Running HYSPLIT using Python and batch code

This manual provides a guide to the code developed for analyzing extratropical cyclones and their moisture sources using HYSPLIT trajectory modeling, ERA5 reanalysis data, and geospatial filtering techniques. This code is designed to identify and trace precipitating air parcels, evaluate their moisture uptake sources, and generate trajectory-based diagnostics in line with the paper [Sources and Transport Pathways of Precipitating Waters in Cold-Season Deep North Atlantic Cyclones (Papritz, 2021)](https://journals.ametsoc.org/view/journals/atsc/78/10/JAS-D-21-0105.1.pdf) – which I will refer to as “the Papritz paper” moving forward. This code assumes that you are a little familiar with HYSPLIT and Python (for troubleshooting).

The overall goal of the project was to better understand where Arctic storms acquire their moisture by using Lagrangian back-trajectory methods. The code performs the following functions:

* Storm track analysis: parses storm position data to identify locations relevant to the study.
* Parcel selection: searches for nearby air parcels that contributed to precipitation along the storm path using thresholds for specific humidity, cloud water content, and precipitation rate.
* Trajectory initialization: uses parcels to create HYSPLIT CONTROL files that drive backward trajectory simulations from multiple vertical levels.
* Trajectory processing: reads and parses tdump files into labeled formats for analysis, parcel tracking, and computation.
* Moisture uptake detection: identifies times and locations along each trajectory where significant increases in specific humidity occur, based on thresholds defined in the Papritz paper.
* Plotting: plots the moisture uptake footprint, HYSPLIT trajectories, and specific humidity changes

This manual is intended for future users or any researchers seeking to replicate or extend the workflow.

**Understanding and organizing files**

Understanding this ZIP file

Within this zip file, there are a few instructional documents. The contents are:

* **main\_code.ipynb:** a python notebook with all the python functions with markdown explanations, the content of the markdown cells is similar to this manual so you do not need to read both for a clear picture.
* **functions\_code.py:** a python file that holds just the functions but no markdown explanations. You can identify what sections of the code correspond to this manual using the comments.
* **run\_hysplit\_multilevel.bat:** a batch file that allows HYSPLIT runs for multiple levels of one storm in succession.
* **run\_hysplit\_singlelevel.bat:** a batch file that allows HYSPLIT runs for a single level of a storm.
* **ASCDATA.CFG:** an ASCII file that contains gridded land-use, terrain, and roughness length data for HYSPLIT.

In both python codes, I have included docstrings and comments to help you understand the functions. Please read them carefully. The functions should be run in the order they’re written in this manual and the jupyter notebook.

Organizing your files

For the python functions to work as is, your code should be in the same directory (called project) as your storm CSVs, CONTROL files, etc. You can store the ERA5 data on a hard drive. If this is not the case, please carefully change the file paths in the functions. Here is the internal file structure that you will have if you were examining one storm (SID = 13, 2003/02) at one height (1500 m):

.

└── project

├── main\_code.ipynb

├── storm13\_200302.csv

├── hysplit\_info

│ ├── metdata

│ │ ├── RP200302.gbl

│ │ └── RP200301.gbl

│ ├── storm13\_200302

│ │ └── height1500

│ │ ├── output

│ │ │ ├── tdump\_200302day02\_hour09\_0

│ │ │ ├── tdump\_200301day31\_hour21\_1

│ │ │ └── ...

│ │ ├── ASCDATA.CFG

│ │ ├── traj\_files\_200302.txt

│ │ ├── control\_names\_200302.txt

│ │ ├── CONTROL\_200302day02\_hour09\_0

│ │ ├── CONTROL\_200301day31\_hour21\_1

│ │ └── ...

│ └── traj\_data\_storm13\_200302

│ ├── moisture\_uptake\_storm13\_height1500m.csv

│ └── traj\_storm13\_height1500m.npz

└── D:/ (hard drive)

├── ERA5\_storm13\_2003

│ ├── HumidityCWC\_200302\_day27to31.nc

│ ├── HumidityCWC\_200302\_day1to6.nc

│ ├── precip\_SST\_SLP\_200302\_day27to30.nc

│ └── precip\_SST\_SLP\_200302\_day1to6.nc

└── CycloneTracking

└── 13\_2E5R

└── BBox10

└── AggregationSystem

└── EASE2\_N0\_25km\_GenesisRegions.nc

└── CSVSystem

└── 2003

├── System13\_2E5R\_BBox10\_200302.csv

│ └── ...

└── ...

As you can tell, there are a lot of files here so there’s a lot of potential for confusion and lost files. While this looks overwhelming, the contents of “project” will largely be created and organized by the code since the code is enabled to create folders as necessary.

**Using this code**

In this section, I will tell you the order of operations and a few tips.

This code can be run sequentially. After downloading the NC files and running all the helper functions, you can run the precipitating parcels code which will also create the CONTROL files for HYSPLIT. After running HYSPLIT using the batch code, you can load the data and save it using the subsequent functions. Finally, you can find the moisture uptake regions.

Your code can get messy if you are running several storms. For this reason, I suggest defining the nc\_paths dictionaries (see subsection 'Accessing NC files' for more information) near the top of the code where you load the libraries and the arctic mask. This makes it so you have it in an easily visible place and you only have one version of each dictionary.

Additionally, it is helpful to define a list of tuples with the storm identifiers and putting that at the top of the code as well. You can loop through this list for a lot of the functions, making it fast to run things with minimal rewriting. This is an example of what it may look like:

STORM\_INFO = [(13, 2003, 2, nc\_paths\_2003, 2), (24, 1995, 1, nc\_paths\_1995, 1), ...]

Each tuple represents a storm's (SID, year, month, nc\_path variable, metdata\_number), in that order. Thus, when you run a function like *find\_precipitating\_parcels*, you can use a code block like below to avoid retyping information.

radius, level = 1200, 1500

**for** info **in** STORM\_INFO:

SID, YEAR, MONTH, nc\_path, met\_num **=** info

p **=** find\_precipitating\_parcels(SID, YEAR, MONTH, level, nc\_path, met\_num, radius)

Finally, you can also put the function get\_dataset\_for\_time (see subsection 'Accessing NC files') at the start with the storm information list and nc\_paths dictionaries. It is used in multiple functions and will likely be helpful later in the project.

When you’re familiar with the functions, you can delete the docstrings. They are there for comprehensibility when you are first working with them but can make the functions a lot to scroll through.

My most common error was with accessing files incorrectly or overwriting files. Be careful in changing directory and filenames in each function to match your file structure and where you want the files to go. The variables that you need to change should be mentioned in the docstrings but give the function a read-through to make sure you're caught all the instances.

**Saving a specific storm in a CSV**

The *save\_storm\_csv* function extracts and saves trajectory tracking data for a specific storm (identified by its SID, year, month) from a larger CSV file containing multiple storms. It reads the full dataset from the specified folder path, filters out rows with missing values, and isolates the subset of data corresponding to the desired storm ID. The resulting storm-specific data includes the storm’s time, position, and cartesian coordinates and this is saved to a new CSV file in the local directory. This means you won’t have to open a massive CSV every time you want to find information about this particular storm, which is useful for analysis.

Some common errors that this function may throw (and their solutions) are:

* **PermissionError:** you may already have a file saved with this name in the same location, please move/rename/delete the existing file and try again.
* **ValueError:** the delimiter may not match what you have in your large CSV. Try " " or "\t", which are both common options.
* **OSError:** check the filename that it mentions in the error message, is that the file that holds the data for that storm? You may have to adjust the folder path.

**Finding and Recording Precipitating Parcels**

Now that you have the storms separated, you will need to find the precipitating parcels around this trajectory. This will be used to do the back trajectories in HYSPLIT. There are several subfunctions defined here because there are multiple steps. A precipitating parcel is an air parcel that meets all of the following criteria:

* Within a fixed distance to the storm track
* Total precipitation exceeds 0.1 mm/h.
* Cloud water content > 10 mg/kg
* Specific humidity decreased by at least 0.1 g/kg over the past hour.

I’m assuming a HYSPLIT run time of -120 hours (5 days backwards) in my explanation – if this isn’t the case for you, substitute the number in your head and change the function to fit the number of hours.

Downloading data

Before running this section, it’s important that you have ERA5 data downloaded as NC files (NOT grib). I downloaded it from this site <https://cds.climate.copernicus.eu/datasets>, specifically, the datasets "ERA5 hourly data on pressure levels from 1940 to present" and "ERA5 hourly data on single levels from 1940 to present". When downloading this data, make sure you have sufficient data that you can go back 15 hours more than your run time (135 total hours in my case) from the storm entering the Arctic. For example, if the storm is recorded in the Arctic from February 2nd to 10th 2003, you will need ERA5 data from January 27th to February 10th. To reduce the data size, I suggest breaking it down in 4–5-day long intervals, cutting off the latitude range (not too closely, spare 10 degrees on the lower and upper limit), and cutting off the pressure levels to above 250 hPa. You will need the following variables when finding the precipitating parcels: specific humidity, cloud ice water content, cloud liquid water content, and total precipitation. For some plots, the sea surface temperature and mean sea level pressure are also used. I recommend saving the humidity and cloud water contents in one file, and precipitation, sea surface temperature and sea level pressure in one file. I also suggest using the same time and grid conventions for all of your ERA5 datasets. For example, for the above storm, *all* the single level variables (precipitation, sea surface temperature, mean sea level pressure) should be defined from latitudes 0° to 90°, longitudes -180° to 180°, and times January 27th to February 10th. All the pressure level variables (cloud liquid/ice water content, specific humidity) should be defined for the same latitudes/longitudes and from 350 hPa to 1000 hPa, even if you break up the time interval. This makes it so you can share lat-lon grids or indices to access certain values.

To store the NC files, I recommend having a structured system. My system was that for a specific storm (ex. SID = 13, Year/Month = 2003/2), I would place all the ERA5 data in a folder called "ERA5\_storm13\_200302" that is in a hard drive.

The other data that you need for HYSPLIT is the meteorological data for HYSPLIT. This provides the background information for HYSPLIT to do its calculations. To download this data, use Filezilla. In the top bar, put the following information:

Host: ftp.arl.noaa.gov

Username: anonymous

Password: (leave blank)

Port: (leave blank)

then, click QuickConnect. In the remote site panel, navigate to /archives/reanalysis and find the file you need. They are organized as RP{year}{month}. On the local site panel, navigate to where you want it saved then drag it in or double click. Make sure you download enough meteorological data! If your storm is recorded from February 2nd to 10th 2003, you will need RP200302 AND RP200301, since the storm will backtrack into January. The number of meterological data files a storm needs corresponds to the metdata\_number in future functions. The example storm will have a metdata\_number of 2. A clue that you have an insufficient amount of meteorology data is that the HYSPLIT output will not be the length of the run\_hours you expected –- it will stop running when it hits the limit of the meteorology data.

Adjusting NC files

ERA5 reanalysis files store cloud liquid water content (clwc) and cloud ice water content (ciwc) as separate variables, but do not directly provide cloud water content (cwc) which is the variable we need. The *adjust\_ncfile* function automates this step by merging the clwc and ciwc variables into a single cwc variable. It then removes the original clwc and ciwc variables to reduce file size and save memory. The cleaned file is saved in a folder named according to the data’s year (e.g., ``ERA5\_2003``), and the filename includes the year, month, and day interval covered by the data. This preprocessing step ensures consistency across storm-specific datasets and simplifies downstream analysis. Feel free to change the organization method, as long as your file paths in the nc\_paths are accurate (see next subsection).

Accessing NC files

If you’re using multiple long storms, you will have lots of NC files and opening multiple NC files all at once can result in HDF errors. The function *get\_dataset\_for\_time* looks through a dictionary of NetCDF file paths and returns the dataset that contains the requested variable for the specified time. The dictionary entries include a start and end day so the function can find which file contains the given time. It opens the file using xarray and returns it. If no matching file is found, it raises a FileNotFoundError. This can occur if there isn’t data available for a day or if the file path/day interval is inaccurate. You may get an error message that says: “ValueError: unrecognized engine 'h5netcdf'”, this is dependent on the version of xarray you have loaded. If so, in the return line of the function, change the engine to “netcdf4”. As mentioned in the section ‘Using this code’, it may be helpful to put this function and the nc\_paths dictionary in a cell near the top.

Generating points within a radius

When searching for precipitating parcels, it’s necessary to restrict the search to a radius (in the Papritz paper, the radius is 500km but the function can be set to any radius in km). The *points\_within\_radius* function returns all latitude-longitude grid points within a specified geodesic radius of a given center point, using a grid centered on the storm location (lat0, lon0). It only returns points within the radius. The spatial resolution is set to 0.5° latitude and 1.0° longitude. You can modify within the function to coarsen or refine the resolution. The function ensures the longitude remains within the [-180, 180] range for longitude and [-90, 90] range for latitude.

Generating CONTROL files

This function prepares trajectory input files (CONTROL files) for use with HYSPLIT, based on a list of precipitating air parcels associated with a specific storm. Each parcel is assigned its own trajectory run, initialized from the parcel’s time and location. The function writes all required CONTROL files to storm-specific folders and logs the paths of those files for future batch processing. It also handles both single and dual month meteorological input configurations depending on whether the trajectory runs will span a one or two months (controlled via metdata\_number). The trajectory duration is set using the run\_hours argument and meteorological data is expected to follow the RP{YYYY}{MM}.gbl naming scheme. I recommend that you do not modify the control\_names\_{yyyy}{mm} file to include the level. The control files have the format explained here <https://www.ready.noaa.gov/hysplitusersguide/S262.htm> and refer to that for any changes to the format to fit your needs. The function also creates as ASCDATA.CFG file that matches the one included in the zip file. Please change it to fit your work, it is necessary for running HYSPLIT. Note that the paths for storm\_dir and met\_dir must be the complete path that ends with a slash! Change these according to your setup.

Finding precipitating parcels

The function *find\_precipitating\_parcels* identifies precipitating air parcels near an Arctic storm and generates the necessary input CONTROL files for running HYSPLIT back-trajectories. For each point along the storm track that lies within the Arctic (based on a global CAO2 mask), the function searches a defined radius for nearby grid points that meets the three conditions outlined in the Papritz paper. For each of these, a CONTROL file is created. The function will accept precipitating parcels that are *outside* the Arctic if the storm center is within the Arctic. This function relies on preprocessed NetCDF weather data and several helper functions for locating relevant variables in space and time. If you have multiple levels and multiple storms to run, this may take a while. I recommend running the python script in PowerShell as it is a little faster. If you are considering a level not listed in the function, you can use <https://www.mide.com/air-pressure-at-altitude-calculator> to find the pressure (Pa) that corresponds to the altitude you are considering, then converting that to hPa. If you have changed the file naming or storage conventions in earlier functions, please adjust this function!

**Running HYSPLIT with batch files**

A batch file is a script file that can be executed sequentially by the command-line interpreter. This is the most efficient way to run a bunch of HYSPLIT CONTROL files all at once. These files can be harder to read than Python files.  Any lines that start with REM are comments -- I like to add '====' around each comment. You can edit batch files in notepad and run it by calling it in command-line or (on Windows) by double-tapping the file. Before attempting to run HYSPLIT, please verify that the following is true:

* There is a copy of ASCDATA.CFG in each folder that contains CONTROL files. I have included the one that I used. Confirm that this is the one that you should also use.
* The meteorological data is stored in the folder the CONTROL files say it is.
* There is a control\_names file in the same folder as the CONTROL files.
* There is a directory for the tdump files to output into, which matches the CONTROL files’ instructions.

HYSPLIT for multiple levels

If you want to run HYSPLIT for multiple levels (for example, levels 1500m, 3000m, 5500m) for the same storm, use the batch file *``run\_hysplit\_multilevel.bat****``***. After accessing the folder for the storm, it will access each level’s folder where it will use control\_names to run each CONTROL file through HYSPLIT. I’ve attached a copy of the batch file below, with the yellow highlighted sections being text to change depending on the storm or your setup. In order of appearance, here is how you should change it:

1. BASEDIR must be the folder that contains the height folders that you want to run HYSPLIT on. Change this every storm you run.
2. These are the heights you want to run. This may or may not be the same from storm to storm, but it was for me.
3. This must match the control\_names file in each height folder. This means it must be changed with each storm (to match the year and month).
4. This is a one-time change, it is the folder that stores hyts\_std.exe. If you used the default loading of HYSPLIT, this should not change for you.

@echo off

setlocal enabledelayedexpansion

set BASEDIR=C:\Users\project\hysplit\_info\storm13\_200302

REM ==== Subfolders to loop through ====

for %%F in (height1500 height3000 height5500) do (

set "WORKDIR=%BASEDIR%\%%F"

echo Working in: !WORKDIR!

pushd "!WORKDIR!"

REM ==== Loop through the list of CONTROL file paths in the current folder ====

FOR /F "usebackq tokens=\* delims=" %%n in ("control\_names\_200302.txt") do (

REM ==== Clean old files ====

IF EXIST "CONTROL" DEL "CONTROL"

IF EXIST "tdump" DEL "tdump"

IF EXIST "SETUP.CFG" DEL "SETUP.CFG"

echo Running trajectory for: %%n

REM ==== Copy current CONTROL file into place ====

copy /Y "%%n" "CONTROL" >nul

REM ==== Run HYSPLIT (hyts\_std.exe must read from this folder) ====

C:\hysplit\exec\hyts\_std.exe)

popd)

endlocal

pause

HYSPLIT for a single level

If you want to run HYSPLIT for a single level, then use this batch file. Much of the inner workings here are the same as the multilevel version above. I’ve attached a copy of the batch file below, with the yellow highlighted sections being text to change depending on the storm. In order of appearance, here is how you should change it:

1. You must cd to the folder for the storm and level you want to run HYSPLIT on. Change this every storm/level you run.
2. This must match the control\_names file in each height folder. This means it must be changed with each storm (to match the year and month)
3. This is a one-time change, it is the folder that stores hyts\_std.exe. If you used the default loading of HYSPLIT, this should not change for you.

@echo off

setlocal enabledelayedexpansion

cd C:\Users\project\hysplit\_info\storm13\_200302\height1500

REM ==== Loop through the list of CONTROL file paths ====

FOR /F "usebackq tokens=\* delims=" %%n in ("control\_names\_200302.txt") do (

REM ==== Clean old files ====

IF EXIST "CONTROL" DEL "CONTROL"

IF EXIST "tdump" DEL "tdump"

IF EXIST "SETUP.CFG" DEL "SETUP.CFG"

echo Running trajectory for: %%n

REM ==== Copy current CONTROL file into place ====

copy /Y "%%n" "CONTROL" >nul

REM ==== Run HYSPLIT (hyts\_std.exe must read from %DIR%) ====

C:\hysplit\exec\hyts\_std.exe

popd)

endlocal

pause

**Loading Trajectory Data**

The *load\_tdump\_files* function reads HYSPLIT trajectory data from tdump files associated with a specific storm event (identified by their SID, date, and level). It uses a text file to locate all the relevant tdump files, skips over metadata lines, and extracts trajectory data points (year, month, day, hour, latitude, longitude, height (m-AGL), and pressure), saving each trajectory as a NumPy array. In the tdump files, the year is a 1 or 2 digit number (3 for 2003, 95 for 1995). This function will save those values in the npz files as the four-digit year. All trajectories are returned as a list of arrays. Optionally, the function can call *save\_traj\_npz* to store the data in a compressed npz file, where each trajectory is assigned a unique traj\_id. The npz files contains column names and trajectory data to make it easier to filter for individual trajectories during analysis. Together, these functions streamline the loading, formatting, and saving of HYSPLIT output files. The benefit of npz files is that they’re compressed and easier to use in Python. However, they cannot be opened using excel like a CSV would.

**Finding moisture uptake parcels**

The *diagnose\_moisture\_uptakes* function identifies when and where HYSPLIT air parcel trajectories experience a rapid increase in specific humidity (called a moisture uptake event in the Papritz paper) -- any hourly increase greater than 0.025 g/kg is marked as an uptake. These events are saved to one or more CSV files so that the diagnosis only needs to be done once, even if the kernel is reset. The function loads trajectories from a npz file, loops through each parcel, and checks for changes in humidity using *get\_dataset\_for\_time()*. If no uptake events are found, the function returns the trajectory data. This makes it easier to manage large sets of storm data and track the sources of moisture in Arctic storms.

Since the moisture uptakes may be split into multiple CSV files, you can use the function *load\_all\_moisture\_csvs* to access all the CSVs at once to create an array of all the moisture uptakes. The primary function of splitting up the uptakes is avoiding a MemoryError, which can occur if you try to open an exceptionally large file in Python. Make sure that the way that you saved the moisture uptake CSVs matches the glob pattern in this function.

**Plotting**

This section will provide sample code for a helper function and 3 different plots:

* Draw a circle on a plot
* A simple plot of a storm track
* A figure similar to Figure 3 in the Papritz paper
* A figure similar to Figure 4 in the Papritz paper

Note that the 2 last plots will not exactly match Figures 3 and 4 in the Papritz paper but provides a good jumping off point for those plots. The differences between those plots and the paper’s figures will be explained here. Depending on your project it may be helpful to create a directory in the folder that holds your code to save all of your plots into.

Draw a circle on a plot

This helper function can be used to draw a circle on any plot around a center lat/lon with a given radius (in km). It is fairly self explanatory but is useful if you want your plots to match Papritz.

A simple plot of a storm track

This will allow you to make a plot of a storm track, it is very barebones but can be easily customizable to fit your purposes. Please change SID, YEAR, MONTH to fit whatever your storm is. It plots the points that are outside the Arctic mask in black, and points within the Arctic mask as red. The starting point of the storm is a purple cross. You can use this base and plot multiple storms on the same map. Here, I'm assuming you're uploading the storm information from the CSV but this can be changed if you have the information in an array/list.

Figure 3 in Papritz’s paper

If you want to plot a moisture uptake footprint like in the Papritz paper, use this code. You will need to have already found the moisture uptakes and saved them in CSVs using the functions in 'Finding moisture uptake parcels'. Additionally, you will need the precipitation, mean sea level pressure, and sea surface temperature if you are using the first function (moisture\_uptake\_plot\_separate). These should all be defined in the storm's nc\_paths dictionary as this function uses `get\_dataset\_for\_time()` to plot these variables as contours. I'm assuming you can access all three variables through a key ["precip\_SST\_SLP"]. If this isn't true, then you'll need to change the key name where "ds\_precip\_sst\_msl" is defined. If the total precipitation is stored in a different key than the sea surface temperature and sea level pressure, then you should use `get\_dataset\_for\_time()` to get that nc file when plotting it. These should all be defined in the storm's nc\_paths dictionary as this function uses `get\_dataset\_for\_time()` to plot these variables as contours. I have also defined the colourmaps that were used in the Papritz paper.

A screenshot of a weather map

AI-generated content may be incorrect.The first function  moisture\_uptake\_plot\_separate will make a 1x3 plot of the moisture uptakes associated with 3 timestamps. The key difference here is that this function will plot the storm as different timestamps than the Papritz paper. Here, t=0h is when the storm enters the Arctic (center panel). It will also make a plot for -24h (left panel) and +24h (right panel). Thus, these footprints will be centered around Arctic entry, rather than intensity. While this level of specificity can be good, it will fail if there are no moisture uptakes associated with any of the three timestamps. You can choose to plot single or multiple layers in one plot. An example of the resulting plot can be found above (Plot 1).

Plot : Based on Figure 3 in Papritz, a plot of a storm's moisture uptake footprint at three separate timestamps combining all levels at each time. Here, t=0h is the Arctic entry time.

The second function moisture\_uptake\_plot\_together plots all of the moisture uptakes for a storm, regardless of time. You will not need the total precipitation, sea surface temperature, or sea level pressure for this. This gives you a look at the general shape of the storm uptakes, which is especially helpful if you have few moisture uptakes. An example of the resulting plot can be found below (Plot 3).

Much like the other functions, if the naming/sorting conventions of the moisture uptake information has changed, please change it in this function to avoid a FileNotFoundError.

Figure 4 in Papritz’s paper

If you want to plot the storm trajectories with scattered circles showing moisture uptakes, use this code. You will need to have already run HYSPLIT and found the trajectories to run this code. Much like the code for figure 3, this section includes code to plot with different timestamps (which is more similar to Papritz) and a single plot that combines all the data.

The first function ``storm\_trajectory\_plot\_separate()`` will make a 1x3 plot of the HYSPLIT trajectories associated with three timestamps. The three timestamps used here are the arctic entry time (when the system enters the Arctic domain), the midpoint of its Arctic residence time, and the final time. The function assumes that you have trajectories associated with these times. You can choose to plot single or multiple layers in one plot. The function also includes the mean sea level pressure associated with the timestamp, a circle representing a radius around the storm center, and a cross representing where the storm center was in the \*previous\* timestamp. Additionally, the function chooses a random location on each trajectory to find the 12-hour change in specific humidity and plots these. All of these elements are included in the Papritz plot.

A screenshot of a map

AI-generated content may be incorrect.Here is a sample of the resulting plot (Plot 2):

Plot 2: Similar to Papritz figure 4, this plot includes the HYSPLIT back trajectories associated with this storm and the changes in specific humidity (the circles). Additionally, it shows the main storm track and a 500km radius around the storm center and the mean sea surface pressure.

A screenshot of a map

AI-generated content may be incorrect.A screenshot of a map

AI-generated content may be incorrect.The second function *storm\_trajectory\_plot\_together()* will create 1x1 plot that shows ALL trajectories, regardless of time and level. I primarily used this function because the storms I was working with did not have a sufficient number of trajectories to plot separate timestamps like in the Papritz paper. This function uses get\_dataset\_for\_time to access the specific humidity. Feel free to work with this function to fit your purposes. An example of the resulting plot can be found below (Plot 4)

Plot 4: Based on Figure 4 in Papritz, this is every 4th trajectory associated with Storm 143 (1989/01) across all levels and times. The scattered circles are 12-hour changes in specific humidity.

Plot 3 : Based on Figure 3 in Papritz, this is ALL the moisture uptakes (across all levels and times) for Storm 143 (1989/01)

**Conclusion**

This manual has outlined the main components of the code used to analyze moisture uptake in Arctic storms using HYSPLIT back trajectories and ERA5 reanalysis data. By following the structure and explanations provided, users should be able to load trajectory data, extract relevant environmental variables, and identify where and when moisture enters the storm system using changes in specific humidity. Many of the functions are modular and reuseable, making it easier to adapt the workflow to new storms, datasets, variables, and search radii. If any issues arise, such as missing files, unexpected errors, or unfamiliar output, double-check that the input files are formatted currently and refer to the function docstrings and manual for the clarification. Additionally, reading online forums helped my process. If necessary, please email me at [kripa.vyas@mail.mcgill.ca](mailto:kripa.vyas@mail.mcgill.ca). With this setup, you can now follow Papritz’s method for diagnosing moisture uptake.