Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2020-279-AC1, 2021

Geoscientific Model Development Discussions
Open Access
EGU

Interactive comment on "pyPI (v1.3): Tropical Cyclone Potential Intensity Calculations in Python" *by* Daniel M. Gilford

Daniel Gilford

daniel.gilford@rutgers.edu

Received and published: 5 February 2021

We thank the reviewer for their support for this effort and their helpful comments which have improved this work.

In response we have endeavored to clarify some opaque details of certain points which were raised by the reviewer, including the algorithm sensitivity to the minimum pressure convergence threshold, the role of dissipative heating, model run times compared with MATLAB, and the differences with other CAPE algorithms. We have also updated the equation formats, added a temperature unit check, and fixed the tone of the piece from first person to a more passive voice.

Minor comments

Printer-friendly version

Discussion paper

Line 40: As suggested, the MATLAB function has been included directly in the pyPI v1.3 code archive. We have removed this link and original line 40 to avoid confusion.

Line 68: Agreed. This line has been changed to, “Tropical cyclones arise as an indirect response to. . .”

Line 147: As highlighted by the SHARPpy introductory BAMS article (Blumberg et al. 2017), different thermodynamic profile analysis routines—and especially CAPE calculations—can vary substantially from one another. In our case, we expect this will also be true. While similar mathematically to other existing CAPE routines (e.g., SHARPY and MetPy), there are several key differences between pyPI’s CAPE algorithm and others.

First, finding the lifting condensation level is a part of the pyPI CAPE routine, found empirically as described in Eqn. 8. This method of determining the LCL contrasts with other programs. SHARPpy, for instance, begins by finding the LCL temperature with an empirical formula (this is possibly from Hart and Korotky 1991), and then determines the p_LCL from that. MetPy has an iterative procedure which convergences on the LCL pressure as a the parcel is lifted (consistent with its temperature and water vapor properties). Across the sample dataset we find that the LCL differences between pyPI and MetPy (when the parcel is lifted from the same sfc. level$\sim$1000hPa) can be up to 8 hPa. As a result, the associated CAPE calculation will be different, as the point where the ascent goes from dry to moist adiabatic can affect the amount of integrated temperature difference.

More importantly are the specific assumptions made when computing CAPE itself. MetPy calculates CAPE assuming it is proportional to the vertically integrated temperature differences. SHARPpy assumes that CAPE is proportional to the vertically integrated virtual temperature differences. pyPI assumes that CAPE is proportional to the vertically integrated density temperature differences. In the lower part of the atmosphere the density temperature and virtual temperature will be the same (as r_T $\rightarrow$ r)

and we would expect SHARPpy and pyPI to yield similar results if they are making the same ascent assumptions (below). However, without the virtual temperature correction, the errors in the MetPy CAPE calculation could be tens of percent (Doswell and Rasmussen 1994).

pyPI also has the capability of scaling between reversible and psuedoadiabatic ascent, whereas MetPy and SHARPpy (to our knowledge) use exclusively pseudoadiabiatic ascent. As noted in the manuscript's main text, this will impact the buoyancy, and make the reversible calculations in pyPI (which are typically the default in PI studies) less comparable with other packages which only look at psuedoadiabatic ascent.

We stress that it is the CAPE difference between the saturated CAPE in the eyewall and the environmental CAPE which drives the PI calculation (see also Garner 2015). We suspect that PI calculations will overall be sensitive to the definition/module of CAPE used. More work is needed to determine the sensitivity of pyPI to thermodynamic assumptions and functional forms (see response to other reviewer, where we discuss this in more detail, and following their suggestion provide a list of assumptions in Appendix B). It is our hope that the availability of the pyPI project will enable these and related studies, but a full analysis of this sensitivity is beyond the scope of this paper.

We have now added a short discussion in the beginning of the CAPE algorithm description noting that other CAPE algorithms exist (citing MetPy and SHARPpy in particular) and have differences based on particular assumptions. The distinctions in pyPI's definition of CAPE (especially the reversible ascent option and LCL definition) are also highlighted within section 3.2.

Line 241: To test the sensitivity of the calculation to this convergence threshold, we reduce its value to 0.05 hPa, as suggested, and re-run the sample analyses. Because the threshold is just an upper bound on the uncertainty of the final minimum pressure (i.e. the final p_m in Eqn. 3), we expect that the related sensitivity on V_max will likewise be small and fractional.

Across the full sample dataset we find that, between the calculations with convergence thresholds of 0.5 hPa and 0.05 hPa, the maximum difference in potential intensity (anywhere) is 0.26 m/s. Simultaneously, the computational cost of this tighter convergence threshold is an increase of ∼3 seconds per 100k profiles on our machine (a ∼29% increase in elapsed run-time). Because the PI numerical errors are much smaller than those associated with the uncertainties in other parameters/assumptions (e.g. the C_k/C_D ratio), and by contrast the associated run-time costs are relatively high, we ultimately do not implement this change in pyPI v1.3. Depending on a user's preference, needs, and available computational resources, they may want to make this trade-off between code efficiency and precision in their individual pyPI computations.

Line 281: This is an excellent question, as it useful to place the pyPI speed in context.

After updating the code with small improvements and new version dependencies, the mean elapses run time (i.e. the wall time for the code's execution) is now about 10.13 seconds on our machine (a laptop).

To find the wall time associated with each pyPI, we resample (with replacement) environmental profiles from the MERRA2 2004 dataset and then run them through the pyPI algorithm. This process is repeated 10 times to assess the variance of the run-time. As we scale the number of samples computed through the algorithm, the run time appears to linearly scale; see attached Fig. R1.

The results are somewhat noisy, possibly because of system processes or sampling; ultimately 100,000 profiles takes about 10.1 seconds to calculate PI with pyPI.

We perform the same analyses with the original Bister and Emanuel (2002) PI algorithm—which is also archived within the pyPI repository, as noted above—see attached Fig. R2.

We find that the MATLAB algorithm appears less noisy, but likewise scales linearly with the number of runs processed through the algorithm. The MATLAB algorithm

takes about 8.3 seconds to calculate PI for 100,000 profiles (about 18% less elapsed run-time than for pyPI). Note that although the MATLAB algorithm outperforms pyPI, it retains the original errors in constants, and poorly handles missing data (as discussed in the main text, section 4.2). In contrast, the conditional statements which properly handle missing data in the pyPI algorithm increases its run time.

We stress that run times will ultimately depend each user's particular implementation and system. Whereas these tests provide context for pyPI's performance relative to MATLAB for our application and system, they are not necessarily representative for each application or user.

We have added a brief discussion on the mean pyPI run time (and comparisons with the MATLAB algorithm) to the main text.

Line 289: Thank you for noting this. We have added Bryan et al. (2012) and Green and Zhang (2014) as additional references for the sensitivity of numerical simulations to the C_k/C_D ratio.

LaTeX Equations: Thank you for the suggested equation improvements. We have modified the formatting for these as follows:

Eq. 8—An exponential superscript (^) has been added and the exponential is no longer vertically compressed

Eqs. 3, 7, 12-13, 16, and throughout text—"log" has been replaced with \log

Eqs. 5, 8-10, and 14—now have parentheses which vertically extend

Eqs. 5 and 14—"e^" has been replaced with \exp as suggested. This also distinguishes the exponential from water vapor pressure introduced in Eq. 4

Lines 306-307: We agree this original statement was not specific enough. As an exercise, one can determine a rough estimate of how much dissipative heating influences the potential intensity calculation. As noted in Bister and Emanuel (1998), including

dissipative heating acts to scale the enthalpy/drag flux ratio (C_k/C_D) by a factor of T_s/T_0 (see Eqn. 2). A rough scaling for this factor during a tropical cyclone season is T_s/T_0∼300K/200K∼3/2. As V_max is proportional to the square root of this quantity, we estimate that dissipative heating scales V_max by sqrt(3/2)∼1.22, or about 22%.

We can also explore the influence of dissipative heating empirically. Using the sample input dataset in this study (from MERRA2 in 2004), we recalculate the potential intensity after turning dissipative heating off (diss_flag=0). Plotting the scatter between the valid PI calculations with and without dissipative heating against one another, we find the result in attached Fig. R3.

There is a strong approximate-linear relationship between PI calculations with and without dissipative heating. The slope of the linear fit over the full range of valid PI values is ∼1.34, which is slightly larger than the rough scaling estimate above.

Looking more closely, it appears that this relationship is not entirely linear. To look at how it changes as a function of the PI calculation, we bin each profile's PI calculations with and without dissipative heating (at each individual time and grid location) by its value of non-dissipative heating PI (every 10 m/s). Then we recalculate the slope of the linear fit between the two potential intensity samples in that bin, and determine the percent difference between them (attached Fig. R4).

The slopes of these linear fits appear to be noisy, which likely comes from the changing number of profiles/samples in each bin, and the individual SST/outflow temperature properties dominating each bin. The percent difference between these potential intensities, however, is consistently around ∼25% at intensities of tropical storm wind speeds or greater. While a full analysis of this topic is beyond the scope of this model development study, it is clear that the effects of dissipative heating are not always static—a result which is consistent with how sea conditions and wind speeds may affect the enthalpy and drag fluxes (e.g. Bao et al. 2011). It also illustrates the usefulness of pyPI for investigating these PI properties with gridded datasets.

Taken altogether, we conclude that (on average) dissipative heating will increase PI calculations by about 20-30%. Old lines 306-307 have been rewritten as: “Scaling arguments and empirical estimates suggest that dissipative heating increases PI by about 20-30% (not shown).” The above results been included in the pyPI Git repository in a Jupyter notebook (dissipative_heating_effect.ipynb).

Line 350: This illustrative analysis shows the flags for a single month (September 2004) of pyPI output. This was previously noted in the Figure 2 caption, but now we also clarify in text.

Line 376: Thank you for noting this needed specificity. We now note this is “instead of 6-hourly calculations” in the main text.

Temperature Units: The distinction of input Celsius units (for the temperature profile/SST) and output Kelvin units (for the outflow temperature) is a hold-over from the original Bister and Emanuel (2002) algorithm. As you note, we suspect the roots of this difference are that some observational input temperatures are provided in degrees Celsius (e.g. soundings). However, other sources (such as the reanalyses here) provide their raw gridded data in kelvin. In light of this, we agree that a unit check for temperatures is warranted.

To follow your suggestion, we have added a very simple set of conditional statements to the main PI algorithm which check whether the SST and the air temperature in the input profile are in degrees Celsius. They examine the input temperatures, and if any of these exceed 100 (which would be indicative of unrealistic Celsius inputs, pointing to a potential input in kelvin), then the code fails and returns missing values and the associated flag.

A long-term goal for improving pyPI (compared with this rudimentary test) is to integrate full unit support, through the use of a tool such as Pint (https://pint.readthedocs.io/en/stable/). This, however, would require an overhaul of the existing code and could also noticeably affect the run time; we therefore leave it for

future work.

Typographical and Grammatical errors

Thank you for noting the paper's original tone took away from the study. While this study is a model description paper, and hence documenting the model development process is important, we understand and agree that using these personal pronouns detracts from the overall presentation of this study.

Accordingly, throughout the study personal pronouns have been removed and those sentences have either been rewritten or "I" has been replaced with a more passive voice. We have also revised the manuscript to remove any unnecessary (too-frequent) references to our own work. In line with your comment, we hope this revision makes the paper more readable and professional, consistent with the tone of more traditional peer-reviewed manuscripts.

Line 21: Earth is now capitalized.

Line 39: Thank you for noting this was incomplete; Kerry Emanuel's full name has been added.

Line 50: Added "a"

Line 465: "approximately" now replaces the original tilde.

Line 483: These commas have been added.

Eq. 14: Subscript "_d" has been added; the equation has also been updated for readability (as noted above).

Fig. 6: While the longitudinal extent of the figure is already global we agree that we need to make the best use of the whitespace around it. We have therefore increased the overall size of the image and removed the original whitespace around the image (especially on the long/longitudinal edge).

Zenodo link: The in text reference to the Zenodo archive has been removed from the main text, and is now only included in the Code Availability section. The line (original line 123-124) discussing comments within the code, which referenced this, has been removed for readability (consistent with the tonal changes discussed above).

References: Bao, J.-W., Fairall, C. W., Michelson, S. A., & Bianco, L. (2011). Parameterizations of Sea-Spray Impact on the Air–Sea Momentum and Heat Fluxes, Monthly Weather Review, 139(12), 3781-3797.

Hart, J.A., and W. Korotky, 1991: The SHARP workstation v1.50 users guide. National Weather Service, NOAA, US. Dept. of Commerce, 30 pp. [Available from NWS Eastern Region Headquarters, 630 Johnson Ave., Bohemia, NY 11716.]

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2020-279, 2020.

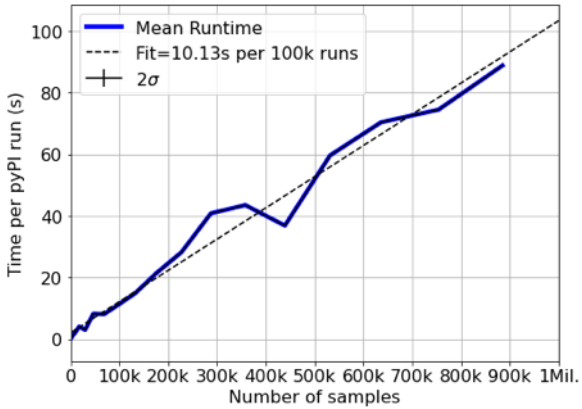

Fig. 1.

Printer-friendly version

Discussion paper

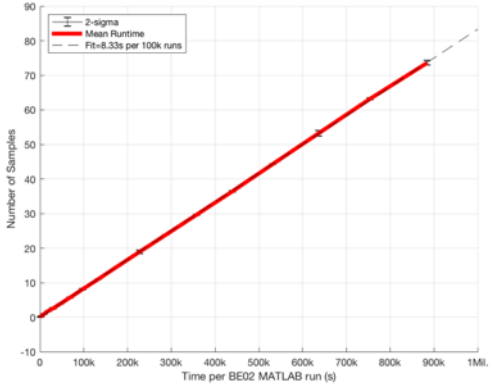

Fig. 2.

Printer-friendly version

Discussion paper

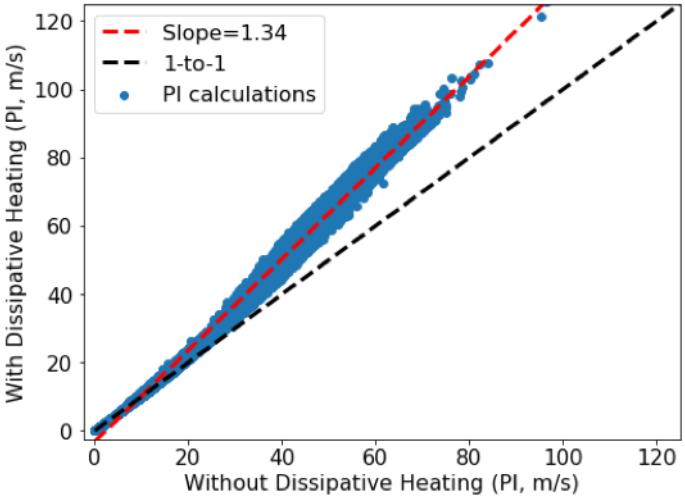

Fig. 3.

Printer-friendly version

Discussion paper

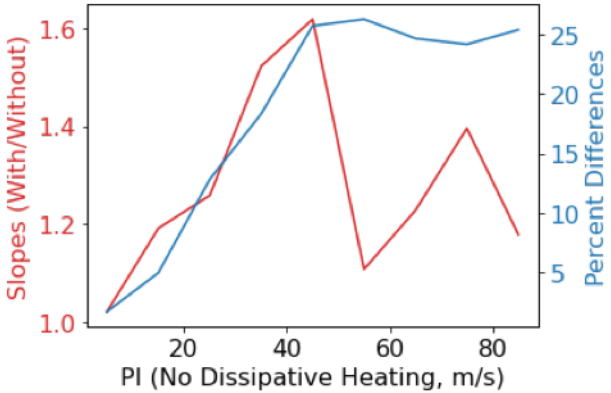

Fig. 4.

Printer-friendly version

Discussion paper