- 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.
- The deprecated seawater package is used instead of its well-maintained successor gsw.
- It's virtually never necessary to explicitly use global variables in Python and best practice to avoid doing so.
- Numerical data appear to be processed into strings at some points?

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.
- Some packages are imported and not used (e.g., decimal).
- Some variables are defined and never 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).
- The code could be run through a linter / auto-styler (e.g. Ruff, Black) to make it more readable and help locate some of the issues noted above.

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.

[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.

[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.

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

Minor comments

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

[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.

[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.

[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.