pyPI: Potential Intensity Calculations in Python User's Guide

Daniel M. Gilford, PhD
Email: daniel.gilford@rutgers.edu
Website: `http://danielgilford.com`
Twitter: @danielgilford

Updated August 14, 2020
v1.3

Contents

1 Introduction

pyPI is a set of functions, scripts, and notebooks that compute and validate tropical cyclone potential intensity (PI) calculations in Python. It is a fully documented adaptation of the Bister and Emanuel (2002) algorithm ([3], hereafter BE02) which was originally written in FORTRAN—and then converted to MATLAB—by Prof. Kerry Emanuel (MIT). The goals in developing and maintaining pyPI are to:

- supply a freely available validated Python potential intensity calculator,
- carefully document the BE02 algorithm and its Python implementation, and
- demonstrate and encourage the use of potential intensity theory in tropical cyclone analysis.

This user's guide provides a full overview of the pyPI project, including (1) how to get started as a new user, (2) a description of the codebase, (3) a validation of pyPI against the existing gold-standard PI calculations module (i.e. the BE02 MATLAB code), (4) example analyses with pyPI, and (5) an adequate accounting of the physics and mathematics contained in the potential intensity code. My intent is that any atmospheric scientist with a working knowledge of Python, some appropriate climate data, and a basic understanding of tropical meteorology, should be able to read this document, clone the repository, and begin to quickly perform (and understand!) their own potential intensity calculations.

2 Getting Started

The pyPI (`tcpypi`) codebase was written and tested with Python v3.7.6. Jupyter notebooks for analyses and validation were developed with Jupyter Notebook 6.0.3. pyPI dependencies are listed in **requirements.txt**, and are:

- NumPy 1.18.1
- Numba 0.48.0
- xarray 0.15.1

To implement pyPI, install the package with the python Packaging Index from the command line:

```
> pip install tcpypi
```

Calculating potential intensity for a given set of environmental inputs requires passing them to the **pi.py** module. An example run over a sample dataset is found in **run_sample.py** (see section 3.2); new users should begin by executing this code to ensure pyPI runs locally on their machine.

```
> python run_sample.py
```

which should print to screen:

```
Beginning PI computations...
...PI computation complete and saved

Performing PI analyses...
...PI analyses complete and saved
pyPI sample run finished!
```

If this returns an error, check that the local python environment has the correct dependencies and Python version, and that the paths within the script are suitable given the local directory structure. If problems persist, please report at ∼/pyPI/issues. When **run_sample.py** has completed its execution, view the /data/ directory to ensure that the sample output files are updated. Users may also proceed to the /notebooks/ directory and run each of them with Jupyter Notebook to reproduce the repository output.

3 pyPI Codebase & Examples

3.1 Potential Intensity Module: pi.py Description

The potential intensity module, **pi.py**, is a direct port of the BE02 MATLAB code **pc_min.m**. It shares the same baseline input and output variables, with additional adjustable parameters and with slight modifications for how flags and missing data are handled (discussed more below). Section 4.1 provides a full derivation of PI, using embedded code for reference. In this section I introduce PI and describe how the calculation is handled by **pi.py** (inputs, outputs, parameters, development decisions, and flags).

From [2, 16], the maximum (near-surface) potential intensity of a tropical cyclone, V_{max}, may be approximated by:

$$(V_{max})^2 = \frac{C_k}{C_D}\frac{(T_s - T_0)}{T_0}(h_o^* - h^*) \tag{1}$$

where C_k and C_D are the enthalpy and momentum surface exchange coefficients, respectively; T_s is the sea surface temperature (SST); T_0 is the outflow temperature; h_o^* is the saturation moist static energy at the sea surface; and h^* is the saturation moist static energy of the air above the boundary layer. The ratio $\frac{C_k}{C_D}$ is defined constant.

3.1.1 Inputs & Outputs

The PI module requires environmental state variable inputs of temperature (T) and mixing ratio (r) profiles on pressure levels (p), and concurrent T_s and mean sea level pressures (p_{msl}). The algorithm is configured to output V_{max}, the tropical cyclone minimum central pressure (p_{min}), T_0, an algorithm status

flag, and the outflow temperature level (OTL). The OTL is the level of neutral buoyancy for a saturated air parcel lifted from sea level, and corresponding to the outflow temperature. Algorithm variables and parameters are shown in Table 1.

Inputs:				
Symbol	Name	pyPI variable	Units	Values
T_s	Sea surface temperature	`SSTC`	°C	—
p_{msl}	Mean sea level pressure	`MSL`	hPa	—
$T(p)$	Temperature profile	`T`	°C	—
$r(p)$	Mixing ratio profile	`R`	g/kg	—
$\frac{C_k}{C_D}$	Ratio of exchange coefficients	`CKCD=0.9`	unitless	0.17–1.05
—	Ascent process proportion	`ascent_flag=0`	fraction	0.0–1.0
—	Dissipative heating flag	`diss_flag=1`	—	0 or 1
—	Reduction of Gradient Winds	`V_reduc=0.8`	fraction	0.0–1.0
—	Upper level pressure bound	`ptop=50`	hPa	< 100
—	Missing data flag	`miss_handle=1`	—	0 or 1
Outputs:				
V_{max}	Potential intensity	`VMAX`	ms^{-1}	—
p_{min}	Minimum central pressure	`PMIN`	hPa	—
—	Algorithm status flag	`IFL`	—	0, 1, 2, or 3
T_0	Outflow temperature	`TO`	K	—
OTL	Outflow temperature level	`OTL`	hPa	—

Table 1: Input/output variables and adjustable algorithm parameters for the PI calculation module **pi.py**. Default parameter values are provided in the "pyPI variable" column. Parameters adjusted by the user *should never* be set outside the "Values" column prescriptions without physical justification and/or module modification.

To calculate PI the user should call:

```
> pi(SSTC, MSL, P, T, R, CKCD=0.9, \
        ascent_flag=0, diss_flag=1, V_reduc=0.8, ptop=50, miss_handle=1)
```

which will return:

```
> return(VMAX, PMIN, IFL, TO, LNB)
```

corresponding to the input environmental conditions. The input profiles (`T`; `R`) must be configured such that they are one-dimensional (along the `P` grid) and with the lowest index corresponding to the lowest input profile level (highest pressure).

3.1.2 Adjustable Parameters

Building on the hard-coded parameters in the BE02 MATLAB script, pyPI has six adjustable parameters in **pi.py**. The user may set the values of these in

the module call, with the caveat that each should be chosen within the defined "Values" column of Table 1. Parameters set outside these values could result in syntax errors or logical errors in the output, or could give rise to unreasonable or unphysical PI estimates.

CKCD (default=0.9)

The ratio $\frac{C_k}{C_D}$ an uncertain quantity, and its value is an ongoing area of field and theoretical research [9]. The constant ratio depends on the sea state, and scales the potential intensity linearly. Table 1 includes the 1σ range of the ratio found with energy and momentum budget methods by the 2003 Coupled Boundary Layers Air–Sea Transfer (CBLAST) field program, [1]. Several studies have set $\frac{C_k}{C_D}$=0.9 when calculating PI (e.g. [25, 26, 16]), but it has not affected their qualitative results; as a constant it cannot influence the variability of PI.

ascent_flag (default=0)

The ascent process flag determines whether the air parcels displaced in each $CAPE$ calculation (cf. section 4.1) follow reversible adiabatic ascent (ascent_flag=0) or pseudoadiabatic ascent (ascent_flag=1). In the case of reversible ascent, the full moist entropy of the buoyant parcel is conserved along its displacement following a moist adiabat. In pseudoadiabatic ascent the heat capacity of liquid water is neglected. Liquid water is assumed to fall out of the parcel as it condenses, while the parcel ascends following the pseudoadiabatic moist adiabat; for more details see [11], their section 4.7. For practical applications using pyPI, the ascent_flag may be set to any value between 0.0 and 1.0, such that ascent is any fraction intermediate to fully reversible and fully pseudoadiabatic ascent.

diss_flag (default=1)

The dissipative heating flag determines whether dissipative heating is accounted for (diss_flag=1) or ignored (diss_flag=0) in maximum potential intensity theory (see [2], their Eqn. 22). When dissipative heating is included in the PI calculation, the leading factor in the BE02 algorithm (Eqn. 4) is $(\frac{C_k}{C_D} * \frac{T_s}{T_0})$, where $\frac{T_s}{T_0} > 1$. In the absence of dissipative heating, the leading factor is $(\frac{C_k}{C_D} * 1)$ following the original findings of [10, 7]. Thus, PI is considerably lower when dissipative heating is neglected.

V_reduc (default=0.8)

Raw potential intensities are maximum gradient wind speeds. As described in [8], gradient winds calculated with the BE02 algorithm are not directly comparable with observed intensities at the near-surface without some approximation for the scaling between gradient and 10 meter winds. Following [19], a crude reduction of 20% (V_reduc=0.8) is typically applied to scale PI for comparison with near-surface winds. The percent reduction in the gradient wind in terms of V_reduc is $P_{reduc} = 100\% * (1 - \texttt{V\_reduc})$. Note that for some applications of PI, such as using it as thermodynamic parameter in climate science—e.g. incorporation into the Genesis Potential Index, [5]—V_reduc should be set to

1.0 (no reduction).

`ptop` (default=50)

The upper level pressure bound is the minimum pressure below which the input profile is ignored during PI computation. Theoretically modeled tropical cyclone outflow can often exceed the tropical tropopause on climatological timescales [16], so setting `ptop`> 100 hPa it is not advisable. Reducing the number of considered levels by increasing `ptop` may potentially increase the speed of calculations, at the risk of finding a too-low OTL and too-warm T_0. Before altering `ptop`, users should consider their particular application and the outflow levels they anticipate given their input profiles.

`miss_handle` (default=1)

The missing data flag prescribes how missing values are handled in the $CAPE$ calculation (discussed in the next section). Following the BE02 MATLAB code (`miss_handle`=0), if missing values are found in the input temperature profile[1], then the algorithm will attempt to calculate PI for all available levels above the missing values. However, the user may more conservatively choose that any missing values in the input profile will immediately set the entire PI calculation output to missing (`miss_handle`=1).

3.1.3 Output Flags & Handling Missing Data

Mirroring the output flag convention of the BE02 MATLAB code: `IFL`=1 when the **pi.py** algorithm successfully returns valid potential intensity ouputs, `IFL`=0 when the input data is improper for a PI calculation (e.g. if $T_s < 5°$C), and `IFL`=2 when the algorithm fails to converge.

The major difference between pyPI and the BE02 MATLAB code is in how the algorithm handles missing data and the (related) flag provided in the output. (Note: by convention in pyPI, missing input variables should always be assigned the NumPy "Not-a-Number"—NaN, `np.nan`—to avoid logical or syntax errors.) The BE02 MATLAB code default is that profiles may contain missing values (specifically temperatures on pressure levels), and the algorithm computes PI over the remaining valid levels.

Because missing values may sometimes be found at the surface—and the primary $CAPE$ calculation relies heavily on the assumption of lifting the parcel within the storm and environment from that level—significant errors could arise from estimating PI when ignoring buoyancy near the surface. In principle, PI should be calculated only over data points with existent sea surface temperatures and lowest profile level temperatures; but in practice missing data may arise at the lowest profile level, leading to an errant PI calculation with the BE02 MATLAB code.

pyPI addresses this challenge in three ways. First, an adjustable parameter (`miss_handle`) is implemented to allow the user to specify how pyPI handles

[1]Note that, following the BE02 MATLAB code, mixing ratios above the boundary layer are unimportant, and may be set to zero if they are missing.

the missing values. If `miss_handle`=0, the code attempts to handle missing values akin to the way that the BE02 MATLAB code did, although there still remain some differences in the outputs between pyPI and MATLAB (see below). Notably, in this case the $CAPE$ calculation proceeds as normal only as long as there are no missing values between the lowest valid (non-missing) level and the OTL; otherwise, $CAPE$ function outputs (and hence PI module outputs) are returned as missing. Second, if `miss_handle`=1, then the $CAPE$ function will automatically interpret temperature profiles with missing data as invalid, and return missing values to the PI algorithm, resulting in the PI outputs being set to missing in the return. Third and finally, a new output flag value (`IFL`=3) is introduced in pyPI which is returned when missing values in the temperature profile results in a missing output return from **pi.py** (i.e. in either of the two cases described above).

Figure 1 shows an example of the output algorithm status flags from pyPI calculations with a month of MERRA2 data, and with the default `miss_handle`=1. The figure illustrates the few global points which had at least some missing data, resulting in a missing PI return from pyPI (`IFL`=3; red grid points). In contrast, these locations have an output (but likely errant) PI from the BE02 MATLAB code. The majority of locations where missing input data results in missing output PI are near land (e.g. the Caribbean and Indo-Pacific), where missing values likely arise as an artifact of the differences between the sample data and the land-sea mask applied (section 3.2). Missing values in the sample data (which has lowest data pressure level of 1000 hPa) could also be in locations where the monthly average p_{msl} is below 1000 hPa, resulting in T(1000 hPa)=NaN.

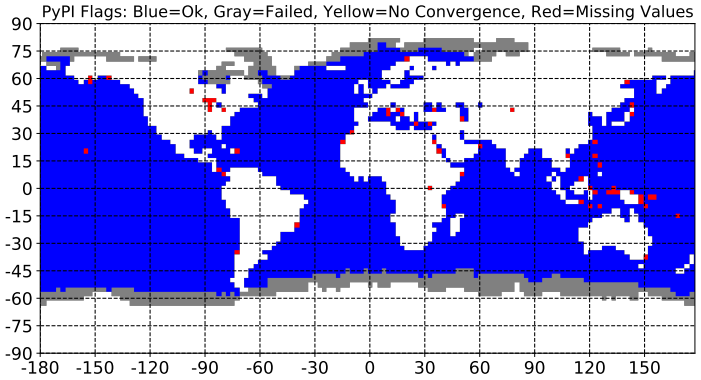

Figure 1: pyPI status flags from September 2004 potential intensity calculations when `miss_handle`=1. Blue grid cells indicate the PI algorithm converged, gray grid cells indicate the PI algorithm failed to pass a check, yellow grid cells indicate the PI algorithm did not converge, and red grid cells indicate the PI algorithm failed due to missing profile data.

In the example pyPI calculation presented in Fig. 1, there are no inputs for which the algorithm did not converge (cf. [3]). One final complication is that BE02 MATLAB code occasionally returns $V_{max} = 0$ as an output. In these cases, pyPI instead returns $V_{max} = \text{NaN}$.

Overall, pyPI converges and outputs non-missing PI values for ∼43.3% of the year-long/global sample data set (compared with the ∼42.5% of non-missing/non-zero PI values returned by the BE02 MATLAB code). In addition to how missing data is handled, output differences may also arise from slight variations in numerical computation between Python and MATLAB. All pyPI validation tests presented below (section 3.3) are computed over the spatio-temporal locations for which both algorithms have non-missing/non-zero potential intensities (and with `miss_handle`=1).

It should be noted that (by definition) profile values in the mid troposphere will have no impact on the PI calculation, so that missing values are not problematic above the boundary layer (hence mixing ratios above the boundary layer may be neglected, see section 4.1). Continuing to refine the handling of missing data in pyPI is a goal for future releases. Feedback or suggestions for improvement on this methodology are particularly welcome.

3.2 Sample Data

Potential intensity calculations require environmental state variables: temperature (T) and specific humidity (q) profiles on a pressure grid (p), sea surface temperatures (T_s), and mean sea level pressure (p_{msl}). The pyPI sample data (stored in **sample_data.nc**) are monthly means of these variables from the second Modern-Era Retrospective Analysis for Research and Applications (MERRA2, [15]) in 2004, interpolated onto a 2.5° x 2.5° global grid [16]. Note that for these example pyPI calculations the water vapor mixing ratio, r, is approximated by substituting in the reanalysis specific humidity (as $q \equiv \frac{r}{1+r} \approx r$, because $r \ll 1$).

Note that potential intensity calculations are generally linear, i.e., mean potential intensities may be estimated as a function of mean environmental variables,

$$E[V_{max}(T_s, p_{msl}, p, T, r)] \approx V_{max}(E[T_s], E[p_{msl}], E[p], E[T], E[r]) \qquad (2)$$

where $E[\,\cdot\,]$ is the expected value of a function or variable. Using monthly mean environmental conditions to compute climatological monthly means of potential intensity (and the algorithm's other output variables) generates a small bias of <1 ms^{-1} anywhere globally and <0.5 ms^{-1} in the tropics [17]. This property is convenient, because it reduces the scale of data needed to compute PI: daily or hourly data are not needed for monthly or longer (climatological) applications. Applications on shorter (e.g. operational or daily) timescales, however, should use appropriately shorter frequency inputs to the algorithm.

For consistency with [16], the sample data uses the land-sea mask from the European Centre for Medium-Range Weather Forecasts Interim (ERA-Interim,

[6]) on a 2.5° x 2.5° global grid. By definition, $V_{max} \equiv 0$ where T_s = NaN (i.e. over land); in some cases (e.g. if skin temperatures—valid over land—are used in lieu of sea surface temperatures) PI may be mistakenly calculated over land with **pi.py**. In these cases, users should assign all PI algorithm outputs over land to the missing value in post-processing. As an alternative, in this pyPI example input variables over land are set to missing in a pre-processing step (see **reference_calculations.m**). Note that the mismatch between using the ERA-I land-sea mask and MERRA2 data in this example results in a set of minor output artifacts caused by missing (MERRA2 land grid points) input data over ERA-I defined ocean grid points (and providing an illustration of the missing data flag developed in pyPI).

3.3 pyPI Validation

Accompanying the environmental conditions in **sample_data.nc** are outputs from the BE02 MATLAB code, **pc_min.m**, written by Kerry Emanuel (see ftp://texmex.mit.edu/pub/emanuel/TCMAX) and revised by Daniel Gilford for climatological research applications (see [16], [17], and MATLAB script **reference_calculations.m**).

Potential intensities calculated over September 2004 with pyPI and the BE02 MATLAB code are compared in Figure 2 over the globe; their difference is computed and plotted in Figure 3. There is excellent agreement between the two algorithms, with the absolute maximum difference anywhere across the sample calculations being 0.22 ms^{-1}. Potential intensities calculated with the Python algorithm exhibit a slightly negative bias relative to the MATLAB calculations. But relative to the uncertainties in the PI calculation, such as the ratio of surface exchange coefficients, these differences are negligible.

To further show this agreement, Figure 4 shows the scatter between all potential intensity values calculated with the two algorithms over the sample data, plotted against a 1-to-1 line (values lying on this line exhibit perfect agreement). The major differences between pyPI and BE02 arise from a correction of the ϵ constant in **pi.py**. This improvement leads to small differences between the two function. Still, R-squared of this comparison is $R^2 \approx 1.0$ to seven significant digits, such that they are nearly identical. All other output variables (cf. Table 1) from the two algorithms have similarly strong levels of agreement. I conclude that the PI calculations made with **pi.py** are adequately validated against the BE02 MATLAB code, and that pyPI is sufficiently accurate for use in research applications.

3.4 Sample Analyses: PI Annual Means

Global 2004 annual mean sea surface temperatures, and pyPI calculated potential intensities, outflow temperatures, and outflow temperature levels are shown in Figure 5. The familiar pattern of warm SSTs in the tropics corresponds with high V_{max} values, suggesting that on an annual timescale PI is strongly influenced by T_s. These warm and high-PI reigons are also accompanied by

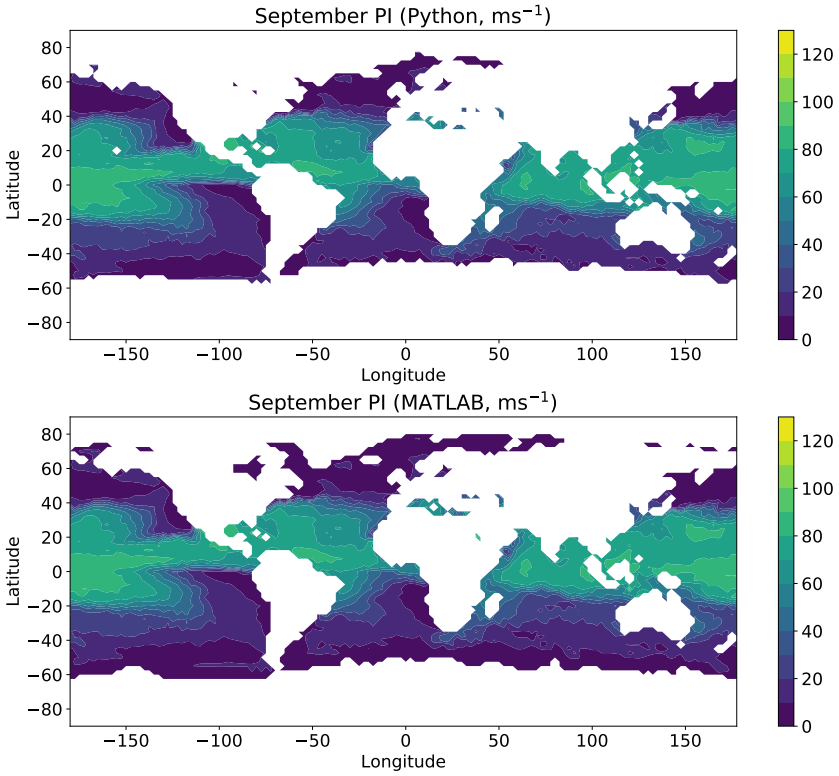

Figure 2: September 2004 mean potential intensities (ms^{-1}) calculated with pyPI (upper) and the BE02 MATLAB code (lower).

outflow temperature levels with annual pressures below 100 hPa, in the tropical tropopause region (e.g. [13]). Near the tropical tropopause, annual mean outflow temperatures are remarkably cold, around 200 K. On average, the coldest outflow temperatures are found in the Western North Pacific basin, where consistent deep convection and stratospheric circulation act to keep tropopause temperatures very cold and highly variable (e.g. [22]).

3.5 Sample Analyses: PI Seasonal Cycles

A slightly more sophisticated application of pyPI is the calculation of potential intensity seasonal cycles. Reproducing the methodology of [16] for only a single year, Figure 6 shows the seasonal cycles of sea surface temperatures, and

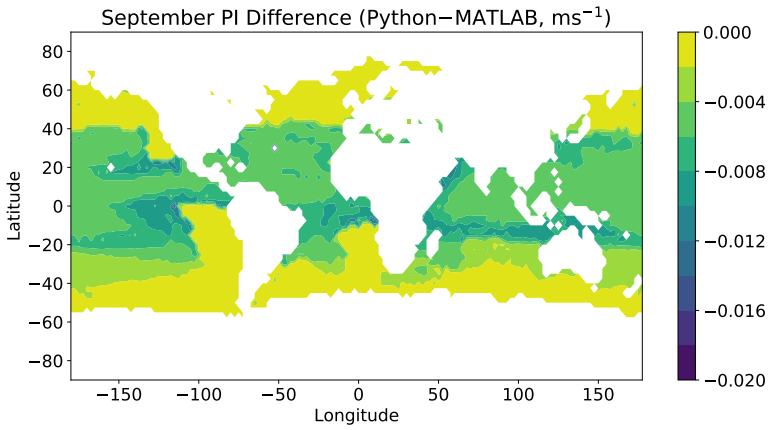

Figure 3: September 2004 mean potential intensity differences (ms^{-1}) between those calculated with pyPI minus those calculated with the BE02 MATLAB code.

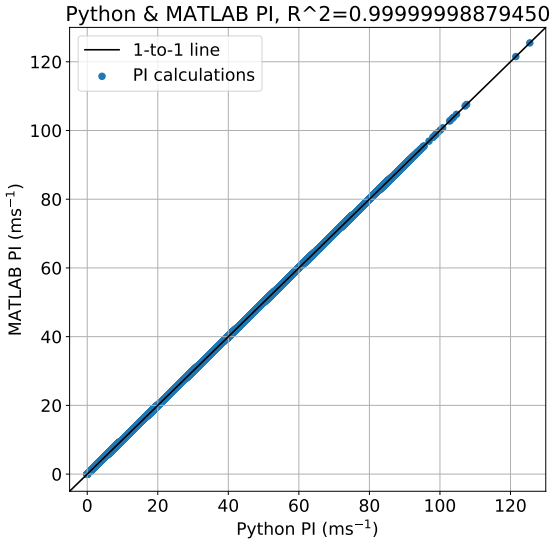

Figure 4: 2004 mean potential intensities (ms^{-1}, blue dots) calculated with pyPI (horizontal axis) and the BE02 MATLAB code (vertical axis). The black curve is the 1-to-1 line.

pyPI calculated outflow temperatures, outflow temperature levels, and poten-

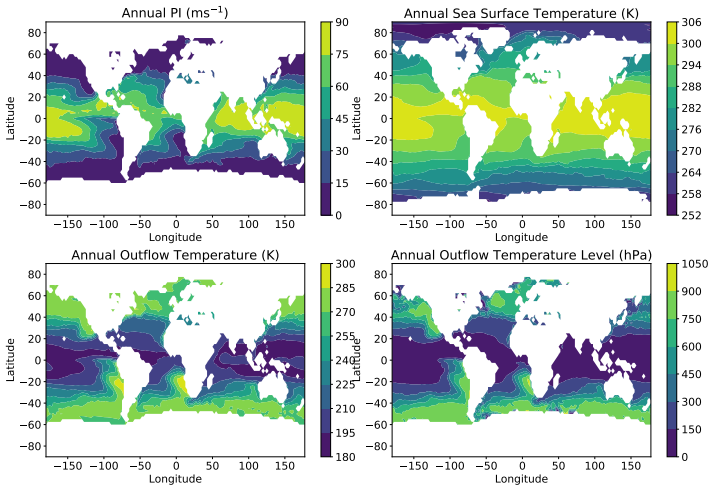

Figure 5: 2004 annual mean potential intensities (ms^{-1}, top left), sea surface temperatures (K, top right), outflow temperatures (bottom left, K), and outflow temperature levels (bottom right, hPa) calculated with pyPI .

tial intensities in 2004 averaged over tropical cyclone main development regions (defined in [16], their Table 1).

The seasonal cycles of PI are known to be quite robust year-over-year and exhibit clear differences between regions. Consistent with the findings of [16], the Western North Pacific has a nearly flat seasonal cycle of PI, while the other basins are more intraseasonally variable. While the muted sea surface temperatures certainly play an important role in this damped cycle, the outflow temperature pattern is typical of the cold-point tropopause seasonal cycle (e.g. [27, 21])—which the OTLs are reaching—which acts to damp the seasonal cycle further by decreasing PI in the boreal summer and increasing PI in the winter. As a result, tropical cyclones in the Western North Pacific have higher speed limits during the winter months (described in detail below). Consistent with this finding, historical observed typhoons show intense wind speeds during the winter and spring months [17]. For example, Typhoon Sudal reached category 4 strength, $\sim$67 ms^{-1}, in early April 2004 when the monthly average PI was about 75 ms^{-1}.

The seasonal cycles of each basin illustrate the complex relationship between sea surface temperatures, OTLs, and outflow temperatures. Figure 7 diagrams this relationship in more detail, showing how one assumption in the pyPI algorithm impacts the output PI values. pyPI (and the BE02 MATLAB code it is based on) assumes during a PI calculation that the outflow temperature and its level are derived by finding the level of neutral buoyancy assuming a sat-

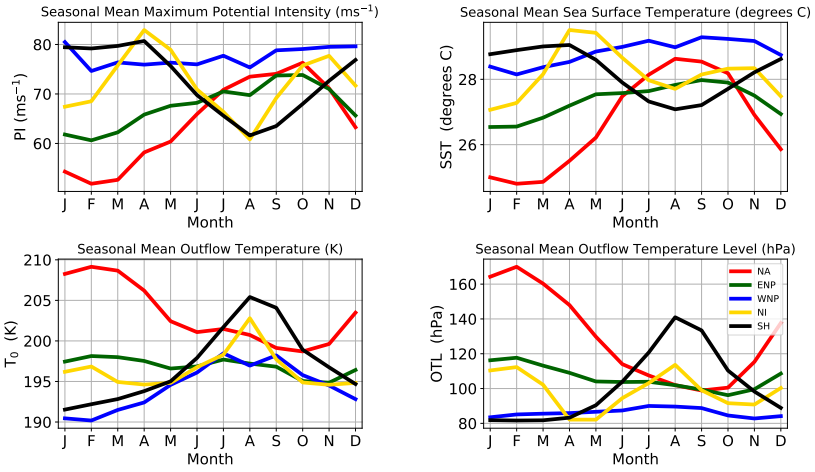

Figure 6: 2004 seasonal cycles of potential intensity (ms^{-1}, top left), sea surface temperature (K, top right), outflow temperature (bottom left, K), and outflow temperature level (bottom right, hPa) calculated with pyPI and averaged over the main development regions defined in [16] (their Table 1): the North Atlantic (red), Eastern North Pacific (green), Western North Pacific (blue), North Indian (yellow), and Southern Hemisphere (black).

urated parcel lifted from the sea surface with temperature, T_s. This implies that, following a moist adiabat, the level of the neutral buoyancy is a function of only T_s and the environmental temperature profile, T. Given a fixed temperature profile, an increase in T_s (e.g. from SST1 $\rightarrow$ SST2 in Fig. 7) requires that the associated OTL will be found at a higher altitude (OTL1 $\rightarrow$ OTL2 in Fig. 7) and the associated outflow temperature will likewise change. As the atmosphere's stratification increases into the lower stratosphere, increases in T_s will be less effective at changing the OTL and its temperature, with the effect nearly saturating when the outflow reaches the cold-point tropopause (e.g. 100 hPa in Fig. 7). At this point, T_s variability is almost completely decoupled from T_0 variability. Instead these T_0 values become influenced by tropopause region variability (e.g. [12, 20, 25, 26, 16]) which is controlled by radiation, dynamics, and deep convection (e.g. [13, 21]).

These properties are borne out in the example 2004 seasonal cycles computed by pyPI in Fig. 6. In the North Atlantic basin SST and OTL seasonal cycles are inversely proportional: colder SSTs have higher pressure OTLs and warmer outflow temperatures found in the upper troposphere (where $\frac{dT}{dz} < 0$) in all months except August-September. In these late summer months the OTL reaches near the cold-point tropopause, and the outflow temperature slightly increases, following the seasonal cycle of warmer tropopause temperatures (e.g.

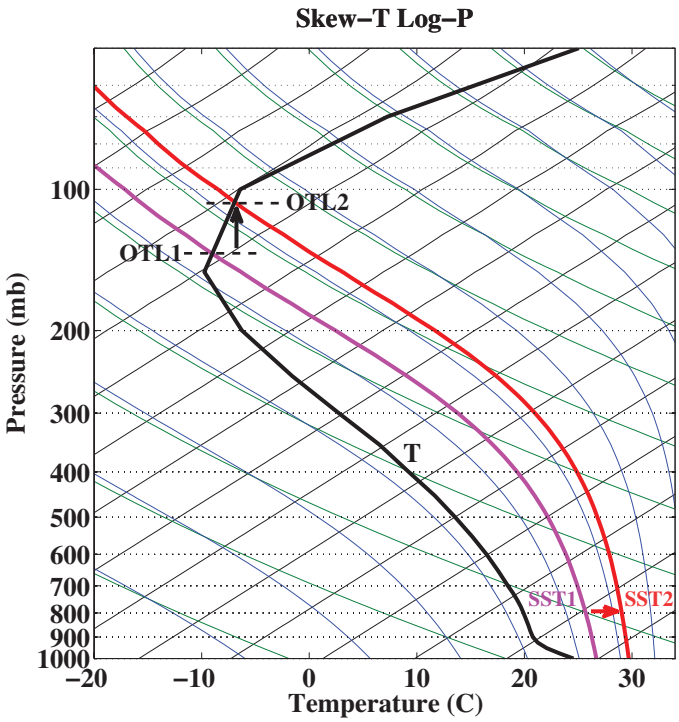

Figure 7: Skew-T log-P thermodynamic diagram with isotherms (thin black curves), dry adiabats (green curves), and moist adiabats (blue curves). The bold black line is a mean environmental temperature profile from the North Atlantic region ([16], their Table 1), the magenta curve is the moist adiabat associated with a mean North Atlantic SST, and the red curve is the moist adiabat associated with an SST 3 degrees warmer than the mean.

[27]). A contrasting pattern is seen in the Western North Pacific, where the OTLs have almost no seasonal cycle: in this basin the calculated outflow *always* reaches the lowermost stratosphere (OTL $\leq$ 90 hPa). Accordingly, the outflow temperature seasonal cycle perennially follows the seasonality of lowermost stratospheric temperatures, which minimize in the boreal winter and maximize in the boreal summer ([27]). Comparing with Eqn. 1, this T_0 seasonality leads to relatively *increased* PI values in the boreal winter and relatively *decreased* PI values in the boreal summer. Overall, the PI seasonal cycle in the Western North Pacific is damped over the year—this pattern is also observed in real-world tropical cyclone intensities (cf. [17]).

3.6 Sample Analyses: Decomposition

The relative contributions to potential intensity may be mathematically derived by decomposing Eqn. 1. Taking the natural logarithm of both sides:

$$2 \times log(V_{max}) = log(\frac{C_k}{C_D}) + log(\frac{T_s - T_0}{T_0}) + log(h_o^* - h^*) \tag{3}$$

then PI variability is related to variability in either tropical cyclone efficiency ($\frac{T_s - T_0}{T_0}$) or thermodynamic disequilibrium ($h_o^* - h^*$); recall that $\frac{C_k}{C_D}$ is taken as a constant. As an example following [16], Eqn. 3 is applied to pyPI calculated 2004 seasonal cycles of potential intensity (from Fig. 6). pyPI calculates PI directly, and efficiency may be directly computed from input T_s and output T_0; following [26] the disequilibrium term is taken as a residual from Eqn. 3.

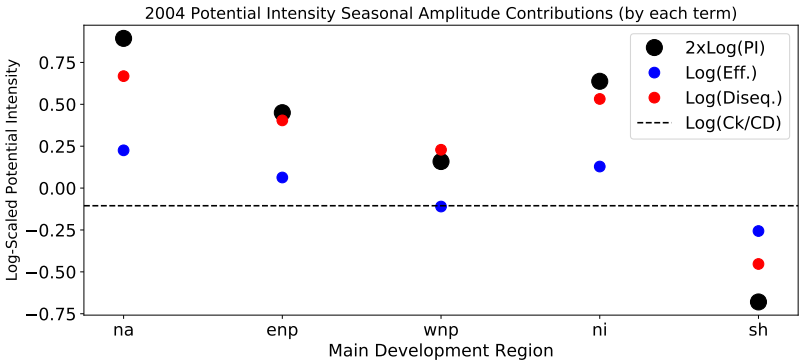

Figure 8: Seasonal amplitudes of each PI decomposition term (Eqn. 3) in 2004 and each main development region, calculated with pyPI. Compare with [16], their Table 2. By convention, negative amplitudes indicate the associated seasonal cycle peaks in the boreal winter. For reference, the dashed black line indicates the magnitude and sign of the seasonally invariant $\log(\frac{C_k}{C_D})$.

After finding each term in Eqn. 3 over each basin and seasonal cycle, the amplitude (equivalent to the annual range on a monthly timescale) of each seasonal cycle is plotted in Figure 8. By convention ([16, 17]), negative amplitudes indicate the ∼sinusoidal seasonal cycle reaches its maximum in the boreal winter and minimum in the boreal summer.

In all basins, the disequilibrium term drives the largest portion of the seasonal amplitude. This is consistent with seasonal cycles in SSTs which dominate the disequilibrium variance (cf. SST seasonality in Fig. 6). The efficiency term is smaller, and in each basin it follows the same cycle as thermodynamic disequilibrium, with the exception of the Western North Pacific (where the efficiency seasonal cycle maximizes in the boreal winter and minimizes in the boreal summer). Referring to the discussion in the previous section 3.5, this opposite signed seasonality between disequilibrium and efficiency in the Western North Pacific

is directly related to the influential seasonality of the near-tropopause outflow temperatures found with the pyPI calculations (see full discussions in [16, 17]). Notably, the Southern Hemisphere shares this outflow temperature seasonality, which actually amplifies the efficiency seasonal cycle through both SST *and* outflow temperature intraseasonal variability. In all other basins, outflow temperature seasonality is offset by the sea surface temperature seasonality, which acts to mute the efficiency term and further lead to disequilibrium dominating the seasonal cycle. Fig. 8 illustrates that, ultimately, the roles of environmental conditions in PI seasonality are basin dependant.

These results show a few short examples of how pyPI may be used to study tropical cyclone intensities, and likewise serve to illustrate the ability of pyPI to reproduce work previously computed with the BE02 MATLAB code.

4 Potential Intensity Computation

4.1 pyPI Algorithm

Here I show the computation of potential intensity and its components (primarily the convective available potential energy, $CAPE$) in the pyPI code. Constants used in the formulation and computation are recorded in Table 2.

Symbol	Constant Name	pyPI variable	Value/Units
c_{pd}	Specific heat of dry air	`CPD`	1005.7 J/kg.K
c_{pv}	Specific heat of water vapor	`CPV`	1870 J/kg.K
c_ℓ	Specific heat of liquid water	`CL`	2500 J/kg.K
R_v	Gas constant of water vapor	`RV`	461.5 J/kg.K
R_d	Gas constant of dry air	`RD`	287.04 J/kg.K
ϵ	Ratio of gas constants	`EPS`	$\frac{R_d}{R_v} = 0.6219...$
L_{v0}	Latent heat of vaporization at 0°C	`ALV0`	2.501e6 J/kg

Table 2: Constants used in PI calculation.

4.1.1 PI Formulation

Following [3] and based on the formulation of [7], idealizing a tropical cyclone as a Carnot heat engine, and assuming:

1. the work done against friction by the outflow is ignored,
2. when the storm intensity reaches its maximum, the anticyclone at the top of the storm is fully developed, and
3. the gradient wind may be approximated by cyclostrophic wind at the RMW,

then the Carnot cycle formulation yields an expression for the maximum potential intensity (which roughly scales with the approximated PI expression, Eqn. 1, [16, 26]):

$$(V_{max})^2 = \frac{T_s}{T_0}\frac{C_k}{C_D}(CAPE^* - CAPE_e)|_{RMW} \tag{4}$$

where $CAPE^*$ is the convective available potential energy of saturated air lifted from sea level to the OTL referencing the environmental profile, and $CAPE_e$ is the convective available potential energy of the environment. Because the final term is evaluated at the RMW and $CAPE^*$ is pressure dependant, an expression for the surface pressure at the RMW is needed. Following [3] (cf. also [14], their Eqn. 6), the minimum pressure of the tropical cyclone at the RMW, p_m is found with[2]:

$$R_d T_v log(\frac{p_{msl}}{p_m}) = \frac{1}{2}(V_{max})^2 + CAPE|_{RMW}, \tag{5}$$

where T_v is the surface environmental virtual temperature, and $CAPE|_{RMW}$ is the environmental convective available potential energy evaluated at the RMW. Because the boundary layer water vapor mixing ratio is higher in the tropical cyclone eyewall than the storm's outer region (assuming a constant relative humidity in the boundary layer across the storm's radius), $CAPE|_{RMW}$ is slightly larger than $CAPE_e$ (discussed more below).

The pressure dependence of $CAPE$ requires solving Eqn. 4 and 5 with numerical iteration, which is performed in pyPI. In the following section I describe the algorithm and each component of the PI numerical computation.

4.1.2 pyPI Computation

The algorithm to compute potential intensity is:

Potential Intensity Algorithm

input: T_s, p_{msl}, p, $T(p)$, $r(p)$
1: Run checks to ensure inputs are appropriate
2: Calculate $CAPE_e$
repeat
    3: Calculate $CAPE|_{RMW}$
    4: Calculate $CAPE^*$, OTL, T_0
    5: Estimate p_m at this i^{th} iteration
until convergence... objective $\Delta_{i,i-1}p_m < 0.5$ hPa
6: Calculate final p_{min} value
7: Calculate V_{max}
return: V_{max}, p_{min}, OTL, T_0, flag.

[2] As noted in [14], Eqn. 4 in [3] mistakenly replaces R_d with c_p. **pc_min.m**, however, includes the correct factor of R_d, which is carried over into pyPI.

In addition to this detailed accounting of the potential intensity algorithm, each line of **pi.py** has been commented for quick reference. I now review the algorithm, stepping through line-by-line in pyPI (and ignoring unit conversions/`if` statements/etc. when unnecessary):

We begin by checking to ensure that the input profiles are appropriate for the PI calculation. If not, missing values are returned by the algorithm. Because mixing ratios outside the boundary layer are unimportant, missing ratios are replaced with 0 g/g (see BE02 MATLAB code).

Next, we calculate the surface saturated water vapor pressure, which is used within the loop to estimate the saturated mixing ratio. The empirical expression for saturated water vapor pressure (in hPa) is given empirically with T in °C by [4] (their Eqn. 10):

$$e_s(T) = 6.112 * exp(\frac{17.67 * T}{T + 243.5}). \tag{6}$$

At the surface, the temperature is given by the sea surface temperature, so that $e_s(T_s)$ is found with:

```
> ES0=6.112*np.exp(17.67*SSTC/(243.5+SSTC))
```

After defining constants (Table 2), we compute $CAPE_e$, which is needed to compute PI (Eqn. 4):

```
> TP=T[NK]
> RP=R[NK]
> PP=P[NK]
> result = cape(TP,RP,PP,T,R,P,ascent_flag,ptop,miss_handle)
> CAPEA = result[0]
```

where `NK=0` (the parcel is lifted from the lowermost input level in `P`). Below, I describe the CAPE function which plays the central role in **pi.py**, before returning to the base function to continue describing the PI algorithm.

* * *

The $CAPE$ calculation is made in an accompanying function within **pi.py**. It assesses $CAPE$ as the total positive and negative areas of buoyancy energy (e.g. [11], their Eqn. 6.3.6, discussed more below). First, we check the user input missing flag (`miss_handle`), and return missing $CAPE$ if the check is failed, or if the input parcel is unsuitable for a $CAPE$ calculation.

After defining constants within the CAPE function (Table 2), we find the parcel moisture characteristics. By the Ideal Gas Law, we know ([4]. their Eqn. 16):

$$e = \frac{r * p}{\epsilon + r} \tag{7}$$

and after defining fractional relative humidity, $RH \equiv \frac{e}{e_s} \leq 1.0$, we may find the parcel's saturated vapor pressure, partial vapor pressure, and relative humidity:

```
> ESP=6.112*np.exp(17.67*TPC/(243.5+TPC))
> EVP=RP*PP/(EPS+RP)
> RH=EVP/ESP
> RH=min([RH,1.0])
```

Assuming the temperature dependence of specific heats is negligible over the range of temperatures in the tropical atmosphere, and integrating Kirchoff's equation (e.g. [11], Eqn. 4.4.3-4.4.4) then the temperature dependence of the latent heat of vaporization is:

$$L_v = L_{v0} + (c_{pv} - c_\ell) * T \tag{8}$$

where T is in °C, so that we find L_v:

```
> ALV=ALV0+CPVMCL*TPC
```

Finally, we are equipped to calculate the parcel's reversible total specific entropy (per unit mass of dry air), s, given by [11] (their Eqn. 4.5.9):

$$s = (c_{pd} + r_T c_\ell) log(T) - R_d log(p) + \frac{L_v r}{T} - r R_v log(RH) \tag{9}$$

where r_T is the total water content mixing ratio, which is identical to the parcel r at the surface. In the **pi.py** CAPE function, we find s of the parcel at the surface:

```
> S=(CPD+RP*CL)*np.log(TP)-RD*np.log(PP-EVP)+ \
  ALV*RP/TP-RP*RV*np.log(RH)
```

Next, the lifting condensation level (LCL) of the parcel must be found to partition the bouyancy calculation between saturated and non-saturated regions of the profile. The pressure level of the LCL is found empirically with[3]:

$$p_{LCL} = p * RH^{(\frac{T}{A - B*RH - T})} \tag{10}$$

where $A = 1669$ and $B = 122$. In the BEO2 MATLAB code p_{LCL} for the parcel is estimated as:

```
> CHI=TP/(1669.0-122.0*RH-TP)
> PLCL=PP*(RH**CHI)
```

Here note that saturated parcels will have $p_{LCL} = p$, so that the saturated parcel's LCL is its original pressure level. Likewise, parcels at levels below the LCL are (by definition) not yet saturated. Starting here, the CAPE function begins the "updraft loop", where the positive and negative buoyancy of the

[3] This appears to be derived empirically from [4]; it was developed for [11] (Kerry Emanuel, *personal communication*). Modern calculations of p_{LCL} are made following the exact expressions of [23]. A goal for a future release is to incorporate the modern LCL formulation into pyPI.

parcel is calculated for every j^{th} level below the upper boundary on pressure (`ptop`).

Starting with calculations at levels *below* the LCL (i.e. $p_j > p_{LCL}$), at each j^{th} level we find the unsaturated parcel temperature by following a dry adiabat with the same temperature as the surface parcel (i.e. by Poisson's equation):

$$T_k = T * (\frac{p_k}{p})^{\frac{R_d}{c_{pd}}} \tag{11}$$

Because $CAPE$ is proportional to the positive and negative areas enclosed by the environmental and lifted parcel density temperatures ($T_{\rho,e}$ and T_ρ, respectively), we seek to calculate the density temperature ([11], their Eqn. 4.3.6):

$$T_\rho = T * (\frac{1 + r/\epsilon}{1 + r_T}) \tag{12}$$

where the net water mixing ratio (r_T) is the same as the parcel water mixing ratio at the surface, and in the environment below the LCL before condensation has occurred (i.e. $r_T = r$ and $r_{T,j} = r_{j<LCL}$). The code accordingly calculates the density temperatures of the environment, the parcel, and their differences (which will be used to determine the buoyant areas):

```
> TG=TP*(P[j]/PP)**(RD/CPD)
> RG=RP
> TLVR=TG*(1.+RG/EPS)/(1.+RG)
> TVENV=T[j]*(1.+R[j]/EPS)/(1+R[j])
> TVRDIF[j,]=TLVR-TVENV
```

Next the code calculates the density temperature differences for all levels *above* the LCL (i.e. $p_j < p_{LCL}$). Because the parcel is saturated above the LCL, its moisture characteristics and temperature must be found iteratively at each j^{th} level. Rearranging Eqn. 7 to solve for r, the code sets initial conditions for the iteration:

```
> TGNEW=T[j]
> TJC=T[j]-273.15
> ES=6.112*np.exp(17.67*TJC/(243.5+TJC))
> RG=EPS*ES/(P[j]-ES)
```

Until the numerical iteration converges (with objective for the parcel temperature: $\Delta T < 0.001$ K), we solve for parcel moisture characteristics which conserve the parcel's reversible entropy s (Eqn. 9) following a moist adiabat; finding these permits an estimation of the level's density temperature differences. At the beginning of the loop, we set the variables equal to the previous iteration's findings:

```
> TG=TGNEW
> TC=TG-273.15
> ENEW=6.112*np.exp(17.67*TC/(243.5+TC))
> RG=EPS*ENEW/(P[j]-ENEW)
```

then we step forward updating the parcel's temperature (and the dependant L_v and water vapor mixing ratios) assuming the parcel's entropy, S, is conserved following saturated reversible adiabatic displacement. Following Newton's method (i.e. $T_{n+1} = T_n + \frac{s(T_n)}{ds(T_n)/dT}$, [24]), when the difference between S and this iteration's entropy, SG, scaled by the rate of change of entropy with temperature, SL, is small (i.e. $\frac{\mathtt{S}-\mathtt{SG}}{\mathtt{SL}} < \mathtt{AP} * 0.001$), then the algorithm will converge (such that we find the parcel temperature at this level, T_j):

```
> ALV=ALV0+CPVMCL*TC
> SL=(CPD+RP*CL+ALV*ALV*RG/(RV*TG*TG))/TG
> EM=RG*P[j]/(EPS+RG)
> SG=(CPD+RP*CL)*np.log(TG)-RD*np.log(P[j]-EM)+ALV*RG/TG
> TGNEW=TG+AP*(S-SG)/SL
```

Note that at saturation the final term in Eqn. 9 vanishes. Furthermore, the step size (AP) changes dynamically depending upon the number of iterations (NC) that have taken place. If the number of iterations exceeds 500 (an arbitrary chosen excessive number of iterations), or if the water vapor pressure nears the level pressure, then the algorithm fails to converge and returns zero $CAPE$.

When the algorithm has converged for a level, the final parcel mixing ratio is set depending on the ascent type chosen by the user (section 3.1.2). For pseudoadiabatic ascent (ascent_flag= 1), liquid water condensed in the parcel during its ascent is assumed to drop out during ascent, such that the heat capacity of liquid water is neglected (i.e. $r_T = r_j$). For reversible ascent (ascent_flag= 0) this water (and its heat capacity) is retained following the parcel (i.e. $r_T = r$).

Note that the density temperature difference (and hence, a parcel's buoyancy) with height is not strictly higher under either assumption. Parcels lifted reversibly always are warmer than those lifted psuedoadiabatically, but the weight of the carried condensate also means these parcels are more dense until they reach the upper troposphere ([11], their Table 4.2). Accordingly, [16] found that psuedoadiabatic (typically more buoyant) PI calculations generally have higher altitude OTLs than reversible (typically less buoyant) PI calculations on monthly timescales.

The parcel's mixing ratio is found following the flag set by the user:

```
> RMEAN=ascent_flag*RG+(1-ascent_flag)*RP
```

which is used to compute the density temperature for the parcel (Eqn. 12). Similar to the approach for levels below the LCL, we find also find the environmental density temperature at this level, and calculate the density temperature differences:

```
> TLVR=TG*(1.+RG/EPS)/(1.+RMEAN)
> TENV=T[j]*(1.+R[j]/EPS)/(1.+R[j])
> TVRDIF[j,]=TLVR-TENV
```

Following the full calculation of $T_\rho - T_{\rho,e}$ at each level, we are equipped to calculate the parcel's convective available potential energy. Parcel $CAPE$ is

given by the vertically integrated buoyant energy between the level from which the parcel is initially lifted ($j = 0$) and the level of neutral buoyancy (LNB; this is sometimes referred to as the "equilibrium level"; $j = LNB$). Following [11], their Eqn. 6.3.6:

$$CAPE = PA - NA \tag{13}$$

where

$$NA \equiv -\int_{p_{j=LFC}}^{p_{j=0}} R_d(T_\rho - T_{\rho,e}) dlog(p) \tag{14}$$

$$PA \equiv +\int_{p_{j=LNB}}^{p_{j=LFC}} R_d(T_\rho - T_{\rho,e}) dlog(p) \tag{15}$$

Negative areas (NA) are vertical regions of negative buoyancy which inhibit spontaneous convection in the profile; positive areas (PA) are vertical regions of positive buoyancy which cause the parcel to rise assuming an initial upward displacement[4]. By definition, the level of free convection (LFC) separates regions that are negatively buoyant (below) from regions that are positively buoyant (above). When $(j = LFC) > (j = LCL)$, then regions of the profile above the LCL and below the LFC will still be negatively buoyant.

The CAPE function assesses Eqns. 13-15 in five steps:

First, we find the maximum level of positive buoyancy (`INB`), i.e. the highest altitude j^{th} level where $T_\rho - T_{\rho,e} > 0$:

```
> INB=0
> for j in range(nlvl-1, jmin, -1):
>     if (TVRDIF[j] > 0):
>         INB=max([INB,j])
```

If the highest level remains at $j = 0$, then there are no positively buoyant levels and the function returns zero $CAPE$.

Second, noting that $dp/(p_{mean,k}) \approx dlog(p)_k$ at each layer, k, between two j levels—where the average pressure of each k^{th} layer is $p_{mean,k} = \frac{1}{2}(p_j + p_{j+1})$—we find the positive and negative areas between the second-highest altitude level (i.e. $j = 1$) and the the maximum level of positive buoyancy (taking care to preserve the sign convention of Eqns. 14-15):

```
> for j in range(jmin+1, INB+1, 1):
    > PFAC=RD*(TVRDIF[j]+TVRDIF[j-1])*(P[j-1]-P[j])/(P[j]+P[j-1])
    > PA=PA+max([PFAC,0.0])
    > NA=NA-min([PFAC,0.0])
```

Third, we find the residual negative area (if $j = LFC > 0$) or positive area (if $j = LFC = 0$) of the mean layer composed of the parcel and the lowest level:

[4]Note that $CAPE$ is not defined for parcels without positive areas.

```
> PMA=(PP+P[jmin])
> PFAC=RD*(PP-P[jmin])/PMA
> PA=PA+PFAC*max([TVRDIF[jmin],0.0])
> NA=NA-PFAC*min([TVRDIF[jmin],0.0])
```

Fourth, we find level of neutral buoyancy, along with the residual positive area of the mean layer between the the maximum level of positive buoyancy and the LNB. We also find the temperature at the LNB:

```
> TOB=T[INB]
> LNB=P[INB]
> if (INB < nlvl-1):
    > PINB=(P[INB+1]*TVRDIF[INB]-P[INB]*TVRDIF[INB+1]) \
        /(TVRDIF[INB]-TVRDIF[INB+1])
    > LNB=PINB
    > PAT=RD*TVRDIF[INB]*(P[INB]-PINB)/(P[INB]+PINB)
    > TOB=(T[INB]*(PINB-P[INB+1])+T[INB+1]*(P[INB]-PINB)) \
        /(P[INB]-P[INB+1])
```

If the `INB` is found at the highest valid level (i.e. constrained by `ptop`), then the LNB and its temperature are set at that level.

Finally, the negative and positive areas are added together (along with the residual) following Eqn. 13:

```
> CAPED=PA+PAT-NA
```

After this last step in the CAPE function, we set the flag to indicate successful computation (`IFLAG=1`) and return $CAPE$ (`CAPED`), the temperature at the LNB (`TOB`), the LNB itself (`LNB`), and the flag to the PI function.

* * *

Returning to the PI function, we have found the environment's $CAPE_e$. Next, we begin an iterative loop to numerically solve Eqns. 4 and 5 (with objective for the minimum pressure at the RMW: $\Delta_{i,i-1}p_m < 0.5$ hPa). After setting the initial conditions, begin the iteration by calculating environmental convective available potential energy at the radius of maximum winds, $CAPE|_{RMW}$:

```
> TP=T[NK]
> PP=min([PM,1000.0])
> RP=EPS*R[NK]*MSL/(PP*(EPS+R[NK])-R[NK]*MSL)
> result = cape(TP,RP,PP,T,R,P,ascent_flag,ptop,miss_handle)
> CAPEM = result[0]
```

where Eqn. 7 has been rearranged to solve for the environmental parcel mixing ratio, which has pressure dependence (r increases slightly as $p \to p_m$ approaching the RMW, see [3]). Next, we calculate the saturation convective available potential energy at the radius of maximum winds, $CAPE^*$. This calculates assumes the parcel is lifted directly from the sea surface, such that $T = T_s$ and $r = r_s(T_s)$ (where r_s is found given T_s via Eqns. 6 and 7):

```
> TP=SSTK
> PP=min([PM,1000.0])
> RP=0.622*ES0/(PP-ES0)
> result = cape(TP,RP,PP,T,R,P,ascent_flag,ptop,miss_handle)
> CAPEMS, TOMS, LNBS, IFLAG = result
```

As discussed in section 3.1.1, the OTL is defined as the level of neutral buoyancy for a saturated air parcel lifted from the sea level, and T_0 is the corresponding outflow temperature at that LNB. Critically, therefore, this step to find $CAPE^*$ also sets the OTL and the outflow temperature (T_0) that will be output upon convergence:

```
> TO=TOMS
> LNB=LNBS
```

The leading factor in Eqn. 4, the ratio $\frac{T_s}{T_0}$, is the scaling of PI by dissipative heating, which increases PI when it is considered ([2]). We set this ratio with the current iteration's outflow temperature and the (fixed) input sea surface temperature:

```
> RAT=SSTK/TO
```

The relevance of this ratio for the PI calculation is set by the user with the adjustable parameter `diss_flag` (section 3.1.2). If dissipative heating is allowed to impact the tropical cyclone potential intensity (`diss_flag`=1) then the ratio remains as defined above. If dissipative heating is neglected in the potential intensity calculation (`diss_flag`=0) then $\frac{T_s}{T_0} \rightarrow 1$:

```
> if (diss_flag == 0):
    > RAT=1.0
```

We next seek to estimate p_m at this step in the iteration, following Eqn. 5. The surface environmental virtual temperature, T_v, is given the average of virtual temperatures over the mean layer composed of the parcel (with temperature, T_s) and the lowest level, i.e. $T_v = \frac{1}{2}(T_{v,s} + T_{v,j=0})$. The virtual temperature is identical to the density temperature (Eqn. 12) at the surface (as $r_T = r$), and may be approximated at the lowest level with $r_T \approx r$:

```
> RS0=RP
> TV0=T[0]*(1.+R[0]/EPS)/(1.+R[0])
> TVAV=0.5*(TV0+SSTK*(1.+RS0/EPS)/(1.+RS0))
```

Combining Eqns. 4-5 to solve for p_m, the algorithm iterates towards a new pressure estimate:

```
> CAT=(CAPEM-CAPEA)+0.5*CKCD*RAT*(CAPEMS-CAPEM)
> CAT=max([CAT,0.0])
> PNEW=MSL*np.exp(-CAT/(RD*TVAV))
```

If the number of iterations exceeds 200 (an arbitrary chosen excessive number of iterations), or if the estimated pressure drops below the unphysical 400 hPa, then the PI algorithm fails to converge and returns missing outputs.

When the algorithm has successfully converged on a stable minimum pressure in the RMW, p_m, then the final central minimum pressure must be set. Assuming cyclostrophic balance and that the azimuthal velocity in the eye is given by $V = V_{max}(\frac{R}{RMW})^b$, we follow a power law scaling with exponent, b (see also [7] their Eqns. 25-26). pyPI assumes (following the BE02 MATLAB code) that $b = 2$, then p_{min} is found by scaling p_m:

$$p_{min} = p_{msl} * e^{(-\frac{CAPE|_{RMW} + \frac{1}{2}(1+\frac{1}{b})V_{max}^2}{RdT_v})} \tag{16}$$

which we solve in the code to find the `PMIN` to be output by the algorithm:

```
> CATFAC=0.5*(1.+1/b)
> CAT=(CAPEM-CAPEA)+CKCD*RAT*CATFAC*(CAPEMS-CAPEM)
> CAT=max([CAT,0.0])
> PMIN=MSL*np.exp(-CAT/(RD*TVAV))
```

We note that the difference, $CAPE|_{RMW} - CAPE_e$, is typically small. Historically, when $CAPE_e$ was used to compute PI in the final term of Eqn. 4, it was found to add noise to the PI calculations (Kerry Emanuel, *personal communication*). Therefore, we instead replace this term with $CAPE^* - CAPE|_{RMW}$ in the PI computation for tractability[5].

Finally, we are equipped to find the tropical cyclone potential intensity. Assuming that the raw computed maximum gradient wind speeds are scaled to 10 m winds with some fraction `V_reduc` (section 3.1.2), then:

```
> FAC=max([0.0,(CAPEMS-CAPEM)])
> VMAX=V_reduc*np.sqrt(CKCD*RAT*FAC)
```

This final step completes the PI computation in pyPI. We set the flag to indicate successful computation of PI (`IFLAG=1`) and return V_{max} (`VMAX`), p_{min} (`PMIN`), the flag, T_0 (`TO`) and the OTL (`LNB`) to the program level which is calling **pi.py**.

5 Contributors & Acknowledgements

I am particularly grateful to Kerry Emanuel for the development of potential intensity theory, for the use of his PI algorithm, and for his encouragement and permission to distribute this project. pyPI was developed by me (Daniel Gilford), with contributions from Daniel Rothenberg; many thanks to Daniel for his help optimizing the pi.py module. Thanks also to Dan Chavas, Jonathan Lin, and

[5] PI calculations with the original [3] formulation have identical OTLs and outflow temperatures, but tend to have higher V_{max} values by between 0 and 32 ms^{-1} (not shown). Global and tropical (20°S-20°N) mean differences from this approximation are ∼3 ms^{-1} and ∼2.5 ms^{-1}, respectively.

Raphael Rousseau-Rizzi for helpful comments on pyPI and this documentation. pyPI was born as an adaptation of the BE02 MATLAB code I used throughout my PhD thesis work ("The Tropopause Region Thermal Structure and Tropical Cyclones", [18]); as an example it recreates of a portion of our research on PI seasonality, published in [16] and [17] by the American Meteorological Society. Those works (and pyPI which follows) benefited from helpful discussions with Paul O'Gorman, Allison Wing, and my PhD advisor (and personal scientific hero) Susan Solomon.

6 License & Citation

Permissions are provided following the MIT License; please see **LICENSE**.

If you make use of pyPI, please include the following citation:

> Gilford, D. M. 2020: pyPI: Potential Intensity Calculations in Python, v1.3, Zenodo, doi:10.5281/zenodo.3985975.

7 Feedback?

If you have any questions, comments, suggestions for improvement, or spot an error, please email daniel.gilford@rutgers.edu or visit ∼/pyPI/issues.

References

[1] Michael M. Bell, Michael T. Montgomery, and Kerry A. Emanuel. "Air–Sea Enthalpy and Momentum Exchange at Major Hurricane Wind Speeds Observed during CBLAST". In: *Journal of the Atmospheric Sciences* 69.11 (2012), pp. 3197–3222. ISSN: 0022-4928. DOI: 10.1175/JAS-D-11-0276.1.

[2] M. Bister and K. A. Emanuel. "Dissipative heating and hurricane intensity". In: *Meteorology and Atmospheric Physics* 65 (1998), pp. 233–240. ISSN: 0177-7971. DOI: 10.1007/BF01030791.

[3] Marja Bister and Kerry A. Emanuel. "Low frequency variability of tropical cyclone potential intensity 1. Interannual to interdecadal variability". In: *Journal of Geophysical Research: Atmospheres* 107.Part 1 (2002). ISSN: 01480227. DOI: 10.1029/2001JD000776.

[4] David Bolton. "The Computation of Equivalent Potential Temperature". In: *Monthly Weather Review* 108.7 (1980), pp. 1046–1053. DOI: 10.1175/1520-0493(1980)108<1046:TCOEPT>2.0.CO;2.

[5] Suzana Camargo et al. "Tropical cyclone genesis potential index in climate models". In: *Tellus A: Dynamic Meteorology and Oceanography* 59.4 (2007), pp. 428–443. DOI: 10.1111/j.1600-0870.2007.00238.x.

[6] D. P. Dee et al. "The ERA-Interim reanalysis: configuration and performance of the data assimilation system". In: *Quarterly Journal of the Royal Meteorological Society* 137.656 (Apr. 2011), pp. 553–597. ISSN: 00359009. DOI: 10.1002/qj.828. URL: http://doi.wiley.com/10.1002/qj.828.

[7] K. A. Emanuel. "Sensitivity of tropical cyclones to surface exchange coefficients and revised steady-state model incorporating eye dynamics". In: 52.22 (1995), pp. 3969–3976. ISSN: 00224928. DOI: 10.1175/1520-0469(1995)052<3969:SOTCTS>2.0.CO;2.

[8] Kerry Emanuel. "A Statistical Analysis of Tropical Cyclone Intensity". In: *Monthly Weather Review* 128 (2000), pp. 1139–1152. ISSN: 0027-0644. DOI: 10.1175/1520-0493(2000)128<1139:ASAOTC>2.0.CO;2.

[9] Kerry Emanuel. "Tropical Cyclones". In: *Annual Review of Earth and Planetary Sciences* 31 (2003), pp. 75–104. ISSN: 0084-6597. DOI: 10.1146/annurev.earth.31.100901.141259.

[10] Kerry A. Emanuel. "An Air-Sea Interaction Theory for Tropical Cyclones. Part I: Steady-State Maintenance". In: 43.6 (1986), pp. 585–605. ISSN: 0022-4928. DOI: 10.1175/1520-0469(1986)043<0585:AASITF>2.0.CO;2.

[11] Kerry A. Emanuel. *Atmospheric Convection*. Oxford University Press, 1994, p. 592. ISBN: 978-0195066302.

[12] Kerry Emanuel et al. "Influence of Tropical Tropopause Layer Cooling on Atlantic Hurricane Activity". In: *Journal of Climate* 26.7 (Apr. 2013), pp. 2288–2301. ISSN: 0894-8755. DOI: 10.1175/JCLI-D-12-00242.1.

[13] S. Fueglistaler et al. "Tropical tropopause layer". In: *Reviews of Geophysics* 47.1 (2009). DOI: 10.1029/2008RG000267. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2008RG000267.

[14] Stephen Garner. "The Relationship between Hurricane Potential Intensity and CAPE". In: *Journal of the Atmospheric Sciences* 72.1 (2015), pp. 141–163. ISSN: 0022-4928. DOI: 10.1175/JAS-D-14-0008.1.

[15] Ronald Gelaro et al. "The Modern-Era Retrospective Analysis for Research and Applications, Version 2 (MERRA-2)". In: *Journal of Climate* 30.14 (2017), pp. 5419–5454. DOI: 10.1175/JCLI-D-16-0758.1. eprint: https://doi.org/10.1175/JCLI-D-16-0758.1. URL: https://doi.org/10.1175/JCLI-D-16-0758.1.

[16] Daniel M. Gilford, Susan Solomon, and Kerry A. Emanuel. "On the Seasonal Cycles of Tropical Cyclone Potential Intensity". In: *Journal of Climate* 30.16 (2017), pp. 6085–6096. DOI: 10.1175/JCLI-D-16-0827.1.

[17] Daniel M. Gilford, Susan Solomon, and Kerry A. Emanuel. "Seasonal Cycles of Along-Track Tropical Cyclone Maximum Intensity". In: *Monthly Weather Review* (2019), pp. 2417–2432. ISSN: 0027-0644. DOI: 10.1175/MWR-D-19-0021.1.

[18] Daniel Michael Gilford. "The Tropopause Region Thermal Structure and Tropical Cyclones". PhD Thesis. Massachusetts Institute of Technology, 2018, pp. 1–207. URL: https://dspace.mit.edu/handle/1721.1/115639.

[19] Mark D. Powell. "Evaluations of Diagnostic Marine Boundary-Layer Models Applied to Hurricanes". In: *Monthly Weather Review* 108.6 (1980), pp. 757–766. ISSN: 0027-0644. DOI: 10.1175/1520-0493(1980)108<0757:EODMBL>2.0.CO;2.

[20] Hamish A. Ramsay. "The Effects of Imposed Stratospheric Cooling on the Maximum Intensity of Tropical Cyclones in Axisymmetric Radiative–Convective Equilibrium". In: *Journal of Climate* 26.24 (Dec. 2013), pp. 9977–9985. ISSN: 0894-8755. DOI: 10.1175/JCLI-D-13-00195.1.

[21] William J. Randel and Fei Wu. "Variability of zonal mean tropical temperatures derived from a decade of GPS radio occultation data". In: *Journal of the Atmospheric Sciences* 72 (2014), pp. 1261–1275. ISSN: 0022-4928. DOI: 10.1175/JAS-D-14-0216.1.

[22] William J Randel and E. J. Jensen. "Physical processes in the tropical tropopause layer and their roles in a changing climate". In: 6.February (2013), pp. 169–176. DOI: 10.1038/NGEO1733.

[23] David M. Romps. "Exact Expression for the Lifting Condensation Level". In: *Journal of the Atmospheric Sciences* 74.12 (2017), pp. 3891–3900. DOI: 10.1175/JAS-D-17-0102.1. eprint: https://doi.org/10.1175/JAS-D-17-0102.1.

[24] John Wallis. *A treatise of algebra, both historical and practical*. eng. printed by John Playford, 1685. DOI: 10.3931/E-RARA-8842.

[25] Shuguang Wang et al. "Impact of the Tropopause Temperature on the Intensity of Tropical Cyclones: An Idealized Study Using a Mesoscale Model". In: *Journal of the Atmospheric Sciences* 71.11 (Nov. 2014), pp. 4333–4348. ISSN: 0022-4928. DOI: 10.1175/JAS-D-14-0029.1.

[26] Allison A. Wing, Kerry Emanuel, and Susan Solomon. "On the factors affecting trends and variability in tropical cyclone potential intensity". In: *Geophysical Research Letters* 42.20 (2015), pp. 8669–8677. ISSN: 19448007. DOI: 10.1002/2015GL066145.

[27] Elena Yulaeva, James R. Holton, and John M. Wallace. "On the Cause of the Annual Cycle in Tropical Lower-Stratospheric Temperatures". In: *Journal of the Atmospheric Sciences* 51.2 (1994), pp. 169–174. ISSN: 0022-4928. DOI: 10.1175/1520-0469(1994)051<0169:OTCOTA>2.0.CO;2.