SyCLoPS User Manual

The SyCLoPS paper: **The System for Classification of Low-Pressure Systems (SyCLoPS): An All-in-One Objective Framework for Large-scale Datasets (Han & Ullrich, 2025)**. Link to the article: https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2024JD041287

Documentation about the **TempestExtremes(TE)** software can be found **HERE**.

Some more comments about the TE commands in SyCLoPS and all the SyCLoPS codes can be found in `SyCLoPS_main.py` on **SyCLoPS @ GitHub**. The 1970-2024 ERA5 low-pressure system (LPS) catalogs are available via **SyCLoPS @ Zenodo.**

Known issues

1. The master branch of TempestExtremes now lacks the full ability to calculate parameters with missing values (e.g. 1e20/1e15) (Note: NaN values are automatically skipped in TE, so they are OK to proceed with). Therefore, it is now not directly applicable to some model outputs and reanalyses that have missing values where the data level is below the surface. We are working on this issue and users can expect a newer TE version with fixes in the near future. **For now, users can use this fork of TempestExtremes, which can be found here: https://github.com/yepkids/tempestextremes, to work around this problem**. This fork provides a temporary solution that adds missing value support for operators used by SyCLoPS and has been tested. Note that this is not a stable release, and please report any problems with this fork to Yushan Han (yshhan@ucdavis.edu). You can also choose to convert all missing values in your input files to NaNs.

Table of Contents:

The two main steps to implement SyCLoPS are:

1. Review the codes and comments in `SyCLoPS_main.py` on GitHub. Change variable names and other specifications according to your needs.

2. Run `SyCLoPS_main.py` and follow the instructions carefully.

There are several optional steps for SyCLoPS applications which are discussed in section 3.2 of this manual.

If this manual does not answer your questions about SyCLoPS, please contact the author Yushan Han at **yshhan@ucdavis.edu**

1. Look-up Tables and the Classification Flowchart

This section reproduces several tables and the SyCLoPS workflow diagram from the SyCLoPS manuscript with some more details.

1.1 Variable requirements

Variable Name	Pressure Level (hPa)
U-component Wind (U)	925, 850, 200^a (,500^b)
V-component Wind (V)	925, 850, 200^a (,500^b)
Temperature (T)	850
Relative Humidity (R)^c	850, 100^d
Mean Sea Level Pressure (MSL)	Sea Level
Geopotential (Z) / Height (H)	Surface (invariant)^e, 925, 850^e, 700, 500, 300^a
Relative Vorticity (VO)^b	500

a. These default levels of U,V,Z can be replaced with the more commonly found 250-hPa level. See Appendix B in the SyCLoPS paper for details.
b. Relative Vorticity (VO) can also be calculated from U and V at 500 hPa if not directly available. See comments in

"SyCLoPS_main.py."
c. Specific humidity can be converted to relative humidity (R) with temperature data at 850 and 100 hPa. To get the R at 100 hPa accurately, you may need to convert it as the relative humidity with respect to ice.
d. The daily frequency for R at 100 hPa is sufficient to maintain good performance of the SyCLoPS classification. See Appendix B in the SyCLoPS paper for details.
e. Z or H at 850 hPa and the surface level is optional if the data set has missing/fill values where the data plane intersects the surface. See comments in "SyCLoPS_main.py."

P.S. 10 m U and V component wind variables (VAR_10U,VAR_10V) are optional and can be used to calculate the maximum surface wind speed (WS) of an Low-Pressure System (LPS) as reference information in the classified catalog. They do not affect the detection and classification process.

1.2 LPS Initialism Table

Initialism	Full Term	Definition
HAL	High-altitude Low	LPSs found at high altitudes without a warm core
THL	Thermal low	Shallow systems featuring a dry and warm lower core
HATHL	High-altitude Thermal Low	LPSs found at high altitudes with a warm core
DOTHL	Deep (Orographic) Thermal Low	Non-shallow LPSs featuring a dry and warm lower core often driven by topography
TC	Tropical Cyclone	LPSs that would be named in IBTrACS
TD	Tropical Depression	Tropical systems that have developed a weak upper-level warm core and are strong enough to be recorded as TDs in IBTrACS
TLO	Tropical Low	Non-shallow tropical systems that fall short of TD requirements
MD	Monsoon Depression	TDs developing in monsoonal environment. A monsoon environment is considered to be dominated by westerly winds (resulting in asymmetric wind fields in monsoon LPS) and very humid Labeled as "TD(MD)" in the classified catalog. TDs that fall short of the monsoonal system condition are labeled "TD"
ML	Monsoon Low	TLOs developing in monsoonal environment. Labeled as "TLO(ML)" in the classified catalog. TLOs that fall short of the monsoonal system condition are labeled "TLO"

Initialism	Full Term	Definition
MS	Monsoonal System	Monsoon LPSs (MDs plus MLs)
TLC	Tropical-Like Cyclone	Non-tropical LPSs that resemble TCs (typically smaller than TCs). For example, they can have gale-force sustained surface wind, well-organized convection (sometimes with an eyewall) and a deep warm core
SS (STLC)	Subtropical Storm (Subtropical Tropical-Like Cyclone)	A type of TLC in the subtropics, represented by Mediterranean hurricanes
PL (PTLC)	Polar Low (Polar Tropical-Like Cyclone)	A type of TLC typically found north of the polar front
SC	Subtropical Cyclone	A type of LPS that is typically associated with a upper-level cut-off low south of the polar jet and has a shallow warm core
EX	Extratropical Cyclone	Most typical non-tropical cyclones
DS	Disturbance	Shallow LPSs or waves with weak surface circulations. DSD, DST and DSE are dry, tropical and extratropical DSs
QS	Quasi-stationary	LPSs that stay relatively localized as labeled by the QS track condition

1.3 The Input LPS Catalog Column Table (for *SyCLoPS_input.parquet*)

Column	Unit	Description
TID	-	LPS track ID (0-based) in both the input and classified catalog
ISOTIME	-	UTC timestamp (ISO time) of the LPS node in both catalogs
LON	°	Longitude of the LPS node in both catalogs
LAT	°	Latitude of the LPS node in both catalogs
MSLP	Pa	Mean sea level pressure at the LPS node in both catalogs
MSLPCC20	Pa	Greatest positive closed contour delta of MSLP over a 2.0° Great Circle Distance (GCD), representing the core region of an LPS
MSLPCC55	Pa	Greatest positive closed contour delta of MSLP over a 5.5° GCD
DEEPSHEAR	$\mathrm{m\,s^{-1}}$	Average deep-layer wind speed shear between 200 hPa and 850 hPa over a 10.0° GCD

Column	Unit	Description
UPPTKCC	$\mathrm{m^2\,s^{-2}}$	Greatest negative closed contour delta of the upper-level thickness between 300 hPa and 500 hPa over a 6.5° GCD, referenced to the maximum value within 1.0° GCD
MIDTKCC	$\mathrm{m^2\,s^{-2}}$	Greatest negative closed contour delta of the middle-level thickness between 500 hPa and 700 hPa over a 3.5° GCD, referenced to the maximum value within 1.0° GCD
LOWTKCC[a]	$\mathrm{m^2\,s^{-2}}$	Greatest negative closed contour delta of the lower-level thickness between 700 hPa and 925 hPa over a 3.5° GCD, referenced to the maximum value within 1.0° GCD
Z500CC	$\mathrm{m^2\,s^{-2}}$	Greatest positive closed contour delta of geopotential at 500 hPa over a 3.5° GCD referenced to the minimum value within 1.0° GCD
VO500AVG	$\mathrm{s^{-1}}$	Average relative vorticity over a 2.5° GCD
RH100MAX	%	Maximum relative humidity at 100 hPa within 2.5° GCD
RH850AVG	%	Average relative humidity over a 2.5° GCD at 850 hPa
T850	K	Air temperature at 850 hPa at the LPS node
Z850	$\mathrm{m^2\,s^{-2}}$	Geopotential at 850 hPa at the LPS node
ZS	$\mathrm{m^2\,s^{-2}}$	Geopotential at the surface at the LPS node
U850DIFF	$\mathrm{m\,s^{-1}\,sr}$	Difference between the weighted area mean of positive and negative values of 850 hPa U-component wind over a 5.5° GCD
WS200PMX	$\mathrm{m\,s^{-1}}$	Maximum poleward value of 200 hPa wind speed within 1.0° GCD longitude
RAWAREA[b]	$\mathrm{km^2}$	The raw defined size (see appendix E) of the LPS
LPSAREA	$\mathrm{km^2}$	The adjusted defined size of the LPS in both catalogs (see appendix E)

a. 925 hPa may be replaced by 850 hPa if data at this level are not available in some datasets.

b. This parameter in the column is for user reference only. It does not affect any results in the SyCLoPS LPS node or track classification.

1.4 Classification Condition Table

Condition Name	Conditions
High-altitude Condition[a]	Z850 > ZS
Dryness Condition	RH850AVG < 60%
Cyclonic Condition	VO500AVG >= (<) 0 s^{-1} if LAT >= (<) 0°
Tropical Condition	RH100MAX > 20%; DEEPSHEAR < 18 $\mathrm{m\,s}^{-1}$; T850 > 280 K
Transition Condition	Tropical Conditon = True; DEEPSHEAR $>$ 10 $\mathrm{m\,s}^{-1}$ or RH100MAX < 55%
TC Condition	MSLPCC20 > 215 Pa; LOWTKCC < 0 $\mathrm{m}^2\ \mathrm{s}^{-2}$; UPPTKCC < -107.8 $\mathrm{m}^2\ \mathrm{s}^{-2}$
TD Condition	MSLPCC55 > 160 Pa; UPPTKCC < 0 $\mathrm{m}^2\ \mathrm{s}^{-2}\}$
MS Condition	U850DIFF > 0 $\mathrm{m\,s}^{-1}$; RH850AVG > 85%
TLC Condition[b]	MSLPCC20 > 190 Pa; LOWTKCC and MIDTKCC $<$ 0 $\mathrm{m}^2\ \mathrm{s}^{-2}$;(LPSAREA < 5.5 × 105 km^2; LPSAREA > 0 km^2) or (MSLPCC20 > 420 Pa; MSLPCC20 : MSLPCC55 > 0.5)
SC Condition	LOWTKCC < 0 $\mathrm{m}^2\ \mathrm{s}^{-2}$; Z500CC $>$ 0 $\mathrm{m}^2\ \mathrm{s}^{-2}$;WS200PMXc > 30 $\mathrm{m\,s}^{-1}$
TC Track Condition	At least 8 (8+) 3-hourly TC-labeled nodes in an LPS track
MS Track Condition	10+ 3-hourly "TLO(ML)" or "TD(MD)"-labeled nodes
SS Track Condition	2+ 3-hourly TLC-labeled nodes ("SS(STLC)" or "PL(PTLC)")
PL Track Condition	2+ 3-hourly TLC-labeled nodes ("SS(STLC)" or "PL(PTLC)")
QS Track Condition	See SI text S3 for details

a. It can be as simple as checking the availability of T850 (or Z850) data (have null/missing values or not) in some records.

b. See section 5.3 in the SyCLoPS paper for a possible alternative.

c. The WS200PMX criteria used in this framework may be supplemented by other parameters in some regional models. See Supporting Information (SI) text S4 for details.

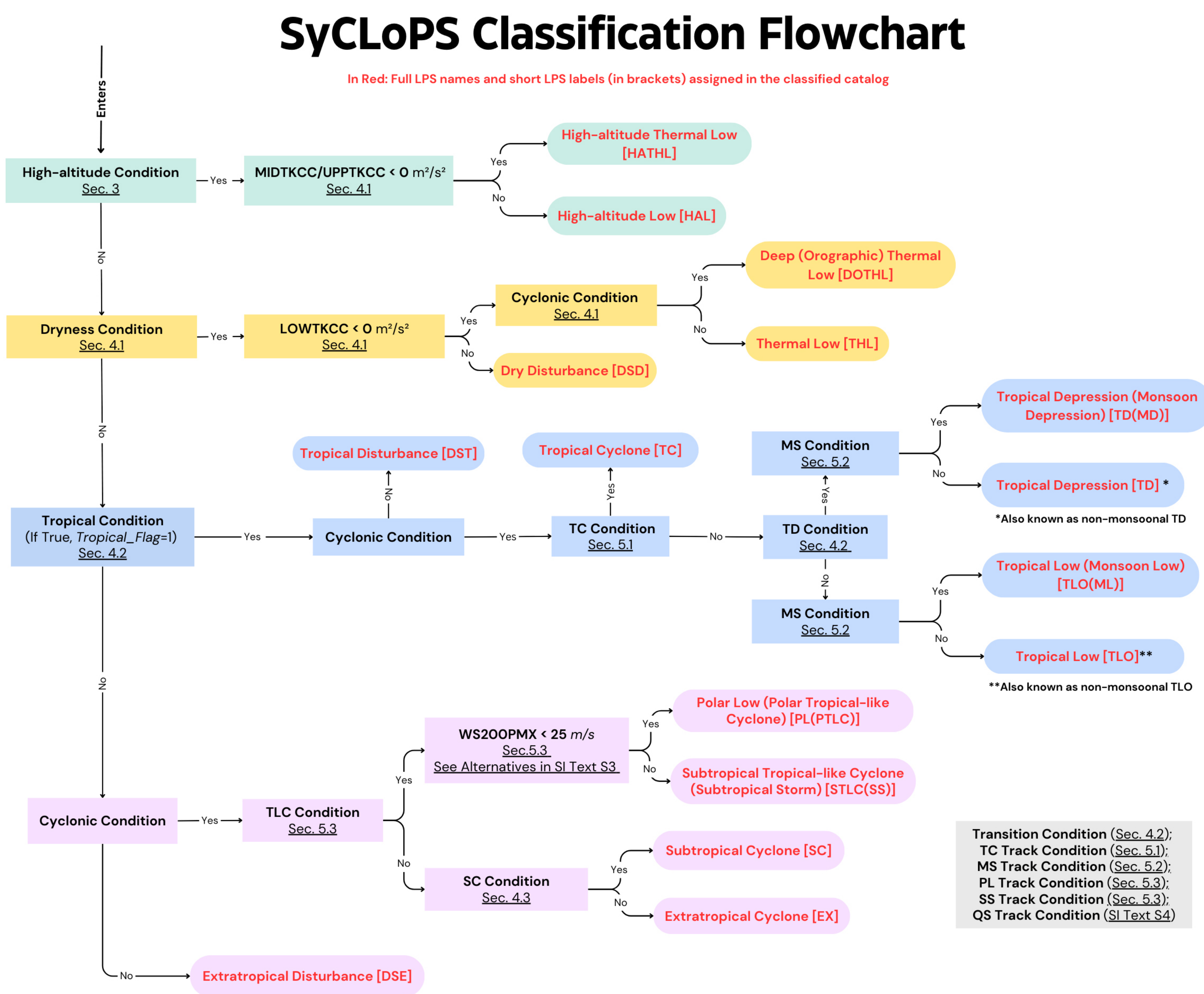
1.5 The Classified LPS Catalog Column Table (for *SyCLoPS_classified.parquet*)

Column	Unit	Description
TID	-	LPS track ID (0-based) in both the input and classified catalog
ISOTIME	-	UTC timestamp (ISO time) of the LPS node in both catalogs
LON	°	Longitude of the LPS node in both catalogs
LAT	°	Latitude of the LPS node in both catalogs
MSLP	Pa	Mean sea level pressure at the LPS node in both catalogs
WS*	$\mathrm{m\,s^{-1}}$	Maximum wind speed at the 10-m level within 2.0° GCD
Full_Name	-	The full LPS name based on the classification
Short_Label	-	The assigned LPS label (the abbreviation of the full name)
Tropical_Flag	-	1 if the LPS is designated as a tropical system, otherwise 0
Transition_Zone	-	1 if the LPS is in the defined transition zone, otherwise 0
Track_Info	-	"TC", "MS", "SS(STLC)", "PL(PTLC)", "QS" denoted for TC, MS, SS, PL and QS tracks; "EXT", "TT" denoted for extratropical and tropical transition completion nodes
IKE*	TJ	The integrated kinetic energy computed based on the LPS size blobs that are used to define RAWAREA

* These two columns are for user reference only. They do not affect any results in the SyCLoPS LPS node or track classification.

1.6 SyCLoPS Classification Flowchart and Assigned Labels and Full Names

Section numbers in the figure refers to the section numbers in the SyCLoPS manuscript.



2. Tips for running SyCLoPS

2.1 Installing TE

- Note that TE is currently not able to handle missing values (e.g. 1e20 or 1e15; NaNs are OK) in some datasets. For now, users can download and install this fork of TE here: **https://github.com/yepkids/tempestextremes**. This fork provides a temporary solution that adds missing value support to the operators used by SyCLoPS. We'll post a official release to fix this issue in the near future. The master TE is loacted at: https://github.com/ClimateGlobalChange/tempestextremes

- It is recommended that you download and compile the TE source using the provided `quick_make_general.sh` file, as TE may not be updated frequently on the conda channel.

- Please refer to the TE documentation for detailed explanations of each function, argument, and operation: https://climate.ucdavis.edu/tempestextremes.php

- To install and run TE in parallel, please make sure that your computer has an MPI implementation installed (e.g., Open MPI). When using MPI in TE commands, simply add `srun -n 128` or `mpirun -np 128` to the beginning of some TE commands to enable parallel computation. For more details, see the MPI documentation provided online or by your supercomputer host.

- DetectNodes, DetectBlobs and VariableProcessor support MPI computation in TE version 2.3.x. The parallelization is achieved by writing the *inputfile* as a list of files, each containing variables of a time slice. See the next subsection for details.

- If you have any questions about installing the TE software, please contact Paul Ullrich at paullrich@ucdavis.edu.

2.2 TE Operations explained in the SyCLoPS main program

1. You will first need to specify the installation path of TE on your computer and adjust other specifications in the beginning of the srcipt mannually, if necessary.

2. Prepare a list of files (in txt format) containing all the required variables (see table in 1.1) required by the first DetectNodes operation as the `$inputfile`. The files in the list should be arranged in time slices, like this:

   Variable1_TimeSlice1;Variable2_TimeSlice1;Variable3_TimeSlice1;...
   Variable1_TimeSlice2;Variable2_time_slice2;Variable3_TimeSlice2;...
   Variable1_TimeSlice3;Variable2_time_slice3;Variable3_TimeSlice3;...
   ...

3. The `$outputfile` txt file should contain the same number of lines as the inputfile. So it should have a filename for each time slice on each line, which should look like this: ERA5_LPSnode_out_TimeSlice1
   ERA5_LPSnode_out_TimeSlice2
   ERA5_LPSnode_out_TimeSlice3
   ...

   Below is a sample shell script to list 4 different variables (Z, MSL, U and ZS (the constant surface geopotential)) with different time slices to generate the input file along with a corresponding output file (the outputfile is generated simultaneously):

In [ ]:

```
'''
ERA5DIR=/global/cfs/projectdirs/m3522/cmip6/ERA5

mkdir -p LPS
rm -rf ERA5_example_in.txt # the input file
rm -rf ERA5_example_out.txt  # the output file

for f in $ERA5DIR/e5.oper.an.pl/*; do
# In this example ERA5 directory, variables are stored in folders named by years and months (e.g., 202001,2
    yearmonth=$(basename $f)
    year=${yearmonth:0:4}
    echo "..${yearmonth}"
    if [[ $year -gt '1978' ]] && [[ $year -lt '2023' ]]
    then
        for zfile in $f/*128_129_z*; do
            zfilebase=$(basename $zfile)
            yearmonthday=${zfilebase:32:8}
            mslfile=`ls $ERA5DIR/e5.oper.an.sfc/${yearmonth}/*128_151_msl*`
            ufile=`ls $ERA5DIR/e5.oper.an.pl/${yearmonth}/*128_131_u.*${yearmonthday}*`
            zsfile=./e5.oper.invariant.Zs.ll025sc.nc
            echo "$zfile;$mslfile;$ufile;$zsfile" >> ERA5_example_in.txt
            echo "LPS/era5.LPS.node.${yearmonthday}.txt" >> ERA5_example_out.txt
        done
    fi
done
'''
```

- Please also note that TE uses the time series of the first file in a row of a list (in the example above, the $zfile or geopotential file) to determine the time slices to look for for the rest of the files in that row. Therefore, it's recommended to put the variable file with the shortest time interval at the beginning of each row to avoid raising errors. For example, if the geopotential file is divided into days and other varialbles' files are divided into months, then the geopotential file should be placed at the beginning of each row.

- Here's another example shell script to list 4 different variables (Z, MSL, U10 and ZS, the constant surface geopotential) with different time slices for the inputfile in a customized data directory with filenames containing data's time period (e.g., 20100101):

In [ ]:
```
'''
#!/bin/bash

DIR="/path/to/your/folder"

# Extract unique dates from filenames
dates=$(ls "$DIR" | grep -oP '\d{8}' | sort -u)

for date in $dates; do
    echo "msl_${date}.nc;u10_${date}.nc;z_${date}.nc;zs_${date}"
done
'''
```

4. If you use MPI in applicable TE commands, each thread will take one time slice (row) in the `$inputfile` at a time and output a corresponding output file. You will need to set `use_srun = True` in `SyCLoPS_main.py`.

5. If you use variables on a unstructured grid (i.e., not lon-lat grid), you will need to set `use_connect = True` and specify a connectivity file in `SyCLoPS_main.py`.

6. **DetectNodes**: This command detects candidate LPS nodes and computes the 15 parameters needed for the classification:

    - This step can be time consuming. It's highly recommended to run this command in parallel, feeding it with a list of files ordered by time slices.

- The time dimension in the invariant surface geopotential file for ZS/Z0 should be removed (averaged) prior to the following procedures (if they have not alreadyß). It can be achieved by something like: "ncwa -a time ZS_in.nc ZS_out.nc."

- "WS" (the near-surface maximum wind speed within 2.0 GCD of the LPS node) is an optional parameter for reference purpose. Add `_VECMAG(VAR_10U,VAR_10V),max,2.0` to the end of the `--outputcmd` if you want to output this parameter. "Z850" is also not needed if your data has missing values where the 850 hPa data level is below the surface.

- `_CURL{16,2.5}(u(500hPa),v(500hPa)),min,0` (if using 25-50km resolution models) or `_CURL{8,2.5}(u(500hPa),v(500hPa)),min,0` (if using lower resolution models) can replace "VO(500hPa),avg,2.5" for VO500AVG if the relative vorticity (VO) is not directly available. The results will be slightly different, but close enough for the purpose. Another option is to calculate VO at 500 hPa using U and V at 500 hPa and precede normally.

- It is suggested to add the `--mergeequal` argument espeically for ERA5 datasets. This argument merge nodes that have the exact same MSLP values nearby in rare scinarios. ERA5 tends to have more of these cases because it has a relatively low precision (2 decimal places). It's extremely rare to have other reanaylsis and model data to have this issue.

7. **StitchNodes**: This command stitches all detected nodes in sequence with parameters formatted in a csv file.

   - If you are using a 6-hour detection rate in *DetectNodes*, you may consider either doubling the "4.0" in the `--range` argument (used for the default 3-hourly resolution) to "8.0" and adding a new `--prioritize MSLP` argument at the end of the command, or increasing the `--range` argument to "6.0" instead. The `--prioritize` argument will prevent false connections at the supposed end of a track when "range" is greater than "mergedist" in *DetectNodes* (See SI text S6 for further details). A 6.0 GCD range is also considered sufficient to cover tropical systems and the vast majority of the fast-moving extratropical cyclones within 6 hours. Follow these general rules when using data with different time resolutions.

The following TE commands (7-9), which are reserved for computing LPSAREA and generating blob files, can be omitted if you are not labeling tropical-like cyclones If you do not need to identify tropical-like cyclones (e.g., only classifying extratropical cyclone, subtropical cyclone and extratropical disturbance in the extratropical branch) or if you are only focusing on tropical systems, you may choose to skip the following steps when prompted in the main program. You can also skip the steps related to computing LPSAREA in `SyCLoPS_classifier.py` in this scenario. 7. **VariableProcessors**: We calcualted the cyclonic relative vorticity here in this step by computing a smoothed 850 hPa relative vorticity (RV) field and revert the sign of RV in the southern Hemisphere to get the cyclonic RV. This command can be run in parallel. Your inputfile should conatin files of U and V at 850 hPa and 925 hPa. If you are specifically interested in LPSs close to mountainous regions above 850 hPa level, it's recommended that you use 700 hPa data instead.

8. **DetectBlobs**: This step is to generate LPS size blobs. It's recommended to run this command in parallel, feeding it with a list of files ordered by time slices. It's possible to use 850 and 700 hPa wind speed to replace the 925 hPa. It's recommended to slightly increase the wind speed threshold used in this step in this scenario. A similar tactic to generate LPS precipitation blobs is introdcued in the `TE_optional.sh` file on Zenodo.

9. **BlobStats**: Generate useful information for calculating LPSAREA and tagging blobs with labels. This step cannot be run in parallel but could potentially be time-consuming if you use "sumvar" to compute IKE of each LPS for reference purposes (note that IKE is not required for classification). In this case, one may opt to use the GNU parallel tool to run multiple commands simultaneously, each using a single thread. To accomplish this, users should first create a txt file containing a list of TE commands to be parallelized (e.g. breaking down into years) and a list of corresponding input and output files. Then run `parallel -j n < blobstats_commands_list.txt` ("n" is the number of threads to use) to start the paralled BlobStats processes after `module load parallel`. You would also need to add files of U and V at 925 hPa to the inputfile list for IKE computation.

10. **The Classifier**: After running all TE commands, the program will prompt you to procceed with executing `SyCLoPS_classifier.py` when you are ready. You would first need to check the comments in `SyCLoPS_classifier.py` and change the specifications at the beginning of the classifier script if necessary.

2.3 Tips on applying SyCLoPS to climate model outputs

- Most high-resolution climate model outputs do not have all the required variables at 3 hours resolution, but they are mostly available at 6-hourly resolution, which is good enough for tracking most LPSs. However, relative humidity (RH) at 100 hPa is usally only available at a daily-mean basis. In the SyCLoPS paper, we show that using daily-mean relative humidity or 6-hourly detection rate will not lower SyCLoPS performance (except that TLC detection skill decreases as detection rate decreases).

- To use daily-mean relative humidity as the input, one must oversample the daily-mean RH files before putting them into TE, since TE's *DetectNodes* assumes that all the input files have the same data frequency that matches your detection rate (in this case, 6-hourly). For example, a typical daily-mean RH file contains only one data point per day, usually at

T12:00 or T00:00 of each day. You would need to resample it to 4 data points per day at 00,06,12,18 UTC to match your 6-hourly detection frequency. You can replicate the daily average four times for each date, or you can do a linear interpolation between time steps. TE may update a feature to address this inconvenience in the future.

- Climate model outputs may not contain the 300 hPa and 200 hPa data used in the default settings of SyCLoPS, but they typically have 250 hPa data available. We have also shown in the paper that using 250 hPa data instead of 200 hPa and 300 hPa does not degrade SyCLoPS performance with some minor adjustments (see Appendix B of the paper). In this senario, just use 250 hPa data to replace 200/300 hPa data and type "Y(Yes)" when asked if you want to use 250 hPa data instead of the default 200/300 hPa when running `SyCLoPS_classifier.py` at the last step of the main program.

2.4 Tips on applying SyCLoPS to regional model data

- Because of the nature of the closed contour critera used in SyCLoPS and TE, false LPS tracks will be detected near the four edges of the regional domain. Hence, it is recommended to define a ~2° buffer zone sourrounding the four boundaries of your regional domain to remove tracks in that zone in a post-processing process.

- Another option is to define `--minlat`, `--maxlat`, `--minlon`, `--maxlon` in the *DetectNodes* command. If your domain boundaries are not parallel to latitude or longitude, you can create a mask file as an input to *DetectNodes* to define a detection zone of your domain (with the grid labeled "1" in that zone). Then add something like `ZONE_MASK,=,1,0.0` to `--thresholdcmd` in the *DetectNodes* command.

- When running `SyCLoPS_classifier.py`, type "Y(Yes)" when asked if you are running with regional model data, so that the program will opt to use the alternative criteria designed for regional models (see SyCLoPS Supporting Information S4 for details) at several points in the classification process.

- Some models may only output specific humidity. In this case, you can use specific humidity and temperature to calculate relative humidity. Remember to calculate RH with respect to ice at 100 hPa.

3. SyCLoPS Catalogs Usages and Applications

3.1 How to select different types of LPS nodes and tracks in the classified catalog:

To open the classified catalog:

In [ ]:

```
import numpy as np
import pandas as pd

ClassifiedCata='SyCLoPS_classified.parquet' # your path to the classified catalog
dfc=pd.read_parquet(ClassifiedCata) # open the parquet format file. PyArrow package required.
```

In [2]:

```
dfc
```

Out[2]:

	TID	LON	LAT	ISOTIME	MSLP	WS	Full_Name	Short_Label	Tropical_Flag	Transition_Zon
0	0	56.75	70.00	1979-01-01 00:00:00	97686.00	12.66622	Extratropical Cyclone	EX	0.0	0.
1	0	56.75	69.75	1979-01-01 03:00:00	97869.81	12.43663	Extratropical Cyclone	EX	0.0	0.
2	0	57.50	69.50	1979-01-01 06:00:00	98085.94	12.29883	Extratropical Cyclone	EX	0.0	0.
3	0	57.75	69.25	1979-01-01 09:00:00	98294.25	11.26188	Extratropical Cyclone	EX	0.0	0.
4	0	59.25	69.25	1979-01-01 12:00:00	98454.31	10.92470	Extratropical Cyclone	EX	0.0	0.
...	...	...	...	...	...	...	...	...	...	.
7781101	379301	336.00	-60.50	2022-12-31 09:00:00	97146.94	12.96320	Extratropical Cyclone	EX	0.0	0.
7781102	379301	337.25	-60.75	2022-12-31 12:00:00	97272.25	13.02440	Extratropical Cyclone	EX	0.0	0.
7781103	379301	339.00	-61.00	2022-12-31 15:00:00	97358.38	13.75519	Extratropical Cyclone	EX	0.0	0.
7781104	379301	340.00	-60.75	2022-12-31 18:00:00	97431.12	13.85164	Extratropical Cyclone	EX	0.0	0.
7781105	379301	341.25	-60.75	2022-12-31 21:00:00	97573.38	14.35034	Extratropical Cyclone	EX	0.0	0.

7781106 rows × 13 columns

If desired, the input and ouput (classified) catalogs can also be combined to produce a larger catalog:

```
In [3]: # InputCata='SyCLoPS_input.parquet'
        # dfin=pd.read_parquet(InputCata)
        # dfc=pd.concat([dfc,dfin],axis=1)
```

Task 1. Select a single type of LPS node (e.g., TC):

```
In [4]: dftc=dfc[dfc.Short_Label=='TC']
```

Task 2. Select two types of LPS node (e.g., EX and SC):

```
In [5]: dfexsc=dfc[(dfc.Short_Label=='EX') | (dfc.Short_Label=='SC')]
```

Task 3. Select two types of TLC node (including SS(STLC) and PL(PTLC)):

```
In [6]: dftlc=dfc[dfc.Short_Label.str.contains('TLC')]
        dftlc
```

Out[6]:

	TID	LON	LAT	ISOTIME	MSLP	WS	Full_Name	Short_Label	Tropical_Flag	Transition_Z
24	1	347.00	68.00	1979-01-02 00:00:00	100267.10	20.49998	Subtropical Tropical-like Cyclone (Subtropical...	SS(STLC)	0.0	
53	2	359.50	58.00	1979-01-01 21:00:00	101263.30	13.46975	Subtropical Tropical-like Cyclone (Subtropical...	SS(STLC)	0.0	
55	2	1.50	56.00	1979-01-02 03:00:00	101188.20	15.56540	Subtropical Tropical-like Cyclone (Subtropical...	SS(STLC)	0.0	
57	2	3.25	54.00	1979-01-02 09:00:00	100931.10	17.09763	Subtropical Tropical-like Cyclone (Subtropical...	SS(STLC)	0.0	
58	2	3.75	53.00	1979-01-02 12:00:00	100894.20	18.27007	Polar Low (Extratropical Tropical-like Cyclone)	PL(PTLC)	0.0	
...	...	...	...	...	...	...	...	...	...	
7780923	379281	229.00	51.00	2022-12-30 21:00:00	98289.44	14.45700	Polar Low (Extratropical Tropical-like Cyclone)	PL(PTLC)	0.0	
7780924	379281	230.00	51.00	2022-12-31 00:00:00	98490.81	14.61593	Polar Low (Extratropical Tropical-like Cyclone)	PL(PTLC)	0.0	
7780925	379281	230.50	51.25	2022-12-31 03:00:00	98671.88	14.43227	Polar Low (Extratropical Tropical-like Cyclone)	PL(PTLC)	0.0	
7780926	379281	231.00	51.50	2022-12-31	98916.38	13.51526	Polar Low (Extratropical	PL(PTLC)	0.0	

	TID	LON	LAT	ISOTIME	MSLP	WS	Full_Name	Short_Label	Tropical_Flag	Transition_Z
				06:00:00			Tropical-like Cyclone)			
7780927	379281	231.25	51.50	2022-12-31 09:00:00	99244.94	12.39370	Polar Low (Extratropical Tropical-like Cyclone)	PL(PTLC)	0.0	

208514 rows × 13 columns

Task 4. Select all nodes in TC trcaks and get the track IDs (TID) of all TC tracks:

In [7]:
```
dftc2=dfc[dfc.Track_Info.str.contains('TC')]
tctid=pd.unique(dftc2.TID)
print(tctid)
```

```
[     17      21      59 ... 378821 378875 379061]
```

Task 5. Select all TC nodes in TC tracks:

In [ ]:
```
dftcms=dfc[(dfc.Short_Label=='TC') & (dfc.Track_Info.str.contains('TC'))]
```

Task 6. Select all DST and all TLO (including TLO and TLO(ML)) nodes in TC tracks:

In [9]:
```
dftc3=dfc[((dfc.Short_Label=='DST')|(dfc.Short_Label.str.contains('TLO'))) & (dfc.Track_Info.str.contains(
```

Task 7. Select LPS track IDs (TIDs) that have at least 5 non-tropical LPS nodes that are not DSE:

In [11]:
```
dfex=dfc[(dfc.Tropical_Flag==0)&(dfc.Short_Label!='DSE')]
extrackid=pd.unique(dfex.TID)[dfex.groupby('TID')['TID'].count()>=5]
```

Task 8. Select Tracks that are both a TC track, SS track and PL(PTLC) track:

In [12]:
```
tcsspl_trackid=pd.unique(dfc[(dfc.Track_Info.str.contains('TC')) & (dfc.Track_Info.str.contains('SS')) & ((
dftcsspl=dfc[dfc.TID.isin(tcsspl_trackid)]
```

Task 9. Select all non-tropical (extratropical) LPS nodes:

In [13]:
```
dfexnode=dfc[(dfc.Tropical_Flag==0)]
```

Task 10. Select all tropical TCs nodes in TC tracks that are not undergoing extratropical transition:

In [14]:
```
dftc3=dfc[(dfc.Track_Info.str.contains('TC')) & (dfc.Tropical_Flag==1) & (dfc.Transition_Zone==0)]
```

Task 11. Select all tropical transition completion nodes:

In [15]:
```
dftt=dfc[dfc.Track_Info.str.contains('TT')]
```

Task 12. Select TC tracks that **do not** undergo extratropical transition

In [16]:
```
tcnoext_trackid=pd.unique(dfc[(dfc.Track_Info.str.contains('TC')) & ~(dfc.Track_Info.str.contains('EXT'))]
```

Task 13. Select potential easterly wave (EW) nodes:

In [17]:
```
dfew=dfc[~(dfc.Track_Info.str.contains('M')) & ~(dfc.Track_Info.str.contains('Q')) & ~(dfc.Short_Label.str
```

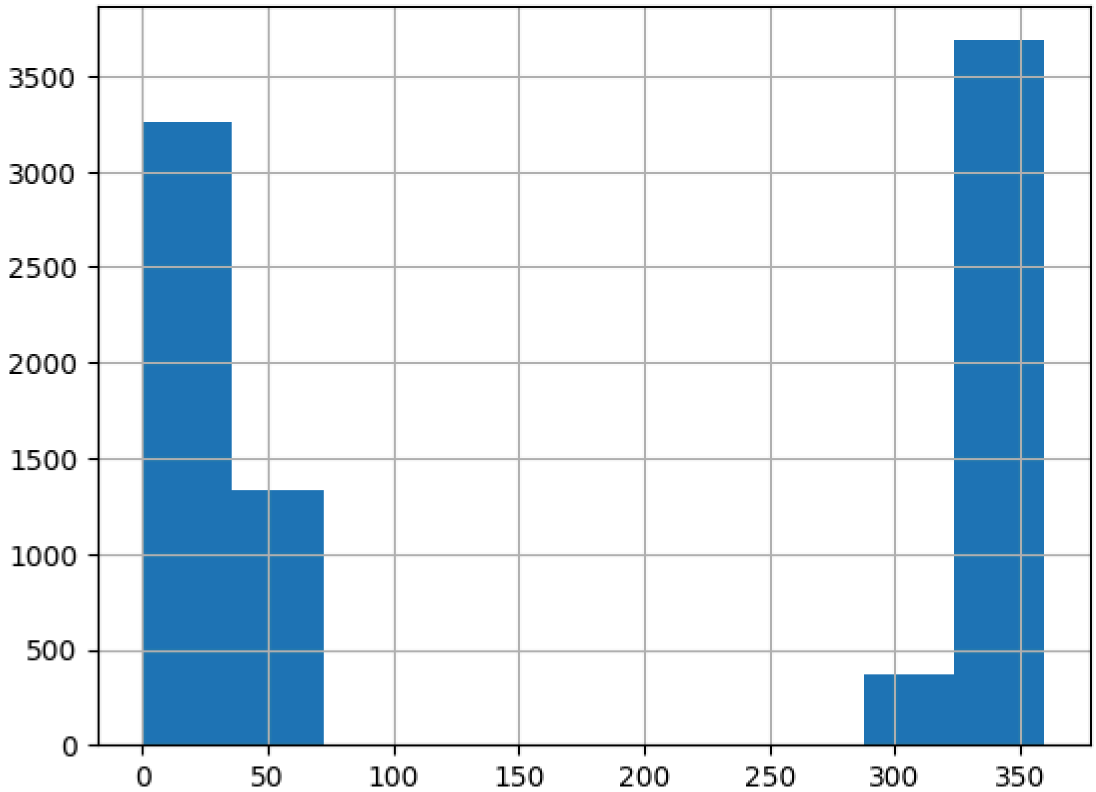
Task 14. Select all LPS nodes within a bounded region in January:

In [18]:
```
dflps=dfc[(dfc.LAT>=30) & (dfc.LAT<=50) & (dfc.LON>=280) & (dfc.LON<=350) & (dfc.ISOTIME.dt.month==1)]
```

Task 15. Select all PL nodes in PL(PTLC) tracks in the Nordic Seas from 1979 to 1999

In [19]:
```
dfpl=dfc[(dfc.LAT>=45) & (dfc.LAT<=85) & (((dfc.LON>=320) & (dfc.LON<360)) | ((dfc.LON>=0) & (dfc.LON<=70)
         (dfc.Track_Info.str.contains('PL')) & (dfc.Short_Label=='PL(PTLC)') & (dfc.ISOTIME.dt.year>=1979)
dfpl.LON.hist()
```

Out[19]:
```
<Axes: >
```



3.2 Other applications based on the classified catalog:

Here we introduce two additional usages of SyCLoPS: calculating intergataed kinetic energy (IKE) accumulation or precipitation contribution percentage for a specific type of LPS.

To perform this task, users need to run `Blob_idtag.py` and `TE_optional.sh`. Users can opt to run the additional TE commands within `Blob_idtag_app.py' (lines 13-15). The procedure can be divided into five steps:

1. The additional TE commands in `TE_optional.sh` detect precipitation blobs and calculate blob statistics (properties) using *BlobStats* in addition to the size blobs already detected in `SyCLoPS_main.py`.
2. Both size and precipitation blobs are masked with a unique ID (1-based, e.g., 1,,2,3,4,5,...) through *StitchBlobs*.
3. The blob-tagging Python script (`Blob_idtag.py`) pairs precipitation blobs to LPS nodes in the same way as we did for size blobs.

4. The Python script assigns tags (labels) to different blobs according to their paired labeled LPS nodes and the blobs IDs given by *BlobStats*. The assigned tags are then used to remask blobs with the tag numbers (e.g., 1=TC, 2=MS, 3=SS, 4=PL, 5=others) in the *StitchBlobs*'s output nc files.
5. Finally, run TE commands demonstrated in the "additional steps" in `TE_optional.sh` to extract 3-hourly precipitation and 925 hPa IKE at each grid point conatined within each size/precipitation blobs that are associated with a tag number (i.e., a type of LPS).

In step four, the Python script uses a tagging arrangement like we described in the last section of the SyCLoPS manuscript. However, there are many ways one can assign those tags. In the manuscript, we define TC blobs (with tag=1) as those blobs that are paired with TC nodes in TC tracks, which corresponds to these nodes in the classified catalog:

In [20]:
```
tcid=dfc[(dfc.Short_Label=='TC') & (dfc.Track_Info.str.contains('TC'))].index.values
blobtag=np.ones(len(dfc))*5 #5 = Other systems
blobtag[tcid]=1
# Subsequent codes in the Python script: ...
```

However, one can also define that blobs paried with all TC nodes (not only those in TC tracks) are considered TC blobs with tag=1:

In [21]:
```
tcid=dfc[dfc.Short_Label=='TC'].index.values
blobtag=np.ones(len(dfc))*5 #5 = Other systems
blobtag[tcid]=1
# Subsequent codes in the Python script: ...
```

One may also define that blobs paried with all tropical LPS nodes in TC tracks are considered TC blobs with tag=1:

In [22]:
```
tcid=dfc[(dfc.Tropical_Flag==1) & (dfc.Track_Info.str.contains('TC'))].index.values
blobtag=np.ones(len(dfc))*5 #5 = Other systems
blobtag[tcid]=1
# Subsequent codes in the Python script: ...
```

If you are using a multiple tag system (e.g. having tag = 1,2,3,4,and more), please be careful not to have overlapping paired LPS nodes among different tags (i.e., making them all mutually exclusive). The below example shows a **bad** practice:

In [23]:
```
tcid=dfc[(dfc.Tropical_Flag==1) & (dfc.Track_Info.str.contains('TC'))].index.values
msid=dfc[(dfc.Tropical_Flag==1) & (dfc.Track_Info.str.contains('MS'))].index.values
```

```
blobtag=np.ones(len(dfc))*5 #5 = Other systems
blobtag[tcid]=1 #1=TCs
blobtag[msid]=2 #2=MSs
# Subsequent codes in the Python script: ...
```

The above codes will produce overlapped LPS nodes within `tcid` and `msid`. This will cause some TC node IDs to be overwritten by the subsequent MS IDs.

You may also just ouput one kind of tag (e.g., just tag=1) for a group of LPSs:

In [24]:
```
msid=dfc[(dfc.Short_Label.str.contains('M')) & (dfc.Short_Label=='TC') & (dfc.Track_Info.str.contains('MS'
blobtag=np.ones(len(dfc))*0 # Other systems are all labeled 0
blobtag[msid]=1 #1=MSs
```

Another example:

In [25]:
```
ssid=dfc[(dfc.Short_Label=='SS(STLC)') & (dfc.Track_Info.str.contains('SS')) & ~(dfc.Track_Info.str.contai
blobtag=np.ones(len(dfc))*0 # Other systems are all labeled 0
blobtag[ssid]=1 #1=SSs
```

In the above two examples, blobs that are not tagged (masked) "1" will be tagged (masked) "0." In binary masking, "0" means that blobs are not detected. Hence, the final output NetCDF blob files will only contain blobs with tag (mask)=1 associated with the desired LPS group.

After tags are assigned to blobs as described in the Python script, they will be used to alter the original blobs masks in the NetCDF files output by *StitchNodes*. If one groups the blob ids in terms of the tag assigned, it will look something like this:

Tag number	Blob IDs
1	50, 139, 236, 337, 438, 553, 554, 663, ...
2	46, 137, 235, 335, 434, 436, 550, 660, ...
3	121, 244, 709, 719, 849, 861, 935, 1153, ...
4	1261, 1324, 1431, 1535, 1637, 1748, 1753, 185, ...
5	1, 2, 3, 4, 5, 6, 7, 8, ...

The output nc files with these alternations will contain blobs with their assigned tag numbers. For example, if tags 1-5 are used, grid points in each blob will be either masked 1, 2, 3, 4 or 5.

Finally, after implementing the last step (step 5) in TE, one can easily calculate the final accumulated IKE of each type of LPS over a period of time by summing each time frame of the output NetCDF files. To calculate the precipitation contribution percentage of a type of LPS, one should first calculate the total precipitation over a period by summing each time frame of the 3-hourly (or other frequency) precipitation file without any blob masks. Then do the same procedure, but with the precipitation blob masks output by TE. Lastly, the (annual/seasonal) contribution percentage of a type of LPS can be easily performed.

Update Date: 2025/07/08