Code Map for the Flight Level Dataset Codeset

Overview

The codeset consists of several discrete steps, each of which is controlled by a main NCL script. In the future it may be possible to run these as a chained job process, but for now they are separate. More information about each step will be provided below, but this section provides a quick snapshot of what main program each does.

Step 1: process_raw_data.ncl - reads in all the raw flight level data for a given storm (Level 0 data, which consists of the various USAF and NOAA ASCII and netCDF formats). These data are then read into standardized variables and written to a combined common format netCDF file (Level 1).

Step 2: flight_plotter.ncl - reads in a common format netCDF file and generates plots of the raw earth-relative data for either one flight or all the flights for a given storm.

Step 3: parse_radials.ncl - reads in a common format netCDF file, the wind center track file, transforms the data to storm-relative coordinates, then parses the radial legs, writes out various metadata including the starting and stopping times for the legs, leg average pressure, etc. to a different netCDF file (Level 2).

Step 4: radial_plotter.ncl - uses the radial leg metadata to transform the data into storm relative coordinates, then retrieve the profiles, performs various calculations, plots the radial legs, writing the good radial legs out to a different netCDF file (Level 3)

Description of Dataset Levels

Level 0: The raw source data files from USAF and NOAA. These contain the flight level data in ASCII, netCDF, and the standard tape files. The Level 0 data also include the wind center track and fixes files (.trak and .fixs).

Level 1: The combined common netCDF file that holds all of the standardized earth-relative flight data for a given storm at the native temporal resolution of the Level 0 data.

Level 2: A netCDF file that holds the various metadata for the radial legs (both good and bad), such as the starting and stopping time of each leg, the average pressure, etc.

Level 3: A netCDF file that holds the flight data in three frames: (a) earth-relative coordinates (lat/lon, similar to Level1, but with a flag showing if it's a good leg or not); (b) storm-relative coordinates (x/y) centered on the stationary storm center; and (c) storm-relative coordinates in the frame of the moving storm center (the component of storm motion is subtracted from the winds). This file may also contain the data for the good radial legs on the fixed radial grid (100 m grid spacing).

How to run the code set

The codeset is currently setup to be run on any of the following five systems: sandy, boojum, yellowstone, caldera, or geyser. Because the processing is very processor intensive, it is recommended that for the parsing step, any storms with more than 15 flights be parsed on geyser.

Locations of codes and data:

sandy:

- source code: /sandy/jvigh/PROJECTS/FLIGHT/source/
- data: /data/jvigh/PROJECT_DATA/FLIGHT/

yellowstone/caldera:

- source code: /glade/p/work/jvigh/flight/source
- data: /glade/p/work/jvigh/flight/

Running Single Storm

A master run script is now available to run the various processing steps of the code for a given storm. The syntax is:

> **rmaster_flight.bash** <processing step> <stormname> <stormyear>

where <processing step> can be one of the following: **process**, **plot**, **parse**, **radial**. These steps are explained in more detail below.

It is also possible to submit a single parsing job to geyser using the bsub queue script: **launch_parser_on_geyser.bsub**.

Step 1: Read and combine the all the raw flight level data for a given storm using rprocess_storm.bash to drive the NCL script, process_raw_data.ncl:

> **rmaster_flight.bash process** <*stormname*> <*stormyear*>

Substep A: If you run into problems, you can use the following checker script to generate a listing of the data for a specific flight from the combined netCDF file or the raw flight level data for a given storm. This is a useful to verify that the program has read in the raw data correctly from the various ASCII and netCDF formats. This calls the NCL program check_data.ncl

> **./rcheck.bash** <*stormname*> <*stormyear*> <*iflight*>

Step 2: Read in the common format netCDF file and generate plots of the raw earth-relative data for either one flight or all the flights for a given storm (calls the NCL program flight_plotter.ncl):

> **./rmaster_flight.bash plot** <*stormname*> <*stormyear*>

Step 3: Read in the common format netCDF file, the wind center track file, transform the data to storm-relative coordinates, then parse the radial legs, write out various metadata including the starting and stopping times for the legs, leg average pressure, etc. to a different netCDF file (calls the NCL program parse_radial_legs.ncl)

> **./rmaster_flight.bash parse** <*stormname*> <*stormyear*>

Step 4: Use the radial leg metadata to transform the data and retrieve the profiles, perform various calculations, plot the radial legs (calls the NCL program radial_plotter.ncl):

> **./rmaster_flight.bash radial** <*stormname*> <*stormyear*>

Running Many Storms

A simple workflow manager driver program **FLIGHT_driver.ncl** has now been implemented to submit many processing/plotting/parsing/radial jobs to various nodes on caldera/geyser. Simply edit the options near the top of this script and then run:

> **ncl FLIGHT_driver.ncl**

Future development of onboard real-time system

==========

The end version of the code could be broken into separate program units that accomplish the following tasks:

0) process all of raw flight level data into individual netCDF files with one standard format

- these files will be named as stormnamestormyear_flightID_raw.nc

1) read in all data for a given storm from individual netCDF files, create one big combined netCDF file

- this file will be named as stormnamestormyear_all_raw.nc

2) compute the wind centers, iterate as necessary, generate the spline tracks

3) transform flight data to the moving storm center, write out a netCDF file of this data for use with data assimilation and objective analysis schemes, plotting

- this file will be named stormnamestormyear_all_processed.nc

4) parse the radial legs, writes out a file with the info on the radial legs (this is an intermediate step - eventually, we'll want to put all the radial data into one file and plot later)

5) parse the data to the determined radial legs, write out a netCDF file with the radial leg data, create quick-look plots

- this file will be named <stormname><stormyear>_all_radials.nc