Quick Guide to TPV Tracker

Nick Szapiro

October 2018

1 Version

The default is the "unified" branch. Other branches provide additional modifications (e.g., parallelization, modifications for cold pools, etc.). Parallel versions (e.g., branch vMPI) have been used for several applications, including tracking over historical reanalysis, climate simulations, and operational ensembles.

2 Python setup

TPVTrack is written in Python as it is portable, friendly to write/read, and broadly used across atmospheric science. Written in Python 2.7, the necessary libraries are NumPy and netCDF4, with optional MPI for Python (for parallel version) and Matplotlib and Basemap (for post-processing). For Anaconda Python users, all packages in the Python environment for the manuscript are in tpvTrack/docs/environment.yml.

3 Overview

Running [python driver.py] on the command line calls driver.py:demo(), which uses the user-provided information in my_settings.py to pre-process meteorological data into the TPVTrack fomat, segment the continuous surface into discrete objects of restricted (anti-)cyclonic watershed basins, describe the basins by geometric metrics, associate basins over time by overlap similarity into major and minor correspondences, and track TPVs along major correspondences. It's sequential (e.g., have to run the segmentation before calculating metrics). While reading the source code is more comprehensive, the main modules are:

my_settings User-defined configuration with inclued comments. Make sure to change the doPreproc, doSeg,... to True to run what you want and then False for any re-runs (to reduce subsequent runtime).

preProcess Take whatever input dataset(s) the user has and (1) generate the underlying Cell and Mesh objects and (2) write the meteorological variables on the horizontal tracking surface (horizontal wind, vorticity, and potential temperature) to file.

For input data from a different source, the user will need to implement Mesh and Cell classes utilizing 1D indexing.

segment A watershed approach where regions are grouped by local gradients around regional extremum. We segment a surface into highs and lows by mapping each cell to its high xor low basin by the sign of its local relative vorticity. Not every resulting basin is in a TPV.

basinMetrics Given TPVs as basins, we can quantify properties like circulation, amplitude, area, moment of inertia,... using the basins as masks for geometric and meteorological fields.

correspondence Candidate correspondences are formed by "horizontal" overlap, and similarity is measured by combining "horizontal" and "vertical" overlap. The type of correspondence is then classified; a major connection is a 1-1 correspondence between TPV A at time t_0 and TPV B at time $t_0 + \Delta t$ if both (1) B is the most similar connection to A at time $t0 + \Delta t$ and (2) A is the most similar connection to B at time t_0. Then, a TPV splitting event is characterized by major and minor pieces. Setting trackMinMax to track min xor max is reasonable, but not both together as then minima could correspond to maxima.

tracks Tracks are formed by stitching together major correspondences over time. Genesis and lifetime criteria are used to define TPVs from tracks (e.g, a lifetime of at least 2 days and 60% of life north of 65° N).

4 my_settings.py

- rEarth Radius of the Earth (m)
- dFilter Filter radius for whether a local extremum is a regional extremum (m)
- areaOverlap Minimum fraction of horizontal overlap that qualifies as overlapping TPVs (typically 0.1 or smaller)
- segRestrictPerc Percentile of boundary amplitudes for restriction of watershed basins ([0,100])
- latThresh Segment polewards of specified latitude (e.g., 30° N)
- trackMinMaxBoth Track cyclones (0) or anticyclones (1)
- info Additional information added to metadata in NetCDF file
- filesData Meteorological input data ($\geq$ 1 file path)
- fileMap Only needed for WRF input data, NetCDF file with map projection information
- deltaT Timestep of input data (s)
- timeStart Start date of tracking (datetime.datetime)
- iTimeStart_fData Starting time index in each input file (0-based indexing)
- iTimeEnd_fData Ending time index in each input file (can use -1 for last time in file)
- fDirSave Directory to save output files
- fMesh Input NetCDF file with mesh information
- fMetr Output meteorological data in TPVTrack format (1-D flat arrays)
- fSeg Output segmentation file
- fMetrics Output metrics file
- fCorr Output correspondence file
- fTrack Output track file
- inputType Type of input data (eraI, mpas, wrf,...)
- doXXX True/False to run module (set to False if re-running and already have previous module output)

5 Output

5.1 driver.py:demo()

- fields.nc - NetCDF file of Meterological input data output from preProcess.py
- seg.nc - NetCDF file of segmentation basins
- metrics.nc - NetCDF file of geometric basin metrics
- correspond_horizPlusVert.nc - NetCDF file of time correspondence between basins
- tracks_low_horizPlusVert.nc - NetCDF file of tracks along major correspondences. TPV tracks are obtained as a subset by requiring additional conditions appropriate for a TPV.

5.2 driver.py:demo_algo_plots()

- seg*png - Maps of each time's segmentation with basins colored by site (e.g., Fig. 1.c)
- corr*png - Maps of correspondences between neighboring times. Basins at t0 are plotted as green +. Basins at t1 are plotted as red o. Minor correspondence are denoted by red, thinner lines. Major correspondences are denoted by blue, thicker lines
- test_tracks.png - Map of tracks with sufficient lifetime colored by core potential temperature

6 Example ERA-Interim test case

Using the default unified source code branch,

- Create a directory for the case, e.g., fDir = /home/user/test-tpvTrack/ . cd into fDir.
- Download ERA-I u, v, potential temperature for your time period 6-hourly (`http://apps.ecmwf.int/datasets/data/interim-full-daily/levtype=pv/`). A small example file is included

  `tpvTrack/example_eraI/ERAI_tpvTracker_2006-08-01To2006-08-04.nc`

- git clone `https://github.com/nickszap/tpvTrack.git`
- In my_settings.py, set (1) info='ERA-Interim test case' (or other descriptive string), (2) fDirData=fDir, (3)

  `filesData=['/home/user/test-tpvTrack/example_eraI/ERAI_tpvTracker_2006-08-01To2006-08-04.nc']`

  (to your own file), (4) timeStart= dt.datetime(2006,8,1,0) (to data's start date), (5) fDirSave=fDir (or other directory).
- Run *python driver.py* from the command line, which calls demo() and demo_algo_plots(). Note that .pyc files will be created (but ignored in .gitignore).
- Compare images to (1) included tpvTrack/test-tpvTrack/*png and (2) tropopause maps for consistency.

7 Further details for limited area WRF

- Interpolation to 2 PVU by searching down a column may find layers well into the stratosphere or not find a tropopause above the surface. This problem is more common for higher resolution simulations. If the recommended seeded flood-fill tropopause diagnostic is not used, several alternatives are possible. Iterative extrapolation from valid neighbors is one option. If the domain is limited in latitude, it may be reasonable to fill missing values of u,v,theta on 2 PVU with the domain mean. More appropriate probably would be to grab data from a different model output level.

- Advection is used to test whether TPVs at consecutive times overlap and can correspond. For limited area grids, points can advect outside the domain. We could either have some larger, enclosing domain. OR, we want to just ignore those points. Currently, any advected point whose closest cell is on the boundary of the domain is dropped. If a point is actually within the domain but closest to a boundary cell, it gets dropped too. Not optimal, but I didn't think of a simple way to test whether a point is in the bounds of a general WRF domain.

8 Future to-do list

Things to consider implementing:

- Acceleration: looping over cells in a mesh, looping over basins, segmenting various times,... are all pretty embarrassingly parallel
- Additional basin metrics
- Track metrics
- Track polewards of an identified jet stream or isentrope (rather than latitude)