THE SENSITIVITY OF THE STRUCTURE AND EVOLUTION OF THE
2010 BOXING DAY BLIZZARD IN THE U.S. TO ENVIRONMENTAL TEMPERATURE

By

PHILLIP YEH

A thesis submitted to the

School of Graduate Studies

Rutgers, the State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Master of Science

Graduate Program in Atmospheric Science

Written under the direction of

Steven G. Decker

And approved by

_____________________________________

_____________________________________

_____________________________________

_____________________________________

New Brunswick, New Jersey

October 2019
ABSTRACT OF THE THESIS

The Sensitivity of the Structure and Evolution of the
2010 Boxing Day Blizzard in the U.S. to Environmental Temperature

by PHILLIP YEH

Thesis Director:
Steven G. Decker

Previous studies have discovered various changes that extratropical cyclones undergo due to climate change. One consistent result is the increasing importance of lower tropospheric processes, such as diabatic heating from latent heat release. Some studies have also continued to develop more advanced approaches to classify extratropical cyclogenesis. Some of these results on climate change and cyclone classification are applied to a pseudo-global-warming experiment of the 2010 Boxing Day blizzard, which affected the United States East Coast. The Weather Research and Forecasting Model is used to perform simulations of the storm at different environmental and surface temperatures. The results show several changes to both the upper tropospheric and lower tropospheric parameters for the storm at cyclogenesis, some of which were different from previous studies. A diabatic Rossby vortex also appears to influence cyclone development in each simulation. At the warmest temperature, the storm track shifts eastward, leading to less precipitation on land, and the surface low becomes slightly weaker.
Table of Contents

Abstract.................................................................................................................................ii

List of Tables............................................................................................................................iv

List of Figures..........................................................................................................................v

Introduction..............................................................................................................................1

Literature Review....................................................................................................................2

Synoptic Overview ................................................................................................................12

Model Setup and Methods......................................................................................................21

Results .....................................................................................................................................26

Conclusions............................................................................................................................46

Appendix..................................................................................................................................51

References...............................................................................................................................106
LIST OF TABLES

Table 1: List of the 24 precursors and their descriptions ..........................................................24

Table 2: Change in each of the precursors compared to the control run ..................................35

Table A1: Calculated values of the 24 precursors ...................................................................101
LIST OF FIGURES

Figure 1: Distribution of events on the PC space ..............................................................8
Figure 2: 300 hPa station plots and analysis at Dec. 25, 0Z ............................................12
Figure 3: 850 hPa and surface station plots and analyses at Dec. 25, 0Z .....................13
Figure 4: As in (Figure 3a), but at Dec. 25, 12Z ..............................................................14
Figure 5: As in (Figure 2), but at Dec. 26, 0Z .................................................................15
Figure 6: As in (Figure 3), but at Dec. 26, 0Z .................................................................16
Figure 7: 500 hPa and surface station plots and analyses at Dec. 26, 12Z .....................17
Figure 8: 300 hPa and surface station plots and analyses at Dec. 27, 0Z .................18
Figure 9: 500 hPa, 850 hPa, and surface station plots and analyses at Dec. 27, 12Z ....20
Figure 10: The nested model domain .............................................................................22
Figure 11: Storm tracks of all the simulations .................................................................25
Figure 12: All 24 precursors for the control run ..............................................................27
Figure 13: Cross section through surface cyclone center, forecast hour 12 ...............28
Figure 14: As in (Figure 13), but at forecast hour 24 .....................................................29
Figure 15: As in (Figure 13), but at forecast hour 36 .....................................................29
Figure 16: As in (Figure 13), but at forecast hour 48 .....................................................30
Figure 17: 850 hPa heights, frontogenesis, winds for control run ..................................31
Figure 18: Graph of each simulation on the PC space, forecast hour 12 ....................32
Figure 19: As in (Figure 18), but at forecast hour 24/30 ..............................................33
Figure 20: Time series of column-averaged EKE .........................................................36
Figure 21: The search region and calculated SLP ..........................................................37
Figure 22: As in (Figure 21), but for the T+10K case .................................................38
Figure 23: Time series of minimum SLP and mean 850-hPa PV ..................................................39

Figure 24: As in (Figure 13), but for the T+10K case .................................................................40

Figure 25: As in (Figure 14), but for the T+10K case .................................................................41

Figure 26: As in (Figure 15), but for the T+10K case .................................................................42

Figure 27: As in (Figure 16), but for the T+10K case .................................................................43

Figure 28: As in (Figure 17), but for the T+10K case .................................................................43

Figure 29: Cumulative precipitation ..............................................................................................44

Figure 30: Total liquid equivalent snowfall ..................................................................................45

Figure A1: As in (Figure 12), but for the T-5K case .....................................................................102

Figure A2: As in (Figure 12), but for the T+1K case .....................................................................103

Figure A3: As in (Figure 12), but for the T+5K case .....................................................................104

Figure A4: As in (Figure 12), but for the T+10K case .................................................................105
INTRODUCTION

Extratropical cyclones are a common phenomenon during a typical North American winter. Frequently, these storms produce a wide variety of precipitation types and greatly disturb daily activities (Cerruti and Decker 2011). One such storm in 2010, known as the Boxing Day blizzard, produced heavy snowfall and blizzard conditions stretching from North Carolina to Maine. Other similar storms have also brought flooding rain, power outages, property damage, and coastal erosion and have sometimes led to loss of life. Since these effects can depend on the track taken by the storm and its timing, scientists have dedicated much research into understanding these extratropical cyclones (e.g., Bjerknes and Solberg 1922; Sutcliffe 1938; Miler 1946; Eady 1949; Pettersen and Smeybe 1971; Schultz et al. 2018).

As climate change leads to warmer temperatures, it is imperative to have a thorough understanding of the different effects these warmer temperatures would have on extratropical cyclones. Previous research has shown that temperature is an important factor that can change the strength or frequency of storms (e.g. Colle et al. 2013). However, the relationship between temperature and cyclone strength is not necessarily linear or monotonic, and other variables may complicate the connections between the two. Consequently, this study aims to better quantify certain changes that occur in the structure of extratropical cyclones due to climate change, especially variations in temperature, with the focus on one actual cyclone.
LITERATURE REVIEW

Many studies have been conducted on the impacts of global warming on extratropical cyclones. Marciano et al. (2015) examined East Coast U.S. extratropical cyclones in the A2 emissions scenario (warm climate regime) from the IPCC AR4 report. Using WRF version 3.2.1 with 36 km grid spacing and 41 vertical levels, they ran 84-hour simulations of 10 Miller A (Miller 1946) storms in both the current climate regime and the warm climate regime. They took five general circulation models (GCMs) from the Coupled Model Intercomparison Project phase 3 (CMIP3) suite, calculated decadal averages for each month in the 1990-1999 period and the 2090-2099 period, and then used the average temperature change from the first period to the second period to set up the warm climate regime. They did not change relative humidity. These adjustments resulted in roughly a 2-4 K rise in temperature, with the greatest warming occurring in high latitudes and high altitudes. It was observed that most of the storms had lower pressures in the warm climate scenario, but one storm (the 2010 Boxing Day Blizzard) featured a higher sea-level pressure (SLP) minimum in the warm climate compared to the SLP minimum in the current climate. Storms also strengthened more rapidly and tended to shift eastward in future simulations. Additionally, they found that the upper level trough moved east in future simulations, and the tropopause pressure increased. When considering both atmospheric water vapor content and storm track changes, total precipitation from each storm was found to increase. However, as a result of higher temperatures, snowfall from the storms decreased in the future scenarios. Interestingly, positive anomalies of 2-meter potential temperature decreased slightly.
Tierney et al. (2018) experimented with both changes in wind profile (representing changes in baroclinicity) and atmospheric temperatures. They ran both idealized “dry” simulations (disabled moist processes in the model) and idealized “moist” simulations with WRF version 3.5.1. With moist processes included, increasing temperature while holding baroclinicity constant lowered the minimum central SLP attained by the cyclone, except for the warmest (308 K) experiments, and increased the cyclone’s maximum value of eddy kinetic energy (EKE), up to some temperature threshold. On the other hand, increasing baroclinicity alone lowered the cyclone’s minimum central pressure and strengthened the cyclone faster but also caused the cyclone to decay earlier. These results were similar when the researchers varied wind profile and temperature together in the bivariate tests. It was found that the lowest SLP and highest EKE occurred at the highest wind shear (or baroclinicity) but not at the highest temperature. They also noticed that the higher temperature simulations developed a stronger negative upper level PV anomaly. Additionally, at higher temperatures and greater baroclinicity, the cyclone’s size decreased. These results provide a possible clue as to why at least one storm from Marciano et al. (2015) failed to produce a lower SLP minimum in a warmer climate.

In a similar setup, Kirshbaum et al. (2018) experimented with both “dry” (relative humidity = 0) and “moist” conditions in differing atmospheric temperatures, using WRF version 3.7. However, Kirshbaum et al. (2018) also tested a single PV perturbation (“ISO”) and a periodic wave train (“PER,” represented by 3 PV perturbations). They set up their initial conditions in a Petterssen type B cyclogenesis (Petterssen and Smebye 1971) configuration. For the moist ISO simulations, a large increase of model convective
precipitation and smaller increase of grid-scale (microphysics) precipitation occurred as the mean environmental temperature was increased. Comparing the minimum SLP perturbation relative to the zonally averaged initial value, all the moist ISO simulations had more negative peak values than their corresponding dry ISO simulations, but the warmer simulations were less negative than the colder simulations. Another result was that peak values of EKE decreased for the warmer simulations in conjunction with a decrease in the efficiency of conversion from available potential energy (APE) to EKE. For the periodic wave train simulations, the moist simulations exhibited the same features, but with more extreme differences with varying temperatures. Moreover, possibly because of diabatic heating along the warm front, the primary area of dry “baroclinic” perturbation vertical velocity ($\omega^B$) was found to shift eastward, such that it went out of phase with the perturbation virtual temperature, leading to an overall weakening of the “baroclinic” contribution to quasi-geostrophic vertical velocity (QG-\omega). These results again show that warmer temperatures may not necessarily correspond with a lowering of minimum SLP in an extratropical cyclone.

One particularly interesting outcome in a warming climate from these studies is the increasing influence of diabatic heating and lower tropospheric variables on extratropical cyclone development. One important feature is the diabatic Rossby wave (DRW) or diabatic Rossby vortex (DRV). Moore et al. (2008) describes a diabatic Rossby vortex “as an isolated low-level vortex present in the vicinity of a surface frontal zone. Its continued existence is regarded as a synergetic interaction, whereby the vortex contributes to ascent on the frontal slope, which results in condensation and the diabatic production of potential vorticity (PV).” Previous studies have connected DRVs to
extratropical warm-core cyclones (e.g. Moore and Montgomery 2005). Additionally, the process in which a DRV encounters a favorable upper-level trough and then leads to explosive cyclogenesis bears resemblance to the process in which tropical cyclones undergo extratropical transition and then strengthen (Boettcher and Wernli 2011). Frequently, lower tropospheric Ertel potential vorticity (EPV) can be a proxy for diabatically driven PV. Marciano et al. (2015) observed a strengthening of the 900-750 hPa EPV in connection with greater latent heat release in warmer cyclones. Tierney et al. (2018) also found greater latent heat release in a warmer environment and stated, “potential vorticity analysis revealed the existence of a diabatic Rossby vortex in the earliest stages of ETC cyclogenesis in the warmest simulations.” In analyzing EKE, Kirshbaum et al. (2018) concluded, “increased latent heat release, which is supported in warmer environments with larger moisture content, does not necessarily energize the baroclinic waves.” Consequently, these variables may play an increasingly important role in the formation of extratropical cyclones in a warming world.

Other studies have also assessed the development and influence of diabatic Rossby vortices. Moore et al. (2008) performed 6 simulations of an East Coast U.S. snowstorm that lasted from February 24 to February 25 in 2005, using analysis data from the European Centre for Medium-Range Weather Forecasts (ECMWF) for the initial and boundary conditions. This storm is interesting because secondary cyclogenesis occurred at the location of a lower tropospheric PV structure that matched the qualifications of a DRV. With latent heat release and surface fluxes disabled in the model (NLNF case), the storm essentially failed to deepen. With no latent heat release (NL case), the storm did strengthen somewhat but did not result in secondary cyclogenesis. Consequently, the
storm structure and track were different from the real case. With no surface fluxes (NF case), the storm deepened very little but did follow the real case more closely in structure and track. Only the control case with both latent heat release and surface fluxes enabled the storm to closely follow the ECMWF analysis storm track.

Next, since the storm featured both a positive PV anomaly near the tropopause and a low-level positive PV anomaly, Moore et al. (2008) completed 2 additional experiments: one with the upper-level PV structure removed (UL case) and one with the low-level PV structure removed (LL case). Without the upper-level structure, changes to the 700 hPa flow (the estimated steering level) led to a farther southeast track for the storm. Without the low-level structure, the storm’s track and strength was completely changed. Therefore, even though the hypothesized DRV was critical to the storm’s development, the storm still required the upper-level features to fully develop. Nevertheless, the removal of either latent heat release or the low-level PV structure significantly altered the resulting storm’s track and evolution.

In a different approach, O’Gorman et al. (2018) investigated various climate conditions on an aquaplanet GCM. They focused on the skewness of \(-\omega\) (comparing the strength of upward vertical motion to downward motion, typically positive) and on the asymmetry parameter \(\lambda\) as they adjusted both surface air temperature and longwave optical thickness (a proxy for greenhouse gas concentration). In these fully nonlinear simulations, in most of the warmer climates, both the skewness and the asymmetry parameter increased. Then, “using the mean state from the fully nonlinear simulations as the basic state,” the researchers analyzed the most unstable mode of moist baroclinic instability. The coldest cases had a smaller skewness for the most unstable mode,
possibly due to smaller Rossby number, compared to the fully nonlinear simulations (gradient wind balance dictates that at higher Rossby number, asymmetry between cyclones and anticyclones should increase, implying greater skewness). Between a mean temperature of 300 K and 307 K, both skewness and the asymmetry parameter for the most unstable mode increased dramatically, with the warmest case (311 K) featuring an isolated diabatic Rossby vortex instead of wavelike features. In a PV analysis of that case, researchers also found that latent heating occurred only on the western side of the upper-level PV anomaly and on the eastern side of the low-level PV anomaly. Since the fully nonlinear simulations and the most unstable modes produced different results of skewness with warming, researchers further experimented with simplified convection by reducing dry static stability by a prescribed factor $r$, with the simulations no longer being fully linear but only macroturbulent. The results suggested that the different results were caused mainly by nonlinear equilibrium. Additionally, in comparing the asymmetry parameter for different values of $r$, “$\lambda$ approaches 1 for small $r$, broadly consistent with the behaviour of the unstable modes. However, for the macroturbulent state, $\lambda$ clearly saturates as a value much smaller than 1 for $r < 0.2$.” O’Gorman et al. (2018) concluded that the most unstable mode was an overestimate of the skewness of $-\omega$ for very warm climates.

Finally, researchers have continued to build upon the results of Pettersen and Smebye (1971) in classifying extratropical cyclones. Graf et al. (2017) attempted to use various atmospheric parameters during cyclogenesis to classify the storms. Having looked through 16,934 cases from 2000 through 2011, they established 30 “precursors,” normalized the values using 30-day running means and standard deviations, and then
projected the normalized values onto a two-dimensional Principal Component (PC) space. PC1 was primarily related to lower tropospheric variables, such as latent heat and strong baroclinicity. PC2 was strongly linked to upper tropospheric variables. Depending on where a given storm lay on the space, it could be classified as one of 5 types: one median class (M) and four edge classes — \( A_{\text{moist}} \), \( A_{\text{dry}} \), \( B_{\text{moist}} \), and \( B_{\text{dry}} \) (Figure 1). Of these classes, both \( A_{\text{moist}} \) and \( B_{\text{moist}} \) featured high 6-hourly precipitation anomalies. Both also
included upper level PV and QG-ω, but these variables were more important for B_{moist}. The low-level PV and upper-level PV were also almost superimposed for B_{moist}. On the other hand, A_{moist} had very significant low-level PV and QG-ω. A_{moist} also had much higher values of precipitable water (Q_{int}) and 850-hPa frontogenesis than B_{moist}. A_{moist} cyclogenesis also tended to occur near the jet streak, while B_{moist} cyclogenesis was more preferred near the left exit region. The greater importance of lower tropospheric variables for A_{moist} points to more influence of diabatic processes and suggests possible involvement of DRVs.

There are several noteworthy similarities to Pettersen and Smebye (1971). A_{moist} is comparable to Pettersen type A cyclogenesis. In describing type A, Graf et al. (2017) notes, “type A cyclones develop first at low levels without a clear upper-level disturbance at the time of genesis. Such a setup is typical for so-called diabatic Rossby waves.” A_{moist} is most common on eastern sides of continents, especially near the North Pacific and the North Atlantic oceans, consistent with Boettcher and Wernli (2013) on DRW climatology. Next, B_{dry} is similar to Pettersen type B, while B_{moist} is similar to type C cyclogenesis (Deveson et al. 2002). The remaining case, A_{dry}, corresponds with lee cyclogenesis. One observation from Graf et al. (2017) is that storms do not fall very distinctly into each category but lie in a “continuum.” Given these results, such a classification method may be useful for measuring how the structure of extratropical cyclones changes as global temperatures change. Since warmer temperatures can affect latent heat release and other diabatic heating changes, this method of classification may show how a cyclone changes from one class to another due to climate change.
Thus, from these studies, diabatic processes, such as latent heating, play an increasingly paramount role in a warming climate. Processes occurring in the lower troposphere also seem to be critical in warm environments. Some of these influences may be tied to the formation of diabatic Rossby vortices and their influence in extratropical storms. Additionally, while storms generally develop faster, as seen through minimum SLP or peak EKE, they are not necessarily stronger, because not all the energy is directed into strengthening the storm itself. Following both Marciano et al. (2015) and Kirshbaum et al. (2018), it is possible that the conversion of APE into EKE becomes less efficient in the warm environment as a consequence of weaker warm temperature anomalies, weakening the thermally direct circulation. Finally, even with the importance of lower tropospheric variables, storms still require upper level influences to fully develop.

However, since many of these studies investigated idealized cases, it is meaningful to consider whether these results would apply to a real scenario. The outlier from Marciano et al. (2015) suggests that there might not be an absolute temperature threshold above which an extratropical cyclone fails to deepen further. Rather, such a threshold may depend on a given cyclone’s existing conditions. Likewise, as diabatic Rossby vortices already exist in today’s climate, the extent to which rising temperatures stop increasing the strength of an extratropical cyclone may also depend on the prior existence of the DRV. Moreover, while the types of cyclogenesis are characterized by different low-level and upper-level forcing, the aforementioned studies did not examine whether changes in these parameters from rising temperatures resulted in changes to the type of cyclogenesis.
Hence, I decided to perform a pseudo-global warming (PGW) experiment on the 2010 Boxing Day blizzard in the eastern United States. There were 5 simulations: a control run (no temperature change), a –5 K run, a +1 K run, a +5 K run, and a +10 K run. For each modified run, I adjusted the air temperature, soil temperature, and sea-surface temperature and kept all the other variables constant. I also attempted a +11 K run, but the settings violated the CFL criterion, so it was not considered for further analysis.
SYNOPTIC OVERVIEW

To more clearly determine what parameters to analyze, I first investigated the observed development of the cyclone. The Boxing Day blizzard followed a typical Miller A storm track (Miller 1946). The storm formed in the northern Gulf of Mexico, crossed the Florida Peninsula, and continued northeast, following the coastline. The storm achieved a minimum central pressure of 960 hPa at 21Z on December 27. Precipitation for this storm was abundant, resulting in snowfall over 60 cm in parts of New Jersey, New York, and Massachusetts (HPC 2010).

Figure 2: 300 hPa station plots at Dec. 25, 0Z. Heights (solid, dam), Temperatures (thick dashed, °C), and Isotachs (thin dashed, kt) are from the NAM analysis.
Source: CSU (http://archive.atmos.colostate.edu/)

At 0Z on December 25, there was a strong ridge over the Rockies at 300 hPa and a trough over the Mississippi River Valley (Figure 2), consistent with the typical pattern leading up to East Coast snowstorms (Kocin and Uccellini 2004). On the east side of the
ridge was a jet streak, with winds around 115 kt (59 m s⁻¹) in the northwest corner of South Dakota, digging south into the Gulf Coast. A weaker jet streak with winds around
85 kt (44 m s⁻¹) remained off the Texas coast, putting Louisiana in its left-exit region. The setup at 500 hPa was similar (figure not shown), with the main trough over the same area and the base of a shortwave trough in central Texas. At 850 hPa, a 146-dam low was over Missouri, with a trough extending south/southwest into eastern Texas (Figure 3a). A strong, vertically stacked cyclone was located east of Newfoundland, near 50°N, 50°W, another feature often seen leading up to notable snowstorms in the East (Kocin and Uccellini 2004). These upper air components formed the framework to support the formation of a significant East Coast storm. At the surface, a cold front followed an inverted trough along the Texas coastline through Louisiana into Mississippi (Figure 3b). By 3Z (figure not shown), this front had moved partly offshore, acting as a coastal front. A local SLP minimum of 1014 hPa had also appeared on the front near Houston, marking the early stages of cyclogenesis. This SLP minimum tracked eastward along the Gulf Coast in tandem with the frontal boundary until 9Z, at which point it became fully separated from the original frontal boundary.

**Figure 4:** As in (Figure 3b), but at Dec. 12, 12Z. Source: CSU (http://archive.atmos.colostate.edu/)

At 12Z on December 25 (figures not shown), the 500-hPa low had not yet closed off but was over the Ontario Peninsula (near Lake Erie), with the axis of the trough now extending south to Louisiana into the former shortwave trough. At 850 hPa, the axis of a
positively tilted trough stretched approximately from Houston to New York City and appeared to extend from the aforementioned vertically stacked cyclone still slightly east of Newfoundland. At the surface, the low pressure was just south of the Mississippi River Delta and had dropped to 1013 hPa (Figure 4). In response to the upper level trough, in the next 12 hours, the surface cyclone continued to follow the northern Gulf Coast while deepening slowly. The warm front also began to develop on the northeast side by 18Z. These developments eventually led to increases in precipitation associated with the storm, especially frozen. Light snow was reported in northern Mississippi and Alabama at 12Z, and snowfall gradually expanded over the southern Appalachians into North Carolina.

Figure 5: As in (Figure 2), but at Dec. 26, 0Z.
Source: CSU (http://archive.atmos.colostate.edu/)
At 0Z on December 26, a strong jet streak at 300 hPa with winds of 125 kt (64 m s\(^{-1}\)) over Minnesota separated a strong ridge over the High Plains and Rockies from the
trough deeply entrenched in the eastern half of the United States (Figure 5). On the eastern side of the trough, a weaker jet streak over the Southeast put southern Georgia, Alabama, and the Florida panhandle in its right-entrance region, favoring ascent there (Lackmann 2011). The southern-stream jet remained over the northern Gulf of Mexico and crossed south Florida. The 500 hPa pattern (figure not shown) again resembled the 300 hPa pattern, with the trough axis over Lake Michigan to Mississippi. The 140 dam 850-hPa low had also closed off over northwest South Carolina, with the trough axis lying along the eastern side of the Appalachian Mountains (Figure 6a). Strong cold air advection was occurring to the southwest of the low, with northerly winds up to 30 kt (15 m s⁻¹), and warm air advection was occurring to the east, further strengthening the low-level baroclinic zone and aiding the cyclone’s development. At the surface, the low was now over Apalachee Bay south of Tallahassee, Florida, with a minimum SLP of 1004 hPa (Figure 6b). The surface warm front cut across the Florida Peninsula and paralleled the Southeast coast offshore, up to Cape Hatteras. The surface low subsequently followed this front, crossing the Florida Peninsula by 6Z and deepening rapidly afterwards. At 0Z, snow was now being reported in much of northern and central Alabama, the eastern half of Tennessee, southern Kentucky, southwest Virginia, western North and South Carolina, and northwest Georgia. In the next 12 hours, snow continued to push north and east, covering much of Virginia and North Carolina.

At 12Z on December 26, the low at 500 hPa had closed off over eastern Tennessee, with a minimum height of 531 dam (Figure 7a). Strong winds up to 100 kt (51 m s⁻¹) were moving through the base of the low with weak winds near the center, implying strong cyclonic vorticity. At 850 hPa (figure not shown), a 131-dam low was
located just offshore the North Carolina coast. At the surface, the low had deepened to 992 hPa (Figure 7b) and was slightly southeast of the 850h-hPa low. All these features indicated rapid strengthening of the storm, and precipitation continued to intensify. Some light snow lingered west of the Appalachians, while steady snow covered southern Virginia, most of North Carolina, and parts of South Carolina. Snow had also begun to fall on the Delmarva Peninsula. The snow moved into southern New Jersey by 15Z and into the New York City region by 18Z. At 21Z, heavy snow was reported in a region from Philadelphia through much of New Jersey into western Long Island.

At 0Z on December 27, there was a closed low at 300 hPa over western Virginia (Figure 8a). To its east was a strong jet streak with winds over 130 kt (67 m s⁻¹) off the Southeast coast and a smaller but equally strong jet streak in northeast Maine. Such a “kissing jet” setup, in which a given region lies within both the left-exit region of one jet streak and the right-entrance region of another jet streak, is favorable for enhanced precipitation over southern New England due to upper-level divergence (Kocin and Uccellini 2004). At 500 hPa (figure not shown), a broad low with heights below 528 dam
was over the central Mid-Atlantic coast. At 850 hPa (figure not shown), the 117-dam low was near 39°N, 72°W (east of Wilmington, Delaware, and south of Rhode Island). The surface low, located in nearly the same place, had rapidly deepened to 972 hPa (Figure 8b), a 32-hPa drop in the past 24 hours. Snow now covered an area north and west of the surface cyclone, stretching from coastal North Carolina, north through eastern Pennsylvania to Albany, and east through southern New England into Portland, Maine, with additional snow showers west of the Appalachians in West Virginia and Kentucky. Strong surface winds over 35 kt (18 m s⁻¹) were also found on the northwest side of the cyclone. The cyclone continued slowly north-northeast, passing slightly west of the “40-70 benchmark” (40°N, 70°W, which many storms that directly affect this region cross; Roller et al. 2016). Snowfall also persisted in much of the coastal Northeast as the strong storm, still fueled by favorable conditions, continued to produce abundant precipitation and strong winds.
At 12Z on December 27, the cyclone was now vertically stacked from the surface up to 500 hPa (Figure 9a), a sign of a mature cyclone, with the center just south of Cape Cod. Cold air advection was wrapping around the 106-dam low at 850 hPa (Figure 9b). The surface low also continued to deepen to 962 hPa (Figure 9c). Snow was reported from eastern New York to central Maine. Strong surface winds persisted north and west of the cyclone, with blowing snow reported in Philadelphia. The cyclone moved slowly toward Nova Scotia, achieving a minimum SLP of 960 hPa at 21Z, while a new low-pressure center was forming at the triple point of the frontal boundaries to the northeast of the parent low (figure not shown).

Figure 9: (a) 500 hPa, (b) 850 hPa, and (c) Surface analyses at Dec. 27, 12Z. Source: CSU (http://archive.atmos.colostate.edu/)
MODEL SETUP AND METHODS

Based on the analysis and on Marciano et al. (2015), WRF version 3.8.1 was configured with a nested domain (Figure 10) inside a larger North American domain. The larger domain spanned 253x178x50 gridpoints in the zonal, meridional, and vertical directions, respectively, at 36 km resolution; the smaller domain was 415x316x50 gridpoints at 12 km resolution. ERA Interim reanalysis (ECMWF 2009) was used for the initial and boundary conditions, including SSTs for the entire length of each run. The WRF Single-moment 5-class (WSM5) scheme was used for microphysics parameterization, the Betts-Miller-Janjic (BMJ) scheme for cumulus parameterization, and the Mellor-Yamada-Janjic (MYJ) scheme for the planetary boundary layer. The Dudhia shortwave scheme and the Rapid Radiative Transfer Model (RRTM) longwave scheme were used for radiation. These parameterizations are the same as the ones used in Marciano et al. (2015). Rather than starting the model at 3Z on December 25 (the starting time in Marciano et al., 2015), since ERA Interim is available only at 6-hour intervals, I ran the model for 90 forecast hours from 0Z on December 25 to 18Z on December 28. To track the location of the cyclone, I first calculated the SLP field for the 12-km domain within an area restricted to near the Texas coast. This constraint was necessary to avoid excessive processing time. Upon identifying the minimum SLP at 0Z on December 25, I searched the period 3 hours later for a new minimum (with the same SLP calculation) within 15 gridpoints to the west, 51 gridpoints to the north, 17 gridpoints to the south, and 79 gridpoints to the east of the former minimum. This box was initially determined from the movement of the observed cyclone; if the SLP field
within the box appeared to miss the center, the box was then expanded at all forecast intervals.

One of the key components of my analysis was the classification method developed by Graf et al. (2017) to label each scenario. To compare with their results, data were interpolated from the 12-km grid to a 1-degree latitude/longitude grid (Figure 10), using bilinear interpolation (Zender 2019). Using the new grid, an SLP field was computed for the entire domain (with the coarser grid, it is less time consuming to calculate the full SLP field) at 3-hour intervals, and a similar strategy was utilized to locate SLP minima at each interval. Because the Lagrangian precursors, which represent backward trajectories of certain variables, were not significant (Sprenger 2019, personal communication), I only computed 24 of the original 30 precursors (Table 1) from Graf et al. (2017) for my case. Most precursors were calculated with the MetPy library (May et al. 2017) for Python (see Appendix). In order to calculate QG-ω for both upper-level forcing and lower-level forcing, I used the same method in Deveson et al. (2002) that
Graf et al. (2017) used, modified for a sub-global domain. The running averages and standard deviations for all the precursors, calculated from ERA-Interim, were provided by the authors of Graf et al. (2017), and these fields were used to normalize all the precursors. The running averages for 500-hPa height, 850-hPa temperature and equivalent potential temperature, and surface skin temperature were modified for the different runs to account for the different base temperature. Following Graf et al. (2017), only the values within 500 km of the cyclone center went into the calculation of each of the normalized precursor values. I then took the mean of these values either above or below the median value, depending on the sign of the “signal” (Table 1).

In order to avoid including tropical cyclones, Graf et al. (2017) ignored cyclogenesis events south of 30°N. However, using their algorithm for cyclogenesis, the Boxing Day blizzard should have formed a little south of 30°N. Additionally, by the time the storm moved north of 30°N, it would already have been approaching the mature stage. Therefore, I performed the PC analysis twice for each simulation: once at forecast hour 12 (more consistent with “cyclogenesis”; I could not do an analysis at an earlier time since the center would have been too close to the model boundary) and once at forecast hour 24 or 30 (more consistent with the 30°N rule). For either condition, the storm had already exhibited at least one closed SLP contour at 0.5 hPa intervals in all the simulations.

Using this classification method, my goal was to see whether the cyclone type changed with warming. Since I expected low-level forcing to increase with temperature, I hypothesized that the warmest simulations would most closely match the $A_{\text{moist}}$ classification. I also wanted to reproduce the change in SLP track and strength that
existed in Marciano et al. (2015). In addition, through a cross-section analysis and a low-level PV horizontal analysis, I hoped to identify whether a diabatic Rossby vortex played a role in the development of this storm in any of the temperature scenarios. As before, I hypothesized that a diabatic Rossby vortex would exist in the warmer scenarios, with the storm developing in a different regime in the warm scenario compared to the cold scenario. Finally, I also calculated EKE for each simulation. I took zonal means of the zonal wind and meridional wind from ERA-Interim for the 90 hours, subtracted the

<table>
<thead>
<tr>
<th>Precursor Name</th>
<th>Description</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PV_{up}$</td>
<td>Upper-level PV (averaged between 600 and 200 hPa)</td>
<td>+</td>
</tr>
<tr>
<td>$PV_{low}$</td>
<td>Lower-level PV (averaged between 1000 and 600 hPa)</td>
<td>+</td>
</tr>
<tr>
<td>$EADY_D$</td>
<td>Eady growth rate in lower troposphere</td>
<td>+</td>
</tr>
<tr>
<td>$EADY_U$</td>
<td>Eady growth rate in upper troposphere</td>
<td>+</td>
</tr>
<tr>
<td>$N^2_{TROPO}$</td>
<td>Tropospheric static stability</td>
<td>–</td>
</tr>
<tr>
<td>$FGEN_{850}$</td>
<td>Pettersen frontogenesis function at 850 hPa</td>
<td>+</td>
</tr>
<tr>
<td>DEF</td>
<td>Environmental deformation at 850 hPa</td>
<td>+</td>
</tr>
<tr>
<td>$Qint$</td>
<td>Vertically integrated water vapour</td>
<td>+</td>
</tr>
<tr>
<td>$ML_CAPE$</td>
<td>Mixed-layer CAPE</td>
<td>+</td>
</tr>
<tr>
<td>$RTOT$</td>
<td>Surface precipitation during 6 h before cyclogenesis</td>
<td>+</td>
</tr>
<tr>
<td>SKT</td>
<td>Skin temperature</td>
<td>+</td>
</tr>
<tr>
<td>SLHF</td>
<td>Latent heat flux at surface</td>
<td>+</td>
</tr>
<tr>
<td>SSHF</td>
<td>Sensible heat flux at surface</td>
<td>+</td>
</tr>
<tr>
<td>$THDIFF$</td>
<td>Difference of pot. temperature between surface and 700 hPa</td>
<td>–</td>
</tr>
<tr>
<td>$T_{PERT}$</td>
<td>Anomaly of the 850 hPa temperature</td>
<td>+</td>
</tr>
<tr>
<td>$QG_{850BOT3}$</td>
<td>Low-level contribution to QG vertical motion</td>
<td>–</td>
</tr>
<tr>
<td>$QG_{850TOP3}$</td>
<td>Upper-level contribution to QG vertical motion</td>
<td>–</td>
</tr>
<tr>
<td>$Z_{ANOM}$</td>
<td>Anomaly of geopotential height at 500 hPa</td>
<td>–</td>
</tr>
<tr>
<td>$Z_{ORIG}$</td>
<td>Geopotential height at 500 hPa</td>
<td>–</td>
</tr>
<tr>
<td>$\n\theta_{850}$</td>
<td>Horizontal temperature gradient at 850 hPa</td>
<td>+</td>
</tr>
<tr>
<td>$\n\theta_{e850}$</td>
<td>Horizontal gradient of equivalent pot. temperature at 850 hPa</td>
<td>+</td>
</tr>
<tr>
<td>$\theta_{e850}$</td>
<td>Equivalent pot. temperature at 850 hPa</td>
<td>+</td>
</tr>
<tr>
<td>$TADV850$</td>
<td>Horizontal temperature advection at 850 hPa</td>
<td>+</td>
</tr>
<tr>
<td>VELJET</td>
<td>Wind speed averaged between 500 and 100 hPa</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 1: List of the 24 precursors and their descriptions. The descriptions are taken directly from Graf et al. (2017) “Signal” represents whether the mean above the median (+) or below the median (–) is calculated for the cyclogenesis area.
values from the 1-degree interpolated values to get the eddy winds, and then took a volume integral over the 1-degree interpolated domain. Since ERA-Interim is 6-hourly data, EKE was only computed at those time intervals.

Figure 11: Storm tracks of all the simulations. A “+” marks the cyclone center every 3 forecast hours.
RESULTS

(a) Control Run

The control run storm track (Figure 11, black line) generally agrees with the WPC track analysis. In the simulation, the storm began to deepen between forecast hour 9 and 12 (9Z to 12Z on December 25), with steady strengthening occurring after forecast hour 24, once the storm had reached the Atlantic. Then, the storm continued to Cape Cod, also passing just west of the “40-70 benchmark” as in the observed track, before turning more eastward and moving past Nova Scotia. The storm achieved a minimum central pressure of 957 hPa at forecast hour 90 and a peak column-averaged EKE of 2.98x10^6 J m^-2 at forecast hour 66. Total precipitation exceeded 35 mm in southern New England and portions of the Canadian Maritimes.

First, using the Graf et al. (2017) parameters for cyclogenesis classification, at forecast hour 12, the storm features high values of 1000-600 hPa PV (Figure 12a) and 600-200 hPa PV (Figure 12b) to the northwest of the surface cyclone center. The highest 6-hourly precipitation amounts (Figure 12j) are in a similar area as the low-level PV, and precipitable water (Figure 12h) is abundant surrounding the cyclone center. These results match the analysis in Graf et al. (2017) for A_{moist} and support the idea that low-level PV is a result of latent heat release. The highest values of negative QG-ω (Figure 12p,q) lie along and north of the center. Positive 850-hPa frontogenesis (Figure 12f) is generally displaced to the north over land, with another local maximum closer to the center. An area of positive 850-hPa warm-air advection (Figure 12w) exists near and east of the surface center.
To better diagnose the PV structure of the storm, cross-sections through the center are taken at forecast hours 12, 24, 36, and 48 (Figures 13 to 16). Frontogenesis at 850 hPa (Figure 17) is also investigated at those intervals. Agreeing with the analysis on the coarser grid, at forecast hour 12 (12Z on the 25th, Figure 13), there are high values of
PV (over 1.5 PVU) from 700 hPa to 450 hPa northwest of the cyclone center. There is also an area of strong low-level PV (600 hPa downwards), detached from the upper-level PV, over the center. This result is a little different from Moore et al. (2008), in which the strong low-level PV remained farther away from the strong upper-level PV in the early stages of their storm. Also consistent with the previous analysis, 850-hPa frontogenesis lies primarily to the north of the surface cyclone (Figure 17a).

Figure 13: A cross section through the surface cyclone center, showing potential temperature (solid contours every 5 K), potential vorticity (PVU; shaded according to legend; the green/yellow boundary at 1.5 PVU marks the dynamic tropopause), and winds (dashed every 5 m s$^{-1}$; warm colors represent into the page; cold colors represent out of the page). The inset shows 700 hPa heights (solid contours 30 m) and the location of the cross section.

At forecast hour 24 (0Z on the 26$^{th}$, Figure 14), the storm has now moved onto the Florida panhandle, a little earlier than observations. Strong low-level PV near the cyclone center persists (Figure 14), while there are additional areas of strong mid-level PV west of the cyclone. The area of strong low-level PV is also starting to interact more closely with the stratospheric PV, but the dynamic tropopause (marked by the 1.5 PVU
contour in the upper troposphere) sits above 350 hPa. Frontogenesis at 850 hPa stretches from southwest to northeast in 2 bands, one running through the surface center and one to the northwest (Figure 17b).

Figure 14: As in (Figure 13), but at forecast hour 24.

Figure 15: As in (Figure 13), but at forecast hour 36.
At forecast hour 36 (12Z on the 26th), the storm is just east of Cape Hatteras, NC, and the pressure has dropped below 990 hPa. A vertically-oriented block of strong low-level PV persists near the cyclone center, while a partly detached branch tilts westward into the area of strong upper-level PV but remains on the warm side of the frontal boundary. The dynamic tropopause remains above 500 hPa, however. A strong southerly low-level jet sits just east of the low-level PV “tower.” At 850 hPa (Figure 17c), the low has closed off over Cape Hatteras, while areas of strong frontolysis have developed adjacent to areas of frontogenesis. The strongest frontogenesis is now northeast of both the surface and 850-hPa low. Along with the 700 hPa inset in Figure 15, it is evident that the storm is beginning to become vertically stacked. This process appears to happen slightly earlier compared to actual observations.

Figure 16: As in (Figure 13), but at forecast hour 48.

At forecast hour 48 (0Z on the 27th), the storm is located east of New Jersey. Upper-level PV over 1.5 PVU now extends directly into the lower troposphere, meeting
the area of strong low-level PV over the cyclone. The 850-hPa low is in approximately the same position, with strong frontogenesis wrapping around north and east of the low (Figure 17d). Frontolysis remains adjacent to areas of frontogenesis, with a prominent region of frontolysis in the vicinity of the low. Based on the areas of frontogenesis, the cyclone has begun to take on the appearance of the “bent-back warm front and frontal T-bone” structure described in the Shapiro-Keyser cyclone model (Shapiro and Keyser 1990).

Figure 17: Map of the 850 hPa heights (brown contours every 30 m), frontogenesis (K 100 km\(^{-1}\) 3 hr\(^{-1}\); shaded according to legend), and winds (half-barb is 2.5 m s\(^{-1}\), full barb is 5 m s\(^{-1}\), pennant is 25 m s\(^{-1}\)) at forecast hours (a) 12, (b) 24, (c) 36, and (d) 48 for the control.

(b) Classification of the cases

Figure 18 shows the location of each simulation on the Graf et al. (2017) PC space. At forecast hour 12, the –5 K simulation starts out with comparatively weaker lower level forcing (farther to the left on PC1), with the +5 K simulation the farthest to
Intriguingly, all the simulations fit well within the region of $A_{\text{moist}}$, with the $-5$ K simulation having the most negative value for PC2. At forecast hour 24 to 30 (Figure 19), the situation changes. All the simulations would still be classified as $A_{\text{moist}}$ cyclogenesis; however, they are significantly more extreme, especially on PC1, further supporting the idea that the storm has developed beyond its cyclogenesis phase by this time. Additionally, the value of PC1 now generally decreases with increasing time.
These results suggest that upper level forcing plays an increasingly important role as this storm continues to develop but less of a role early on. Nevertheless, the location of past 6-hour precipitation remains in approximately the same area as the region of strongest low-level PV (figure not shown).

The individual values and spatial distribution of all the parameters for each non-control case at forecast hour 12 are given in the Appendix. First, many variables increase steadily from the coldest case to the warmest case. Precipitable water near the cyclone
is much higher in the +10 K simulation than in the −5 K simulation. Similarly, for the warmest 2 cases, there is significantly more mixed-layer CAPE near the cyclone center compared to the other runs, while the coldest run has very little CAPE in the cyclone’s vicinity. Nevertheless, some of the temperature and baroclinicity precursors feature a counterintuitive shift. Although both the actual temperature and the mean temperature are increased, the warm temperature anomaly at 850 hPa near the cyclone center weakens in the warmer cases, while the cold surface skin temperature anomaly strengthens. A possible cause for this change is the presence of cold air advection near the center in the warmer runs. This decrease in warm anomalies also corresponds to the decrease found in Marciano et al. (2015). For baroclinicity, Eady growth rate also weakens for the warm simulations, while the magnitude of the horizontal 850 hPa $\nabla \theta$
weakens only for the warmest 2 runs.

Other variables for the same timeframe feature a non-monotonic trend with increasing temperature along with an intriguing spatial distribution. All the runs have significant values of 1000-600 hPa PV and 600-200 hPa PV. At both layers, the highest values of PV are to the west-northwest of the cyclone center. Both past 6-hourly rainfall and mixed-layer CAPE increase steadily with warming temperatures; however, the precipitation in the +10 K case is farther removed from the cyclone center compared to the other cases and leads to a reduction in the precursor value. In all the runs, the area of highest precipitation matches best with the regions of high upper-level PV. However, the highest low-level PV also occurs in the same vicinity, and it is inconclusive from these plots alone whether the precipitation is more closely tied to upper-level PV or low-level PV. The existence of strong upper-level PV near the storm represents a slight deviation
from previous research on DRVs, though a possible reason may be that cyclogenesis more accurately occurred slightly earlier in the run.

Two other variables with similar trends are frontogenesis and temperature advection. The strongest 850 hPa frontogenesis lies to the northwest of the center for most simulations. However, the local maximum begins to shift to the northeast side in the warmest 2 cases. Again, the strongest frontogenesis is farther removed from the
cyclone center for the +10 K case. Warm air advection at 850 hPa also strengthens up to the +5 K case. For the +10 K case, a weaker area of warm air advection exists north of the cyclone, while cold air advection has begun to appear near the center. These results are comparable to the findings in Kirshbaum et al. (2018), which featured an eastward movement of the low-level PV, the southerly flow of high-$\theta_e$ air, and frontal clouds in the warm simulations. Similar to that study, the frontal boundaries and the regions of temperature advection appear to have shifted eastward in the warmer scenarios.

Not all of the more prominent precursors (based on Graf et al. 2017) feature significant changes from one run to another. Both QG-$\omega$ forcing and 500 hPa heights remain at similar values throughout the cases, though the magnitudes decrease slightly in the warmest case. All of the cases have a generally weak upper-level jet in the vicinity, based on the 500-100 hPa mean wind. A summary of all the changes is given in Table 2. Figure 20: Time series of column-averaged EKE (10^5 J m^-2) for each simulation.

(c) EKE analysis

The time evolution of column-averaged EKE for all the simulations is plotted on Figure 20. Up until forecast hour 60, all the simulations experienced an increase in EKE
as the storm developed. Unlike previous studies, in which the maximum EKE decreased with warming temperatures, the +10 K run featured the highest attained EKE, at $3.11 \times 10^6$ J m$^{-2}$, while the coldest run had the lowest EKE. To account for this apparent discrepancy, EKE in only the 1000-900 hPa layer was also computed (figure not shown).

In this layer, the +10 K run indeed had the lowest EKE ($6.35 \times 10^4$ J m$^{-2}$) out of all the simulations. On the other hand, more consistent with the previous studies, the warmest simulation also plateaued the earliest and then began to decline at forecast hour 66, while the –5 K simulation plateaued about 6 hours later and then declined after forecast hour 72. Furthermore, after forecast hour 72, the warmest simulation declined most rapidly.

**Figure 21:** The search region and calculated SLP plots (solid contours every 2 hPa) for the control run at forecast hours 0, 3, 12, and then every 3 hours between f24-f60.
(d) Analysis of cyclone tracks

Changing the environmental temperature did not significantly change the track of the cyclone for most of the scenarios (Figure 11), though the higher temperature simulations did feature a slight eastward shift. However, the +10 K simulation produced a notable southeast shift in the track compared to all the other cases. This is consistent with previous studies that also showed such a shift at warmer temperatures. On the other hand, during the early part of the storm, the warmer simulations tracked farther inland (into Mississippi and Alabama), while the coldest simulations remained offshore until reaching Florida.

Figure 22: As in (Figure 21), but for the T+10K case.
Comparing the SLP field near the vicinity of the cyclone center at each 3-hour interval of the control case (Figure 21) and the +10 K simulation (Figure 22), a few interesting features emerge. The control case retains a tightly wrapped circulation (assuming geostrophic flow) as it moves just offshore from the East Coast of the United States. In the +10 K simulation, starting at forecast hour 27, the storm begins to feature multiple areas of low pressure. Especially after forecast hour 33, the minima to the east and southeast of the parent low appear to skew the track of the storm farther to the southeast. These multiple lows help to explain the sharp jumps in the track on Figure 11.

For the time evolution of the cyclone’s minimum SLP (Figure 23a), the +10 K simulation was again the outlier compared to all the runs. Except for the +10 K simulation, which had a minimum of 961.8 hPa, all the runs achieved a minimum SLP

Figure 23: Time series of (a) minimum SLP and (b) mean 850 hPa PV (taken from 25 gridpoints centered on the surface low) for each simulation.
below 960 hPa. This is a little different from Marciano et al. (2015), in which the A2 emissions scenario was sufficient to effect a change in minimum SLP. In addition, unlike previous studies, in which the warmer cyclones deepened faster but also weakened faster, the warmer cyclones in my study, especially the +10 K run, took slightly longer to deepen.

(e) Analysis of PV, Frontogenesis, and Precipitation

Following Boettcher and Wernli (2013), I also computed mean 850-hPa PV surrounding the SLP minimum for each forecast at 3 hourly intervals (see Figure 23b). Overall, there were few differences from run to run. Interestingly, for all the runs, the mean 850-hPa PV stays above 0.8 PVU for nearly all 90 hours. The +10 K run starts out with the highest PV but declined the earliest, while the −5 K run features the lowest PV for the first half of the simulation but gradually increases to levels comparable to the

Figure 24: As in (Figure 13), but for the T+10K case.
other runs. One caveat is that the point of maximum 850-hPa PV and the location of the SLP minimum do not always line up at a given forecast hour (figure not shown). Nonetheless, these results are generally consistent with my classification analysis, in which all the simulations featured strong low-level PV forcing.

To better compare the +10 K run and the control run, cross-sections (Figures 25-28) and 850-hPa frontogenesis (Figure 28) for the +10 K run are also analyzed. At forecast hour 12 (Figure 24), the dynamic tropopause already extends down to 600 hPa, and multiple branches of low-level PV exist near the surface cyclone. This is unlike the control run, in which the stratospheric PV and the strong low-level PV were not directly interacting in the early stages of the storm. At 850 hPa (Figure 28a), the setup of the trough is similar, with the 850-hPa low having yet to form. There is slightly stronger but more concentrated 850-hPa frontogenesis running through the surface low in a southwest to northeast orientation. The ridge east of Florida is also slightly stronger.

**Figure 25:** As in (Figure 14), but for the T+10K case.
At forecast hour 24 (Figure 25), the cross section for the +10 K simulation looks distinctly different from the control run cross section. For the +10 K simulation, the dynamic tropopause extends below 550 hPa in 2 areas, one west of the cyclone center and one over the cyclone. The area west of the center appears to interact more directly with the strong low-level PV over the center, while the other area is more closely associated with additional PV maxima in the middle troposphere east of the cyclone. Near the surface, there are also 2 areas of cyclonic flow, rather than a single area associated with the storm center. Again, at 850 hPa (Figure 28b), the trough is in a similar area as in the control run, and there is again a slightly stronger ridge east of Florida. In addition, frontogenesis is more concentrated in several narrow bands to the northeast of the surface low.

At forecast hour 36 (Figure 26), the cross section for the +10 K simulation is taken through an area between the multiple SLP minima. The area of strong low-level PV again

*Figure 26: As in (Figure 15), but for the T+10K case.*
tilts westward into the dynamic tropopause. Additional areas of middle tropospheric PV over 1.5 PVU exist over and east of the cyclone, with the area to the east interacting
with another branch of upper level PV. The 850-hPa low has also already developed by this time but is a little more elongated and displaced slightly southwest (Figure 28c). Frontogenesis is more concentrated in a narrower band, and a weaker north-south band to the east has begun to develop.

At forecast hour 48 (Figure 27), there are 2 distinct “towers” of PV over 1.5 PVU. One again tilts westward from the parent cyclone center, while another is east of the secondary low-pressure center. Again, near the surface, there are multiple areas of cyclonic flow. At 850 hPa (Figure 28d), the low is a little weaker and farther east. The frontogenesis structure also looks less like the Shapiro-Keyser structure, with the primary band still running southwest to northeast through the low. Consequently, the

Figure 29: Cumulative precipitation (mm, shaded according to legend). Left to right: grid-scale precipitation, convective precipitation, and sum of grid-scale and convective precipitation. Top to bottom: T-5K run, control run, and T+10K run.
strongest frontogenesis is mostly offshore rather than over Long Island and Cape Cod, as in the control run. As before, a weaker band running south to north exists east of the low, suggesting a secondary front. At the apparent triple point (based on the areas of frontogenesis), the 850-hPa height contours also suggest possible development of a new low.

Precipitation increases from the –5 K run to the +10 K run (Figure 29). The region of precipitation also shifts eastward, consistent with the results of previous studies. Additionally, convective precipitation increases significantly for the +10 K run and covers a broader area. These changes also reflect the increase in precipitable water and CAPE that were seen earlier at cyclogenesis. Consequently, despite the overall increase in precipitation, total precipitation over the areas that received heavy snowfall from the storm decreases significantly. Along with the increase in temperatures, the shift in the precipitation shield leads to a smaller area over land that experiences snow, with significantly less snowfall along the East Coast (Figure 30) in the +10 K run.

**Figure 30:** Total liquid-equivalent snowfall (mm, shaded according to legend) for each simulation.
CONCLUSIONS

A diabatic Rossby vortex likely played a role in the development of the Boxing Day Blizzard of 2010. Around cyclogenesis, there was ample strong low-level PV in the vicinity of the cyclone, on the warm side of the frontal boundary, similar to what was observed in Moore et al. (2008). Although there was also strong upper tropospheric PV not too far from the storm, it did not appear to be interacting directly with the low-level PV based on cross section analysis (Figure 13). The storm continued to track in tandem with the low-level PV structure, strengthening more rapidly once the stratospheric PV began to interact with the low-level PV. In the +10 K simulation, it also appeared that multiple pockets of low-level PV contributed to the formation of multiple low-pressure centers to the northeast and southeast of the main low. The proximity of precipitation to the areas of strong low-level PV further supports the hypothesis of the DRV’s existence and influence for this storm.

My hypothesis that low-level parameters would become more extreme in a warmer environment was shown to be incorrect for this particular case study. All of the simulations, including the –5 K simulation, featured relatively strong normalized low-level parameters, especially when compared to the storms classified in the cyclogenesis phase space of Graf et al. (2017) (Figure 1). The normalized values instead decreased slightly at the warmest temperature, suggesting that there may be a limit after which a warming environment fails to further strengthen moist processes such as latent heat release. Additionally, the normalized upper-level parameters consistently increased with warming temperatures, which would reflect a possible shift away from $A_{moist}$ (i.e. stronger low-level parameters and weaker upper-level parameters) toward $B_{moist}$ (strong
low-level and upper-level parameters) in Graf et al. (2017). Following the changes of the most unstable mode seen by O’Gorman et al. (2018), I would have assumed a sudden jump from stronger upper-level parameters to weaker upper-level parameters to reflect the change to a DRV in an extremely warm environment. Additional studies are necessary to determine whether even the DRV structure breaks down in an extremely (though perhaps unrealistically) warm climate.

On the other hand, I was able to replicate many of the results from the other studies. Both storm track and precipitation did move eastward with warming temperatures, and precipitation (especially convective) increased, just as Marciano et al. (2015) had found. In the warm environment, based on minimum SLP, the storm also failed to strengthen as much as it would have in a cooler environment, while the peak strength was attained earlier, both results consistent with Tierney et al. (2018) and Kirshbaum et al. (2018). Similarly, the warmest environment led to a reduction of warm temperature anomalies (Marciano et al. 2015), which may be associated in a breakdown of the thermally direct circulation and the conversion efficiency of APE into EKE (Kirshbaum et al. 2018).

Thus, in a warming environment, extratropical cyclones involving DRV interactions would be expected to track farther offshore (Figure 3), with more frequent cases of multiple low centers due to secondary cyclogenesis at the triple point of frontal boundaries (Figures 23, 28, 29d). These multiple centers may complicate forecasting as subtle changes in structure during the evolution of the storm lead to changes in areas favorable for upward vertical motion, which in turn lead to sudden shifts in regions of strongest precipitation as the storm moves along the coast. Overall, precipitation...
associated with these storms would increase, while total frozen precipitation from the storms would decrease (Figure 30). With the farther offshore track, some of this additional precipitation would fall over the ocean, so that areas on land would not necessarily see the increase (Figure 29). Strong winds are often associated with these cyclones, but since the minimum central pressure does not decrease (Figure 23a), there would not be a tighter pressure gradient necessary to support even stronger winds. Nevertheless, since this study only focused on one particular storm, additional research may help to determine the degree to which these results apply to different storms and environmental conditions.

One limitation of this study has been the use of 30-day running means and standard deviations provided by Graf et al. (2017) for the precursors. Since these quantities were given on a 1-degree grid, it was necessary to perform interpolation of the WRF output to match the grids and to provide a consistent comparison with Graf et al. (2017). Future work could be done to assess the sensitivity of the cyclogenesis classification to the interpolation. Additionally, it was found that the SLP fields on the 1-degree grid resulted in a slightly different track in the +5 K simulation and the +10 K simulation compared to the original 12-km grid (figure not shown), likely due to subtle differences in the location pressure field amounting to a little over 1 hPa. While previous studies also conducted sensitivity tests (Marciano et al. 2015; Kirshbaum et al. 2018), these tests were not done with respect to cyclone classification. Therefore, especially since many of the precursors involved gradients, a classification analysis done on a higher resolution grid may yield more precise results.
Another result that could be further investigated is the robustness of the shifts in cyclone track. During the original event, about 48 hours before the storm reached the Northeast United States, there had been approximately a 9-degree spread in the east-west direction for the ensemble members of the major global and regional models, including the GFS and ECMWF (HPC 2010). Based on Figure 3, there was approximately a 4-degree difference between the track of the control run and the +10 K simulation, which lies within the model spread. Consequently, experiments that expand each temperature simulation into an ensemble of runs with varying parameters and/or initial conditions would better show the expected shifts in track and the significance of such shifts.

Building on this study, several other parameters could be analyzed to determine other dynamical changes to this or other cyclones with warming. An investigation of Lagrangian parameters may show the origin of the low-level PV and provide additional information on the development of the DRV. A calculation of total APE throughout the duration of the storm could also provide additional comparisons to Kirshbaum et al. (2018) on the efficiency of conversion of APE to EKE in warmer temperatures. Moreover, since vorticity is another important quantity that affects cyclone strength, an analysis of vorticity of the atmospheric flow and vorticity advection may also be relevant.

Additionally, other experiments can be conducted on the Boxing Day Blizzard to ascertain climate change's impacts in future scenarios. One possible experiment is to use a more realistic method to modify temperatures, such as in Marciano et al. (2015), rather than an arbitrary temperature increase at all levels of the atmosphere. Another experiment to focus on a different aspect of climate change would be to modify the
wind profile, as in Tierney et al. (2018), while keeping temperature constant. These changes may help to quantify how much the wind profile affects DRV’s.

Finally, since this study only investigated one storm, it is inconclusive whether there is an absolute temperature threshold above which an extratropical cyclone fails to deepen further, or if the threshold depends on the cyclone’s environment. It is also uncertain whether the low-level precursors at cyclogenesis failed to become stronger in the warmer environment because they had already attained some upper limit given the DRV’s existence. Future work could investigate whether these results would apply to a non-outlier storm from Marciano et al. (2015), or to another storm that was already influenced in the present climate by a DRV, such as in Moore et al. (2008). These additional experiments may help to pinpoint whether the DRV’s existence in today’s climate plays a direct role in limiting how cyclones strengthen in warmer climates. For the non-outlier storms in Marciano et al. (2015), these experiments may show whether the storms acquire DRV-like features and whether there is a consistent temperature threshold at which point the DRV dominates storm development.
APPENDICES

Part 1: Python Scripts

# file: center_500.py
import sys
import numpy as np
import netCDF4 as nc
import pandas as pd
import metpy.calc as mpcalc
from metpy.interpolate import log_interpolate_1d
from metpy.units import units
import metpy.constants as constants
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap

print('done import')

def read_var(NCfile, varname, hr):
    fid = nc.Dataset(NCfile, 'r')
    var_out = fid.variables[varname][:]
    var_time = var_out[hr]
    fid.close()
    return var_time

# array averaging function similar to numpy.diff function (arr[n] + arr[n+1])
def av_ar(a, n=1, axis=-1):
    a = np.asanyarray(a)
    nd = a.ndim
    axis = np.core.multiarray.normalize_axis_index(axis, nd)
    slice1 = [slice(None)] * nd
    slice2 = [slice(None)] * nd
    slice1[axis] = slice(1, None)
    slice2[axis] = slice(None, -1)
    slice1 = tuple(slice1)
    slice2 = tuple(slice2)
    op = not_equal if a.dtype == np.bool_ else np.add
    for _ in range(n):
        a = op(a[slice1], a[slice2]) / 2.
    return a

# finding location on global grid
def combine_arr(lat1, lat2, lon1, lon2):
    lat_bool = np.in1d(lat1, lat2)
    lon_bool = np.in1d(lon1, lon2)
    return np.nonzero(lat_bool)[0][0], np.nonzero(lat_bool)[0][-1],
    np.nonzero(lon_bool)[0][0], np.nonzero(lon_bool)[0][-1]

try:
    t_adj, ym1, xm = float(sys.argv[1]), int(sys.argv[2]),
    int(sys.argv[3]) # (6, 8) for all except +10 (4, 9)
    print(t_adj, ym1, xm)
except:
    print('error: missing inputs')
sysexit()

FILE_LISTS = '/home/py84/research_new/'
if t_adj == -5.:
    ra = '105'
    f_input = FILE_LISTS+'r105_grid.nc'.format(t_adj)
    t_head = 'T-5K'
else:
    ra = '{:03}'.format(int(t_adj))
    f_input = FILE_LISTS+'r{} grid.nc'.format(ra)
    t_head = 'T+{}K'.format(int(t_adj))
print(ra, t_head)
# forecast hour
h_ind = 12
# use same size gridbox for all the runs, only modifying center marker
ym = 6

lat_tot = np.arange(-90., 91., 1.)
lon_tot = np.arange(-180., 181., 1.)
# on the regridded 1-deg data (lat is len 30, lon is len 49)
# i-10:i+11 is preferred if not out of bounds; otherwise, use :2*i+1
lats = read_var(FILE_LISTS+'r000_grid.nc', 'lat', None)[0, :2*ym+1]
lons = read_var(FILE_LISTS+'r000_grid.nc', 'lon', None)[0, :2*xm+1]
# midpoints after slicing
m1y, m1x = len(lats)//2, len(lons)//2  # use if i-10:i+11
m1y, m1x = ym, xm  # use if 0:2*i+1; ym/xm not necessarily midpoints
j1, j2, i1, i2 = combine_arr(lat_tot, lats, lon_tot, lons)

# pressure
def calc_pres(fid_name, hr):
    p0 = read_var(fid_name, 'PB', hr)[j1:j2+1, i1:i2+1] * units.Pa
    p1 = read_var(fid_name, 'P', hr)[j1:j2+1, i1:i2+1] * units.Pa
    pres_pa = p0 + p1
    return pres_pa.to('hPa')
pres = calc_pres(f_input, h_ind)
print(pres.shape)

# geopotential, staggered on edge of grid
def calc_hght(fid_name, hr):
    gvty0 = read_var(fid_name, 'PHB', hr)[j1:j2+1, i1:i2+1]
    gvty1 = read_var(fid_name, 'PH', hr)[j1:j2+1, i1:i2+1]
    geopot0 = gvty0 + gvty1  # geopotential in m2/s2
    geopot1 = av_ar(geopot0, axis=0) * units.m**2 / units.s**2
    h = mpcalc.geopotential_to_height(geopot1)
    return h
hght = calc_hght(f_input, h_ind)
print(hght.shape)
p0 = [500.] * units.hPa
h500 = log_interpolate_1d(p0, pres, hght, axis=0)[0]
del hght

# climatological means
z_climo_m = read_var('/home/py84/research_new/acm/C1225_12',
        'Z_ORIG_AV', 0)[0, j1:j2+1, i1:i2+1] * units.m
if t_adj != 0.:
    thk_adj = constants.dry_air_gas_constant * t_adj * units.K * np.log(2.) / constants.earth_gravity
    z_climo_m = z_climo_m + thk_adj.to('m')
z_anom_m = read_var('/home/py84/research_new/acm/C1225_12', 'Z_ANOM_AV', 0)[0, j1:j2+1, i1:i2+1] * units.m
# standard deviations
z_climo_sd = read_var('/home/py84/research_new/acm/C1225_12', 'Z_ORIG_STD', 0)[0, j1:j2+1, i1:i2+1] * units.m
z_anom_sd = read_var('/home/py84/research_new/acm/C1225_12', 'Z_ANOM_STD', 0)[0, j1:j2+1, i1:i2+1] * units.m

# height anomaly
z500_anom = h500 - z_climo_m
# the normalization calculations
z500_norm = (z500_anom) / z_climo_sd
z500_a_norm = (z500_anom - z_anom_m) / z_anom_sd

# variable grid spacing, using map factors
def create_deltas(fid_name, dx_in): # assume same grid spacing for both x and y directions
    m_file = read_var(fid_name, 'MAPFAC_M', 0)[:2*ym+1,:2*xm+1]
    map_x = av_ar(m_file, axis=1)
    map_y = av_ar(m_file, axis=0)
    print(map_x.shape, map_y.shape)
dx_out = dx_in / map_x
dy_out = dx_in / map_y
    return dy_out, dx_out # following same order convention as wrf output

dy, dx = create_deltas(f_input, 111000*units.m) # 1 degree approx. 111 km
scale_y = 111000*units.m # constant spacing for 1 degree
scale_x = np.mean(dx[m1y,m1x-1:m1x+1]) # estimated constant spacing

# functions to calculate radial distances
def dist_from_center(sc_y, sc_x, y_len, x_len, ymid, xmid):
    y_dist, x_dist = np.ogrid[0:y_len,0:x_len]
dist2 = np.linalg.norm((sc_y*(y_dist-ymid), sc_x*(x_dist-xmid)),axis=0)
    return dist2

# sort normalized values within specific distance from center
def norm2med(arr, sy, sx, lim, use_top = True):
    ay, ax = arr.shape
    b = arr.copy()
b = np.ma.array(b, mask=m2)
a_out = b[b.mask==False]
m_ind = len(a_out)//2
    return pc_val

z5_sort = norm2med(z500_norm, scale_y, scale_x, 500000*units.m, False)
z5a_sort = norm2med(z500_a_norm, scale_y, scale_x, 500000*units.m, False)
print(z5_sort)
print(z5a_sort)

df = pd.read_csv('precursors_case0.csv', sep=',,', header=None, names=[None, 'T+0K', 'T-5K', 'T+1K', 'T+5K', 'T+10K'], index_col=0)
df.at['nstd_Z_ORIG.p5l.0',t_head] = z5_sort
df.at['nstd_Z_ANOM.p5l.0',t_head] = z5a_sort
print(df)
df.iloc[:,df.notnull().any().values].to_csv('precursors_case0.csv', header=False) # only includes columns if at least 1 value isn't NaN

if t_adj < 5:
zl = np.arange(5400., 5901., 50.)
else:
zl = np.arange(5600., 6101., 50.)
al = np.arange(-85., 86., 5.)
al2 = np.arange(-280.,1., 20.)

def draw_map(dataset1, l, x_axis, y_axis, str_ver, cm):
    ax = plt.subplot(1, 1, 1)
m = Basemap(llcrnrlon = x_axis[0], llcrnrlat = y_axis[0], urcrnrlon = x_axis[-1], urcrnrlat = y_axis[-1], lat_ts = y_axis[ym], resolution = 'l', area_thresh = 100., projection = 'merc')
nx, ny = np.meshgrid(x_axis, y_axis)
x, y = m(nx, ny)
x_point, y_point = m(x_axis[xm], y_axis[ym1])
parallels = np.arange(10., 65., 2.)
meridians = np.arange(-125., -9., 4.)
m.drawparallels(parallels, labels = [1,0,0,0])
m.drawmeridians(meridians, labels = [0,0,0,1])
m.drawcoastlines()
m.drawcountries()
m.drawstates()
cf = plt.contourf(x, y, dataset1, l, cmap = cm, extend='both')
plt.colorbar(cf, shrink = 0.6)
m.plot(x_point, y_point, marker='$\mathrm{L}$', color='r', markersize=18)
ax.set_title(str_ver, fontsize = 11)
return ax

figw, figh = 9,7
plt.figure(figsize=[figw, figh])
map1 = draw_map(h500, zl, lons, lats, 'Heights (m)', 'plasma')
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh, top=1-0.4/figh)
plt.savefig('fin'+ra+'_05_03_z500.png', dpi=60)

plt.figure(figsize=[figw, figh])
if t_adj == 0.:
    map1 = draw_map(z500_anom, al2, lons, lats, 'Z$_{ANOM}$ from 30-day Mean (m)', 'winter')
else:
    map1 = draw_map(z500_anom, al2, lons, lats, 'Z$_{ANOM}$ from Temp-Corrected 30-day Mean (m)', 'winter')
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
                   top=1-0.4/figh)
plt.savefig('fin'+ra+'_'05_02_a500.png', dpi=60)
# file: center_def.py
import sys
import numpy as np
import netCDF4 as nc
import pandas as pd
import metpy.calc as mpcalc
import metpy.constants as constants
from metpy.interpolate import log_interpolate_1d
from metpy.units import units
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
print("done import")

def read_var(NCfile, varname, hr):
    fid = nc.Dataset(NCfile, 'r')
    var_out = fid.variables[varname][:]
    var_time = var_out[hr]
    fid.close()
    return var_time

def read_wnd(NCfile, varname, hr):
    fid = nc.Dataset(NCfile, 'r')
    var_out = fid.variables[varname][:]
    var_time = var_out[hr-12:hr+1]  # allows reading multiple hours
    print(var_time.shape)
    var_mean = np.mean(var_time, axis=0)
    fid.close()
    return var_mean * units.m / units.s

# array averaging function similar to numpy.diff function (arr[n] + 
# arr[n+1])
def av_ar(a, n=1, axis=-1):
    a = np.asanyarray(a)
    nd = a.ndim
    axis = np.core.multiarray.normalize_axis_index(axis, nd)
    slice1 = [slice(None)] * nd
    slice2 = [slice(None)] * nd
    slice1[axis] = slice(1, None)
    slice2[axis] = slice(None, -1)
    slice1 = tuple(slice1)
    slice2 = tuple(slice2)
    op = not_equal if a.dtype == np.bool_ else np.add
    for _ in range(n):
        a = op(a[slice1], a[slice2]) / 2.
    return a

# function to replace NAN's with the above value, loops through vertical
# axis
def xtrap_values(arr):
    for k in range(len(arr)-2,-1,-1):
        if np.any(np.isnan(arr[k])):
            for j in range(len(arr[k])):
                for i in range(len(arr[k,j])):
                    if np.isnan(arr[k,j,i]):
                        arr[k,j,i] = arr[k+1,j,i]
    return arr
# finding location on global grid

```python
def combine_arr(lat1, lat2, lon1, lon2):
    lat_bool = np.in1d(lat1, lat2)
    lon_bool = np.in1d(lon1, lon2)
    return np.nonzero(lat_bool)[0][0], np.nonzero(lat_bool)[0][-1], np.nonzero(lon_bool)[0][0], np.nonzero(lon_bool)[0][-1]
```

```python
try:
    t_adj, ym1, xm = float(sys.argv[1]), int(sys.argv[2]), int(sys.argv[3])  # (6, 8) for all except +10 (4, 9)
    print(t_adj, ym1, xm)
except:
    print('error: missing inputs')
sys.exit()
```

```python
FILE_LISTS = '/home/py84/research_new/'
if t_adj == -5.:
    ra = '105'
    fname_grid = FILE_LISTS+'r105_grid.nc'.format(t_adj)
    fname_uwnd = FILE_LISTS+'r105_uwnd.nc'.format(t_adj)
    fname_vwnd = FILE_LISTS+'r105_vwnd.nc'.format(t_adj)
    t_head = 'T-5K'
else:
    ra = '{:03}'.format(int(t_adj))
    fname_grid = FILE_LISTS+'r{}_grid.nc'.format(ra)
    fname_uwnd = FILE_LISTS+'r{}_uwnd.nc'.format(ra)
    fname_vwnd = FILE_LISTS+'r{}_vwnd.nc'.format(ra)
    t_head = 'T+{}K'.format(int(t_adj))
```

```python
# forecast hour 12
h_ind = 12
# use same size gridbox for all the runs, only modifying center marker
ym = 6
```

```python
# on the regridded 1-deg data (lat is len 30, lon is len 49)
lat_tot = np.arange(-90., 91., 1.)
lon_tot = np.arange(-180., 181., 1.)
# i-10:i+11 is preferred if not out of bounds; otherwise, use :2*i+1
lats = read_var(FILE_LISTS+'r000_grid.nc', 'lat', None)[0,:2*ym+1]
lons = read_var(FILE_LISTS+'r000_grid.nc', 'lon', None)[0,:2*xm+1]
print(lats.shape, lons.shape)
# midpoints after slicing
j1, j2, i1, i2 = combine_arr(lat_tot, lats, lon_tot, lons)
u_grid = read_var(fname_uwnd, 'U', h_ind)[j1:j2, i1:i2] * units.m / units.s
v_grid = read_var(fname_vwnd, 'V', h_ind)[j1:j2, i1:i2] * units.m / units.s
print("u, v", u_grid.shape, v_grid.shape)
```

```python
# pressure
```

```python
def calc_pres(fid_name, hr):
    p0 = read_var(fid_name, 'PB', hr)[j1:j2, i1:i2] * units.Pa
    p1 = read_var(fid_name, 'P', hr)[j1:j2, i1:i2] * units.Pa
    pres_pa = p0 + p1
    return pres_pa.to('hPa')
```
pres = calc_pres(fname_grid, h_ind)
print(pres.shape)

# geopotential, staggered on edge of grid
def calc_hght(fid_name, hr):
gvty0 = read_var(fid_name, 'PHB', hr)[:, :2*ym+1, :2*xm+1]
gvty1 = read_var(fid_name, 'PH', hr)[:, :2*ym+1, :2*xm+1]
geopot0 = gvty0 + gvty1  # geopotential in m^2/s^2
gleopot1 = av_ar(geopot0, axis=0) * units.m**2 / units.s**2
h = mpcalc.geopotential_to_height(geopot1)
return h
hght = calc_hght(fname_grid, h_ind)
print(hght.shape)

# variable grid spacing, using map factors
def create_deltas(fid_name, dx_in):
    # assume same grid spacing for both x and y directions
    m_file = read_var(fid_name, 'MAPFAC_M', 0)[:, :2*ym+1, :2*xm+1]
    #dx_out = np.empty((13,20))  # make sure array size is 1 less than
data in dir of interest
    #dy_out = np.empty((12,21))
    map_x = av_ar(m_file, axis=1)
    map_y = av_ar(m_file, axis=0)
    print(map_x.shape, map_y.shape)
dx_out = dx_in / map_x
    dy_out = dx_in / map_y
    return dy_out, dx_out  # following same order convention as wrf

# do the interpolation
def sigma2pres(p, h, u, v):
    plevs = np.arange(875., 599., -25.) * units.hPa
    heights, u_int, v_int = log_interpolate_1d(plevs, p, h, u, v, axis=0)
    #temps = calc_temp(thta_iso)
    print(heights.shape, u_int.shape)
    #h_edit, T_edit = estimate_values(plevs, heights, thta_int)
u_edit = xtrap_values(u_int)
v_edit = xtrap_values(v_int)
    return plevs, u_edit, v_edit

pres_final, u_final, v_final = sigma2pres(pres, hght, u_grid, v_grid)
del pres, hght, u_grid, v_grid
print("done interpolate")

dy, dx = create_deltas(FILE_LISTS+'r000_grid.nc', 111000*units.m)  # 1 degree approx. 111 km
scale_y = 111000*units.m  # constant spacing for 1 degree
scale_x = np.mean(dx[m1y,m1x-1:m1x+1])  # estimated constant spacing
df_full = mpcalc.total_deformation(u_final[1], v_final[1], dx, dy, dim_order='yx')
del u_final, v_final

# climatological mean and standard deviation
df_climo_m = read_var('/home/py84/research_new/acm/C1225_12', 'DEF_AV', 0)[0,j1:j2+1,i1:i2+1] / units.s
df_climo_sd = read_var('/home/py84/research_new/acm/C1225_12', 'DEF_STD', 0)[0,j1:j2+1,i1:i2+1] / units.s
# normalized value

```
df_norm = (df_full - df_climo_m) / df_climo_sd
```

# functions to calculate radial distances

```
def dist_from_center(sc_y, sc_x, y_len, x_len, ymid, xmid):
    y_dist, x_dist = np.ogrid[0:y_len,0:x_len]
    dist2 = np.linalg.norm((sc_y*(y_dist-ymid), sc_x*(x_dist-xmid)),axis=0)
    return dist2
```

# sort normalized values within specific distance from center

```
def norm2med(arr, sy, sx, lim, use_top = True):
    ay, ax = arr.shape
    b = arr.copy()
    d_arr = dist_from_center(sy, sx, ay, ax, m1y, m1x)
    m2 = np.zeros_like(arr, dtype=bool)
    m2[d_arr > lim] = 1.
    #print(m2)
    b = np.ma.array(b, mask=m2)
    a_out = b[b.mask==False]
    m_ind = len(a_out)//2
    #print(m_ind)
    a_out = np.sort(a_out)
    if use_top:  # top half of median
        pc_val = np.mean(a_out[m_ind:]
    else:  # bottom half of median
        pc_val = np.mean(a_out[0:m_ind+1])
    return pc_val
```

```
df_sort = norm2med(df_norm, scale_y, scale_x, 500000*units.m)
df_sort
```

# use pandas to open and assign values

```
df = pd.read_csv('precursors_case0.csv', sep=',', header=None,
    names=[None, 'T+0K', 'T-5K', 'T+1K', 'T+5K',  'T+10K'], index_col=0)
```

```
df.at['nstd_DEF.p5u.0',t_head] = df_sort
```

```
df.iloc[:,df.notnull().any().values].to_csv('precursors_case0.csv',
    header=False)  # only includes columns if at least 1 value isn't NaN
```

```
flevs = np.arange(0., 3.1, 0.25)
l2 = np.arange(-3., 3.1, 0.2)
def draw_map(dataset1, l, cm1, x_axis, y_axis, str_ver, n):
    print(n)
    ax = plt.subplot(1, 1, n)
    m = Basemap(llcrnrlon = x_axis[0], llcrnrlat = y_axis[0],
    urcrnrlon = x_axis[-1], urcrnrlat = y_axis[-1], 
    lat_ts = y_axis[ym], resolution = 'l', area_thresh = 100.,
    projection = 'merc')
    nx, ny = np.meshgrid(x_axis, y_axis)
x, y = m(nx, ny)
x_point, y_point = m(x_axis[xm], y_axis[ym1])
parallels = np.arange(10., 65., 2.)
meridians = np.arange(-125., -9., 4.)
m.drawparallels(parallels, labels = [1,0,0,0])
m.drawmeridians(meridians, labels = [0,0,1])
m.drawcoastlines()
m.drawcountries()
```
m.drawstates()
if n==1:
    cs = plt.contourf(x, y, dataset1, l, cmap = cm1)
else:
    cs = plt.contourf(x, y, dataset1, l, cmap = cm1,
    extend='both')
plt.colorbar(cs, shrink = 0.6)
m.plot(x_point, y_point, marker='$\mathrm{L}$', color='r',
markersize=18)
ax.set_title(str_ver, fontsize = 11)
return ax

figw, figh = 9,7
plt.figure(figsize=[figw, figh])
map1 = draw_map(df_full.m * 1.e4, flevs, 'brg', lons, lats, '850 hPa
Deformation (10^{-4}\text{s}^{-1})', 1)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
top=1-0.4/figh)
plt.savefig('fin'+ra+'_02_03_def.png', dpi=60)
# file: center_qg.py
import sys
import numpy as np
import netCDF4 as nc
import pandas as pd
import metpy.calc as mpcalc
import metpy.constants as constants
from metpy.interpolate import log_interpolate_1d
from metpy.units import units
from metpy.plots import convert_gempak_color
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
print("done import")

# array averaging function similar to numpy.diff function (arr[n] + arr[n+1])
def av_ar(a, n=1, axis=-1):
    a = np.asanyarray(a)
    nd = a.ndim
    axis = np.core.multiarray.normalize_axis_index(axis, nd)
    slice1 = [slice(None)] * nd
    slice2 = [slice(None)] * nd
    slice1[axis] = slice(1, None)
    slice2[axis] = slice(None, -1)
    slice1 = tuple(slice1)
    slice2 = tuple(slice2)
    op = not_equal if a.dtype == np.bool_ else np.add
    for _ in range(n):
        a = op(a[slice1], a[slice2]) / 2.
    return a

def read_var(NCfile, varname, hr):
    fid = nc.Dataset(NCfile, 'r')
    var_out = fid.variables[varname][:]
    var_time = var_out[hr]
    fid.close()
    return var_time

# finding location on global grid
def combine_arr(lat1, lat2, lon1, lon2):
    lat_bool = np.in1d(lat1, lat2)
    lon_bool = np.in1d(lon1, lon2)
    return np.nonzero(lat_bool)[0][0], np.nonzero(lat_bool)[0][-1],
    np.nonzero(lon_bool)[0][0], np.nonzero(lon_bool)[0][-1]

try:
    t_adj, ym1, xm = float(sys.argv[1]), int(sys.argv[2]),
    int(sys.argv[3]) # (6, 8) for all except +10 (4, 9)
    print(t_adj, ym1, xm)
except:
    print('error: missing inputs')
sys.exit()

FILE_LISTS = '/home/py84/research_new/
QG_LISTS = '/home/py84/research_new/qg_module/data_qg/
if t_adj == -.5:
    ra = '105'
fname = QG_LISTS+'Q20101225_105'
t_head = 'T-5K'
else:
    ra = '{:03}'.format(int(t_adj))
fname = QG_LISTS+'Q20101225_{ra}.format(ra)
t_head = 'T+\{ra\}K'.format(t_adj)
# print(ra, t_head)

# use same size gridbox for all the runs, only modifying center marker
ym = 6
lat_tot = np.arange(-90., 91., 1.)
lon_tot = np.arange(-180., 181., 1.)
# on the regridded 1-deg data (lat is len 30, lon is len 49)
lats = read_var(FILE_LISTS+'r000_slp.nc', 'latitude', None)[0,:2*ym+1]
lons = read_var(FILE_LISTS+'r000_slp.nc', 'longitude', None)[0,:2*xm+1]

# midpoints after slicing
m1y, m1x = len(lats)//2, len(lons)//2 # use if i-10:i+11
m1y, m1x = ym1, xm # use if 0:2*i+1; ym/xm not necessarily midpoints
j1, j2, i1, i2 = combine_arr(lat_tot, lats, lon_tot, lons)
qg_low = read_var(fname, 'OM_BOT_1D', 0)[16,:2*ym+1,:2*xm+1] *
        units.Pa / units.s
qg_up = read_var(fname, 'OM_TOP_1D', 0)[16,:2*ym+1,:2*xm+1] * units.Pa /
        units.s

# climatological mean and 2-day mean
z_qgl_m = read_var('/home/py84/research_new/acm/C1225_12', 'OM_BOT3_AV',
        0)[0,j1:j2+1,i1:i2+1] * units.Pa / units.s
z_qgu_m = read_var('/home/py84/research_new/acm/C1225_12', 'OM_TOP3_AV',
        0)[0,j1:j2+1,i1:i2+1] * units.Pa / units.s

# standard deviations
z_qgl_sd = read_var('/home/py84/research_new/acm/C1225_12',
        'OM_BOT3_STD', 0)[0,j1:j2+1,i1:i2+1] * units.Pa / units.s
z_qgu_sd = read_var('/home/py84/research_new/acm/C1225_12',
        'OM_TOP3_STD', 0)[0,j1:j2+1,i1:i2+1] * units.Pa / units.s

# the normalization calculations
qg_low_norm = (qg_low - z_qgl_m) / z_qgl_sd
qg_up_norm = (qg_up - z_qgu_m) / z_qgu_sd

# variable grid spacing, using map factors
def create_deltas(fid_name, dx_in): # assume same grid spacing for both
x and y directions
    m_file = read_var(fid_name, 'MAPFAC_M', 0)[2*ym+1,2*xm+1]
    map_x = av_ar(m_file, axis=1)
    map_y = av_ar(m_file, axis=0)
    print(map_x.shape, map_y.shape)
    dx_out = dx_in / map_x
    dy_out = dx_in / map_y
    return dy_out, dx_out # following same order convention as wrf

output
dy, dx = create_deltas(FILE_LISTS+'r000_grid.nc', 111000*units.m) # 1
degree approx. 111 km
scale_y = 111000*units.m # constant spacing for 1 degree
scale_x = np.mean(dx[m1y,m1x-1:m1x+1]) # estimated constant spacing
# functions to calculate radial distances
def dist_from_center(sc_y, sc_x, y_len, x_len, ymid, xmid):
    y_dist, x_dist = np.ogrid[0:y_len,0:x_len]
    dist2 = np.linalg.norm((sc_y*(y_dist-ymid), sc_x*(x_dist-xmid)), axis=0)
    return dist2

# sort normalized values within specific distance from center
def norm2med(arr, sy, sx, lim, use_top = True):
    ay, ax = arr.shape
    b = arr.copy()
    d_arr = dist_from_center(sy, sx, ay, ax, m1y, m1x)
    m2 = np.zeros_like(arr, dtype=bool)
    m2[d_arr > lim] = 1.
    b = np.ma.array(b, mask=m2)
    a_out = b[b.mask==False]
    m_ind = len(a_out)//2
    if use_top: # top half of median
        pc_val = np.mean(a_out[m_ind:]
    else: # bottom half of median
        pc_val = np.mean(a_out[0:m_ind+1])
    return pc_val

qgl_sort = norm2med(qg_low_norm, scale_y, scale_x, 500000*units.m, False)
nqu_sort = norm2med(qg_up_norm, scale_y, scale_x, 500000*units.m, False)
print(qgl_sort)
print(qqu_sort)

# use pandas to open and assign values
df = pd.read_csv('precursors_case0.csv', sep=',', header=None,
names=[None, 'T+0K', 'T-5K', 'T+1K', 'T+5K', 'T+10K'], index_col=0)
df.at['nstd_OM_BOT3.p5l.0',t_head] = qgl_sort
df.at['nstd_OM_TOP3.p5l.0',t_head] = qgu_sort
print(df)
df.iloc[:,df.notnull().any().values].to_csv('precursors_case0.csv',
header=False) # only includes columns if at least 1 value isn't NaN

levs = np.array((-4.,-2.,-1.,-0.5,-0.25,-0.125,0.125,0.25,0.5,1.,2.))
l2 = np.arange(-2., 2.1, 0.2)
cm1 = convert_gempak_color([17,19,5,21,23,32,4,28,29,30])

def draw_map(dataset1, x_axis, y_axis, l, str_ver):
    ax = plt.subplot(1, 1, 1)
    m = Basemap(llcrnrlon = x_axis[0], llcrnrlat = y_axis[0],
    urcrnrlon = x_axis[-1], urcrnrlat = y_axis[-1],
    lat_ts = y_axis[ym], resolution = 'l', area_thresh = 100.,
    projection = 'merc')
x = np.arange(nx, ny)
x_point, y_point = m(x_axis[xm], y_axis[yml1])
    parallels = np.arange(10., 65., 2.)
    meridians = np.arange(-125., -9., 4.)
m.drawparallels(parallels, labels = [1,0,0,0])
m.drawmeridians(meridians, labels = [0,0,0,1])
m.drawcoastlines()
m.drawcountries()
m.drawstates()
cs = plt.contourf(x, y, dataset1, l, colors = cm1, extend='neither')
cb1 = plt.colorbar(cs, shrink = 0.6, ticks = l)
cb1.ax.invert_yaxis()
m.plot(x_point, y_point, marker='$\text{L}$', color='r', markersize=18)
ax.set_title(str_ver, fontsize = 11)
return ax

figw, figh = 9,7
plt.figure(figsize=[figw, figh])
map1 = draw_map(qg_low.to('microbar/s'), lons, lats, levs, "600 hPa QG-$\omega_{bot}$ ($\mu$b/s)"
plt.subplots_adjust(left=0.8/figw, right=1-0.3/figw, bottom=0.4/figh, top=1-0.4/figh)
plt.savefig('fin'+ra+'_04_04_qg-bot.png',dpi=65)

plt.figure(figsize=[figw, figh])
map1 = draw_map(qg_up.to('microbar/s'), lons, lats, levs, "600 hPa QG-$\omega_{top}$ ($\mu$b/s)"
plt.subplots_adjust(left=0.8/figw, right=1-0.3/figw, bottom=0.4/figh, top=1-0.4/figh)
plt.savefig('fin'+ra+'_05_01_qg-top.png',dpi=65)
# file: out4_regrid.py
import sys
import numpy as np
import netCDF4 as nc
import pandas as pd
import metpy.calc as mpcalc
from metpy.units import units
from metpy.plot import convert_gempak_color
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
print("done import")

# array averaging function similar to numpy.diff function (arr[n] +
arr[n+1])
def av_ar(a, n=1, axis=-1):
a = np.asanyarray(a)
da = a._ndim
axis = np.core.multiarray.normalize_axis_index(axis, nd)
slice1 = [slice(None)] * nd
slice2 = [slice(None)] * nd
slice1[axis] = slice(1, None)
slice2[axis] = slice(None, -1)
slice1 = tuple(slice1)
slice2 = tuple(slice2)
op = not_equal if a.dtype == np.bool_ else np.add
for _ in range(n):
a = op(a[slice1], a[slice2]) / 2.
return a

def read_var(NCfile, varname, hr):
    fid = nc.Dataset(NCfile, 'r')
    var_out = fid.variables[varname][:]
    var_time = var_out[hr]
    fid.close()
    return var_time

# finding location on global grid
def combine_arr(lat1, lat2, lon1, lon2):
    lat_bool = np.in1d(lat1, lat2)
    lon_bool = np.in1d(lon1, lon2)
    return np.nonzero(lat_bool)[0][0], np.nonzero(lat_bool)[0][-1],
    np.nonzero(lon_bool)[0][0], np.nonzero(lon_bool)[0][-1]

try:
t_adj, ym1, xm = float(sys.argv[1]), int(sys.argv[2]),
int(sys.argv[3])  # (6, 8) for all except +10 (4, 9)
print(t_adj, ym1, xm)
except:
    print('error: missing inputs')
sys.exit()(

FILE_LISTS = '/home/py84/research_new/
if t_adj == -5.:
    ra = '105'
f_input = FILE_LISTS+'r105_grid.nc'.format(t_adj)
t_head = 'T-5K'
else:
ra = '{:03}'.format(int(t_adj))
f_input = FILE_LISTS+'r{}_grid.nc'.format(ra)
t_head = 'T+{}K'.format(int(t_adj))

# print(ra)
# forecast hour
h_ind = 12
# use same size gridbox for all the runs, only modifying center marker
ym = 6

lat_tot = np.arange(-90., 91., 1.)
lon_tot = np.arange(-180., 181., 1.)

# on the regridded 1-deg data (lat is len 30, lon is len 49)
# i-10:i+11 is preferred if not out of bounds; otherwise, use :2*i+1
lats = read_var(FILE_LISTS+'r000_grid.nc', 'lat', None)[0,:2*ym+1]
lons = read_var(FILE_LISTS+'r000_grid.nc', 'lon', None)[0,:2*xm+1]

print(lats.shape, lons.shape)

# midpoints after slicing
m1y, m1x = len(lats)//2, len(lons)//2 # use if i-10:i+11
m1y, m1x = ym1, xm # use if 0:2*i+1; ym/xm not necessarily midpoints

j1, j2, i1, i2 = combine_arr(lat_tot, lats, lon_tot, lons)

# modified to do 6-hourly total

def read_acc(NCfile, varname, h2, h1):
    fid = nc.Dataset(NCfile, 'r')
    var_out = fid.variables[varname][:]
    var_time = var_out[h2,:2*ym+1,:2*xm+1] -
               var_out[h1,:2*ym+1,:2*xm+1]
    return var_time

rr6h = read_acc(f_input, 'RAINNC', h_ind, h_ind-6) * units.mm # grid scale
rc6h = read_acc(f_input, 'RAINC', h_ind, h_ind-6) * units.mm # convective
slhf = read_acc(f_input, 'ACLHF', h_ind, h_ind-6)# * (units.joule /
       units.m**2) / (6.0 * units.hr)
sshf = read_acc(f_input, 'ACHFX', h_ind, h_ind-6)# * (units.joule /
       units.m**2) / (6.0 * units.hr)
t_sk = (read_var(f_input, 'TSK', h_ind)
       [:2*ym+1,:2*xm+1] * units.K)

# climatological means
rr_climo_m = read_var('/home/py84/research_new/acm/C1225_12', 'RTOT_AV',
                      0)[0,j1:j2+1,i1:i2+1] * units.mm
lh_climo_m = read_var('/home/py84/research_new/acm/C1225_12', 'SLHF_AV',
                      0)[0,j1:j2+1,i1:i2+1]# * (units.joule / units.m**2) / (6.0 * units.hr)
sh_climo_m = read_var('/home/py84/research_new/acm/C1225_12', 'SSHF_AV',
                      0)[0,j1:j2+1,i1:i2+1]# * (units.joule / units.m**2) / (6.0 * units.hr)
sk_climo_m = (read_var('/home/py84/research_new/acm/C1225_12', 'SKT_AV',
                      0) + t_adj)[0,j1:j2+1,i1:i2+1] * units.K

# standard deviations
rr_climo_sd = read_var('/home/py84/research_new/acm/C1225_12',
                        'RTOT_STD', 0)[0,j1:j2+1,i1:i2+1] * units.mm
lh_climo_sd = read_var('/home/py84/research_new/acm/C1225_12',
                        'SLHF_STD', 0)[0,j1:j2+1,i1:i2+1]
sh_climo_sd = read_var('/home/py84/research_new/acm/C1225_12',
                        'SSHF_STD', 0)[0,j1:j2+1,i1:i2+1]
sk_climo_sd = read_var('/home/py84/research_new/acm/C1225_12',
                        'SKT_STD', 0)[0,j1:j2+1,i1:i2+1] * units.K
# normalized values
rr_norm = (rr6h + rc6h + - rr_climo_m) / rr_climo_sd
lh_norm = (slhf + lh_climo_m) / lh_climo_sd # climo values are
multiplied by -1
sh_norm = (sshf + sh_climo_m) / sh_climo_sd # climo values are
multiplied by -1
sk_norm = (t_sk - sk_climo_m) / sk_climo_sd

# variable grid spacing, using map factors
def create_deltas(fid_name, dx_in): # assume same grid spacing for both
x and y directions
    m_file = read_var(fid_name, 'MAPFAC_M', 0)[:2*ym+1,:2*xm+1]
    map_x = av_ar(m_file, axis=1)
    map_y = av_ar(m_file, axis=0)
    print(map_x.shape, map_y.shape)
    dx_out = dx_in / map_x
    dy_out = dx_in / map_y
    return dy_out, dx_out # following same order convention as wrf

output
dy, dx = create_deltas(FILE_LISTS+'r000_grid.nc', 111000*units.m) # 1
degree approx. 111 km
scale_y = 111000*units.m # constant spacing for 1 degree
scale_x = np.mean(dx[m1y,m1x-1:m1x+1]) # estimated constant spacing

# functions to calculate radial distances
def dist_from_center(sc_y, sc_x, y_len, x_len, ymid, xmid):
    y_dist, x_dist = np.ogrid[0:y_len,0:x_len]
    dist2 = np.linalg.norm((sc_y*(y_dist-ymid), sc_x*(x_dist-xmid)),axis=0)
    return dist2

# sort normalized values within specific distance from center
def norm2med(arr, sy, sx, lim, use_top = True, log_trans = False):
    ay, ax = arr.shape
    b = arr.copy()
    d_arr = dist_from_center(sy, sx, ay, ax, m1y, m1x)
    m2 = np.zeros_like(arr, dtype=bool)
    m2[d_arr > lim] = 1.
    b = np.ma.array(b, mask=m2)
    a_out = b[b.mask==False]
    m_ind = len(a_out)//2
    a_out = np.sort(a_out)
    if use_top: # top half of median
        if log_trans:
            a_up = np.log(a_out[m_ind:])
        else:
            a_up = a_out[m_ind:]
        pc_val = np.mean(a_up)
    else: # bottom half of median
        pc_val = np.mean(a_out[0:m_ind+1])
    return pc_val

rr_sort = norm2med(rr_norm, scale_y, scale_x, 500000*units.m,
log_trans=True)
lh_sort = norm2med(lh_norm, scale_y, scale_x, 500000*units.m)
sh_sort = norm2med(sh_norm, scale_y, scale_x, 500000*units.m)
sk_sort = norm2med(sk_norm, scale_y, scale_x, 500000*units.m)

# print sorted normalized values for test
print(rr_sort)
print(lh_sort)
print(sh_sort)
print(sk_sort)

# use pandas to open and assign values
df = pd.read_csv('precursors_case0.csv', sep=',', header=None,
                 names=[None, 'T+0K', 'T-5K', 'T+1K', 'T+5K', 'T+10K'], index_col=0)
df.at['nstd_RTOT.p5u.0', t_head] = rr_sort
df.at['nstd_SLHF.p5u.0', t_head] = lh_sort
df.at['nstd_SSHF.p5u.0', t_head] = sh_sort
df.at['nstd_SKT.p5u.0', t_head] = sk_sort
df.iloc[:, df.notnull().any().values].to_csv('precursors_case0.csv',
                                           header=False) # only includes columns if at least 1 value isn't NaN

# convert units for plotting
slhf = slhf * (units.joule / units.m**2) / (6.0 * units.hr)
sshf = sshf * (units.joule / units.m**2) / (6.0 * units.hr)
rr6h = rr6h + rc6h
lev1 = [0.0, 0.01, 0.25, 1.0, 2.5, 5.0, 10., 15., 20., 25., 35., 50.]
color1 = convert_gempak_color([32,29,28,27,25,24,22,21,20,19,18])
if t_adj < 5:
    lev2 = np.arange(0., 436., 15.)
    lev3 = np.arange(0., 176., 5.)
    lev4 = np.arange(-20.0, 35.0, 2.5)
else:
    lev2 = np.arange(0., 481., 15.)
    lev3 = np.arange(0., 176., 5.)
    lev4 = np.arange(-10.0, 45.0, 2.5)

def draw_map(data1, l, x_axis, y_axis, cm0, str_ver, i1, i2, n,
e='neither', mp=True):
    print(n)
    ax = plt.subplot(i1, i2, n)
    m = Basemap(llcrnrlon = x_axis[0], llcrnrlat = y_axis[0],
                urcrnrlon = x_axis[-1], urcrnrlat = y_axis[-1],
                lat_ts = y_axis[ym], resolution = 'l', area_thresh = 100., projection = 'merc')
    nx, ny = np.meshgrid(x_axis, y_axis)
x, y = m(nx, ny)
x_point, y_point = m(x_axis[xm], y_axis[ym1])
    parallels = np.arange(10., 65., 2.)
    meridians = np.arange(-125., -9., 4.)
    m.drawparallels(parallels, labels = [1,0,0,0])
    m.drawmeridians(meridians, labels = [0,0,0,1])
    m.drawcoastlines()
    m.drawcountries()
    m.drawstates()
    if mp:
        cf = plt.contourf(x, y, data1, l, cmap=cm0, extend=e,
                          alpha=0.7)
        plt.colorbar(cf, shrink=0.6)
    else:
cf = plt.contourf(x, y, data1, l, colors=cm0, extend=e, alpha=0.7)
plt.colorbar(cf, ticks=l, shrink=0.6)
m.plot(x_point, y_point, marker='$\mathrm{L}$', color='r', markersize=18)
ax.set_title(str_ver, fontsize = 10)
return ax

figw, figh = 9.7
plt.figure(figsize=[figw, figh])
map1 = draw_map(rr6h, lev1, lons, lats, color1, "Grid-Scale + Convective Precip (mm)", 1, 1, 1, mp=False)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh, top=1-0.4/figh)
plt.savefig('fin'+ra+'_03_02_rtot.png', dpi=60)

plt.figure(figsize=[figw, figh])
map2 = draw_map(slhf.to('W/m**2'), lev2, lons, lats, "PuRd", "surface latent heat flux (W m$^{-2}$)", 1, 1, 1, e='max')
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh, top=1-0.4/figh)
plt.savefig('fin'+ra+'_03_04_slhf.png', dpi=60)

plt.figure(figsize=[figw, figh])
map2 = draw_map(sshf.to('W/m**2'), lev3, lons, lats, "PuRd", "surface sensible heat flux (W m$^{-2}$)", 1, 1, 1, e='max')
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh, top=1-0.4/figh)
plt.savefig('fin'+ra+'_04_01_sshf.png', dpi=60)

plt.figure(figsize=[figw, figh])
map3 = draw_map(t_sk.to('degC'), lev4, lons, lats, "plasma", "surface skin temp. (degC)", 1, 1, 1)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh, top=1-0.4/figh)
plt.savefig('fin'+ra+'_03_03_tsk.png', dpi=60)
# file: tdelta_grid.py
import sys
import numpy as np
import netCDF4 as nc
import pandas as pd
import metpy.calc as mpcalc
import metpy.constants as constants
from metpy.interpolate import log_interpolate_1d
from metpy.units import units
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
print("done import")

# array averaging function similar to numpy.diff function (arr[n] + arr[n+1])
def av_ar(a, n=1, axis=-1):
    a = np.asanyarray(a)
    nd = a.ndim
    axis = np.core.multiarray.normalize_axis_index(axis, nd)
    slice1 = [slice(None)] * nd
    slice2 = [slice(None)] * nd
    slice1[axis] = slice(1, None)
    slice2[axis] = slice(None, -1)
    slice1 = tuple(slice1)
    slice2 = tuple(slice2)
    op = not_equal if a.dtype == np.bool_ else np.add
    for _ in range(n):
        a = op(a[slice1], a[slice2]) / 2.
    return a

def read_var(NCfile, varname, hr):
    fid = nc.Dataset(NCfile, 'r')
    var_out = fid.variables[varname][:]
    var_time = var_out[hr]
    fid.close()
    return var_time

# finding location on global grid
def combine_arr(lat1, lat2, lon1, lon2):
    lat_bool = np.in1d(lat1, lat2)
    lon_bool = np.in1d(lon1, lon2)
    return np.nonzero(lat_bool)[0][0], np.nonzero(lat_bool)[0][-1],
    np.nonzero(lon_bool)[0][0], np.nonzero(lon_bool)[0][-1]

try:
    t_adj, ym1, xm = float(sys.argv[1]), int(sys.argv[2]),
    int(sys.argv[3]) # (6, 8) for all except +10 (4, 9)
    print(t_adj, ym1, xm)
except:
    print('error: missing inputs')
sys.exit()

FILE_LISTS = '/home/py84/research_new/
if t_adj == -5.:
    ra = '105'
    f_input = FILE_LISTS+'r105_grid.nc'.format(t_adj)
t_head = 'T-5K'
else:
    ra = '{:03}'.format(int(t_adj))
    f_input = FILE_LISTS+'r{0}_grid.nc'.format(ra)
    t_head = 'T+{}K'.format(int(t_adj))

    # print(ra)
    # forecast hour
    h_ind = 12
    # use same size gridbox for all the runs, only modifying center marker
    ym = 6

    lat_tot = np.arange(-90., 91., 1.)
    lon_tot = np.arange(-180., 181., 1.)
    # on the regridded 1-deg data (lat is len 30, lon is len 49)
    # i-10:i+11 is preferred if not out of bounds; otherwise, use :2*i+1
    lats = read_var(FILE_LISTS+'r000_grid.nc', 'lat', None)[0,:2*ym+1]
    lons = read_var(FILE_LISTS+'r000_grid.nc', 'lon', None)[0,:2*xm+1]
    print(lats.shape, lons.shape)

    # midpoints after slicing
    m1y, m1x = len(lats)//2, len(lons)//2  # use if i-10:i+11
    m1y, m1x = ym, xm  # use if 0:2*i+1; ym/xm not necessarily midpoints
    j1, j2, i1, i2 = combine_arr(lat_tot, lats, lon_tot, lons)
    thta = (read_var(f_input, 'T', h_ind)[j1:j2,i1:i2] + 300.0) * units.K
    t_s = (read_var(f_input, 'TH2', h_ind))[j1:j2,i1:i2] * units.K
    print('thta', 'thta_sfc', thta.shape, t_s.shape)

    # pressure
    def calc_pres(fid_name, hr):
        p0 = read_var(fid_name, 'PB', hr)[j1:j2,i1:i2] * units.Pa
        p1 = read_var(fid_name, 'P', hr)[j1:j2,i1:i2] * units.Pa
        pres_pa = p0 + p1
        return pres_pa.to('hPa')
    pres = calc_pres(f_input, h_ind)
    print(pres.shape)

    # geopotential, staggered on edge of grid
    def calc_hght(fid_name, hr):
        gvty0 = read_var(fid_name, 'PHB', hr)[j1:j2,i1:i2]
        gvty1 = read_var(fid_name, 'PH', hr)[j1:j2,i1:i2]
        geopot0 = gvty0 + gvty1  # geopotential in m2/s2
        geopot1 = av_ar(geopot0, axis=0) * units.m**2 / units.s**2
        h = mpcalc.geopotential_to_height(geopot1)
        return h
    hght = calc_hght(f_input, h_ind)
    print(hght.shape)

    plevs = [700.] * units.hPa
    # do the interpolation
    def sigma2pres(pl, p, h, th):
        heights, th_int = log_interpolate_1d(pl, p, h, th, axis=0)
        print(heights[0].shape, th_int[0].shape)
        return heights[0], th_int[0]
    hght_iso, thta_iso = sigma2pres(plevs, pres, hght, thta)
    del pres, hght
    print("done interpolate")
# print(ttha_iso[2:11,6:15] - t_s[2:11,6:15])
dt = np.ma.masked_array(ttha_iso-t_s, np.isnan(ttha_iso))  # use function to make masked array

# climatological mean and standard deviation
dt_climo_m = read_var('/home/py84/research_new/acm/C1225_12', 'THDIFF_AV', 0)[0,j1:j2+1,i1:i2+1] * units.K
dt_climo_sd = read_var('/home/py84/research_new/acm/C1225_12', 'THDIFF_STD', 0)[0,j1:j2+1,i1:i2+1] * units.K

# normalized value
dt_norm = (dt - dt_climo_m) / dt_climo_sd

# variable grid spacing, using map factors
def create_deltas(fid_name, dx_in):
    # assume same grid spacing for both x and y directions
    m_file = read_var(fid_name, 'MAPFAC_M', 0)[:2*ym+1,:2*xm+1]
    map_x = av_ar(m_file, axis=1)
    map_y = av_ar(m_file, axis=0)
    print(map_x.shape, map_y.shape)
    dx_out = dx_in / map_x
    dy_out = dx_in / map_y
    return dy_out, dx_out  # following same order convention as wrf output

dy, dx = create_deltas(FILE_LISTS+'r000_grid.nc', 111000*units.m)  # 1 degree approx. 111 km
scale_y = 111000*units.m  # constant spacing for 1 degree
scale_x = np.mean(dx[m1y,m1x-1:m1x+1])  # estimated constant spacing

# functions to calculate radial distances
def dist_from_center(sc_y, sc_x, y_len, x_len, ymid, xmid):
    y_dist, x_dist = np.ogrid[0:y_len,0:x_len]
    dist2 = np.linalg.norm((sc_y*(y_dist-ymid), sc_x*(x_dist-xmid)),axis=0)
    return dist2

def norm2med(arr, sy, sx, lim, use_top=True):
    ay, ax = arr.shape
    b = arr.copy()
    d_arr = dist_from_center(sy, sx, ay, ax, m1y, m1x)
    m2 = np.zeros_like(arr, dtype=bool)
    m2[d_arr > lim] = 1.
    #print(m2)
    b = np.ma.array(b, mask=m2)
    a_out = b[b.mask==False]
    m_ind = len(a_out)//2
    #print(m_ind)
    a_out = np.sort(a_out)
    if use_top:  # top half of median
        pc_val = np.mean(a_out[m_ind:])
    else:  # bottom half of median
        pc_val = np.mean(a_out[0:m_ind+1])
    return pc_val

dt_sort = norm2med(dt_norm, scale_y, scale_x, 500000*units.m, False)
print(dt_sort)
# use pandas to read and open files

df = pd.read_csv('precursors_case0.csv', sep=',', header=None, names=['nstd_THDIFF.p5l.0', 'T+0K', 'T-5K', 'T+1K', 'T+5K', 'T+10K'], index_col=0)
df.at[dt_sort, t_head] = dt_sort
print(df)
df.iloc[:, df.notnull().any().values].to_csv('precursors_case0.csv', header=False)  # only includes columns if at least 1 value isn't NaN

lev = np.arange(30.,)
l2 = np.arange(-2.6, 2.7, 0.2)

plt.figure(figsize = [17,8])
def draw_map(dataset1, l, cm1, x_axis, y_axis, str_ver, n):
    print(n)
    ax = plt.subplot(1, 1, n)
    m = Basemap(llcrnrlon = x_axis[0], llcrnrlat = y_axis[0], urcrnrlon = x_axis[-1], urcrnrlat = y_axis[-1], lat_ts = y_axis[ym], resolution = 'l', area_thresh = 100., projection = 'merc')
    nx, ny = np.meshgrid(x_axis, y_axis)
    x, y = m(nx, ny)
    x_point, y_point = m(x_axis[ym1], y_axis[ym])
    parallels = np.arange(10., 65., 2.)
    meridians = np.arange(-125., -9., 4.)
    m.drawparallels(parallels, labels = [1,0,0,0])
    m.drawmeridians(meridians, labels = [0,0,0,1])
    m.drawcoastlines()
    m.drawcountries()
    m.drawstates()
    if n==1:
        cs = plt.contourf(x, y, dataset1, l, cmap = cm1)
    else:
        cs = plt.contourf(x, y, dataset1, l, cmap = cm1, extend='both')
    plt.colorbar(cs, shrink = 0.6)
    m.plot(x_point, y_point, marker='$\mathrm{L}$', color='r', markersize=18)
    ax.set_title(str_ver, fontsize = 11)
    return ax

figw, figh = 9,7
plt.figure(figsize=[figw, figh])
map1 = draw_map(dt, lev, 'brg', lons, lats, r'$\Delta\theta$ Between Surface and 700 hPa (K)', 1)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh, top=1-0.4/figh)
plt.savefig('fin'+ra+'_04_02_delta700.png',dpi=60)
# file: center_850.py
import sys
import numpy as np
import netCDF4 as nc
import pandas as pd
import metpy.calc as mpcalc
import metpy.constants as constants
from metpy.interpolate import log_interpolate_1d
from metpy.units import units
from metpy.plots import convert_gempak_color
import matplotlib.pyplot as plt
from matplotlib.colors import BoundaryNorm
from matplotlib.colors import LinearSegmentedColormap
from mpl_toolkits.basemap import Basemap

print('done import')

def read_var(NCfile, varname, hr):
    fid = nc.Dataset(NCfile, 'r')
    var_out = fid.variables[varname][:]
    var_time = var_out[hr]
    fid.close()
    return var_time

def log_av(arr):
    arr_log = np.log(arr)
    #a2 = np.mean(arr_log, axis=0)
    z = len(arr_log)
    a2 = (arr_log[:z-1] + arr_log[1:]) / 2.
    arr_out = np.exp(a2)
    return arr_out

# array averaging function similar to numpy.diff function (arr[n] +
# arr[n+1])
def av_ar(a, n=1, axis=-1):
    a = np.asanyarray(a)
    nd = a.ndim
    axis = np.core.multiarray.normalize_axis_index(axis, nd)
    slice1 = [slice(None)] * nd
    slice2 = [slice(None)] * nd
    slice1[axis] = slice(1, None)
    slice2[axis] = slice(None, -1)
    slice1 = tuple(slice1)
    slice2 = tuple(slice2)
    op = not_equal if a.dtype == np.bool_ else np.add
    for _ in range(n):
        a = op(a[slice1], a[slice2]) / 2.
    return a

def combine_arr(lat1, lat2, lon1, lon2):
    lat_bool = np.in1d(lat1, lat2)
    lon_bool = np.in1d(lon1, lon2)
    return np.nonzero(lat_bool)[0][0], np.nonzero(lat_bool)[0][-1],
    np.nonzero(lon_bool)[0][0], np.nonzero(lon_bool)[0][-1]

try:
t_adj, ym1, xm = float(sys.argv[1]), int(sys.argv[2]), int(sys.argv[3]) # (6, 8) for all except +10 (4, 9)
print(t_adj, ym1, xm)
except:
    print('error: missing inputs')
sys.exit()

FILE_LISTS = '/home/py84/research_new/
if t_adj == -5.:
    ra = '105'
    fname_grid = FILE_LISTS+'r105_grid.nc'.format(t_adj)
    fname_uwnd = FILE_LISTS+'r105_uwnd.nc'.format(t_adj)
    fname_vwnd = FILE_LISTS+'r105_vwnd.nc'.format(t_adj)
    t_head = 'T-5K'
else:
    ra = '{:03}'.format(int(t_adj))
    fname_grid = FILE_LISTS+'r{}_grid.nc'.format(ra)
    fname_uwnd = FILE_LISTS+'r{}_uwnd.nc'.format(ra)
    fname_vwnd = FILE_LISTS+'r{}_vwnd.nc'.format(ra)
    t_head = 'T+{}K'.format(int(t_adj))
print(ra, t_head)
# forecast hour 12
h_ind = 12
# use same size gridbox for all the runs, only modifying center marker
ym = 6

lat_tot = np.arange(-90., 91., 1.)
lon_tot = np.arange(-180., 181., 1.)
# on the regridded 1-deg data (lat is len 30, lon is len 49)
lats = read_var(FILE_LISTS+'r000_slp.nc', 'latitude', None)[0,:2*ym+1]
lons = read_var(FILE_LISTS+'r000_slp.nc', 'longitude', None)[0,:2*xm+1]
print(lats.shape, lons.shape)
# midpoints after slicing
#m1y, m1x = len(lats)//2, len(lons)//2 # use if i-10:i+11
m1y, m1x = ym1, xm # use if 0:2*i+1; ym/xm not necessarily midpoints
j1, j2, i1, i2 = combine_arr(lat_tot, lats, lon_tot, lons)

w_base = read_var(fname_grid, 'QVAPOR', h_ind)[:,2*ym+1,2*xm+1] * units.dimensionless
u_grid = read_var(fname_uwnd, 'U', h_ind)[:,2*ym+1,2*xm+1] * units.m / units.s
v_grid = read_var(fname_vwnd, 'V', h_ind)[:,2*ym+1,2*xm+1] * units.m / units.s
th_sig = (read_var(fname_grid, 'T', h_ind) + 300.0)[:,2*ym+1,2*xm+1] * units.K
print("u, v, pot temp", u_grid.shape, v_grid.shape, th_sig.shape)

# pressure
def calc_pres(fid_name, hr):
    p0 = read_var(fid_name, 'PB', hr)[:,2*ym+1,2*xm+1] * units.Pa
    p1 = read_var(fid_name, 'P', hr)[:,2*ym+1,2*xm+1] * units.Pa
    pres_pa = p0 + p1
    return pres_pa.to('hPa')
pres = calc_pres(fname_grid, h_ind)
print(pres.shape)

# geopotential, staggered on edge of grid
def calc_hght(fid_name, hr):
def calc_hght(fname_grid, h_ind):
    hght = mpcalc.geopotential_to_height(geopot1)
    print(hght.shape)
    print("done base variables")

    # hypsometric equation to calculate temp/height at 1 point
    def calc_point(p, h, T1):
        p_next = p + (25.000 * units.hPa)
        dTdz = 0.00650 * units.K / units.m
        dlnp = np.log(p_next / p)
        T_num = dTdz * constants.dry_air_gas_constant * dlnp
        z_num = constants.dry_air_gas_constant * T1 * dlnp
        all_den = constants.earth_gravity - (0.500 * 
            constants.dry_air_gas_constant * dTdz * dlnp)
        T_out = T1 * (1.000 + (T_num / all_den))
        th_out = mpcalc.potential_temperature(p_next, T_out)
        z_out = h - (z_num / all_den)
        return z_out, T_out, th_out

    # functions to replace NAN's with extrapolated values
    # hypsometric equation extrapolation, loops through vertical axis
    def estimate_values(p_arr, h_arr, T_arr, th_arr):
        has_nan_occurred = np.full_like(T_arr[0], False)
        print(has_nan_occurred.shape)
        for k in range(len(p_arr)-2,-1,-1):
            print(k)
            if np.any(np.isnan(T_arr[k])):
                print("yes")
                for j in range(len(T_arr[k])):
                    if np.isnan(T_arr[k,j]):
                        if has_nan_occurred[j] == False:
                            m_layer = mpcalc.mixed_layer(p_arr[k+1:k+6], T_arr[k+1:k+6,j], 
                                                        h_arr[k+1:k+6,j], interpolate=False)
                            use_T = m_layer[0]
                            has_nan_occurred[j] = True
                        else:
                            use_T = T_arr[k+1,j]
                        z_next, T_next, th_next = calc_point(p_arr[k+1], h_arr[k+1,j], use_T)
                        h_arr[k+1,j] = z_next
                        T_arr[k+1,j] = T_next
                        th_arr[k+1,j] = th_next
            return h_arr, T_arr, th_arr

    # function to replace NAN's with the above value, loops through vertical axis
    def xtrap_values(arr):
        for k in range(len(arr)-2,-1,-1):
            if np.any(np.isnan(arr[k])):
                for j in range(len(arr[k])):
for i in range(len(arr[k,j])):
    if np.isnan(arr[k,j,i]):
        arr[k,j,i] = arr[k+1,j,i]

    return arr

# variable grid spacing, using map factors
def create_deltas(fid_name, dx_in):
    # assume same grid spacing for both x and y directions
    m_file = read_var(fid_name, 'MAPFAC_M', 0)[2*ym+1,2*xm+1]
    map_x = av_ar(m_file, axis=1)
    map_y = av_ar(m_file, axis=0)
    print(map_x.shape, map_y.shape)
    dx_out = dx_in / map_x
    dy_out = dx_in / map_y
    return dx_out, dy_out # following same order convention as wrf

# function to calculate theta_e at 1 pressure level
def eq_pot_temp(p_wanted, t_grid, w2):
    e_part = mpcalc.vapor_pressure(p_wanted, w2)
    dp = mpcalc.dewpoint(e_part)
    t_e = mpcalc.equivalent_potential_temperature(p_wanted, t_grid, dp)
    return t_e

# do the interpolation
def sigma2pres(p, h, th, w1, u, v):
    plevs = np.arange(875., 499., -25.) * units.hPa
    heights, th_int, w1_int, u_int, v_int = log_interpolate_1d(plevs, p, h, th, w1, u, v, axis=0)
    print(heights.shape, th_int.shape, w1_int.shape, u_int.shape)
    temps_in = mpcalc.temperature_from_potential_temperature(plevs[:], None, None, th_int)
    h_edit, temps_K, thta, temps_K, w_edit, u_edit, v_edit
    return plevs, h_edit, th_edit, temps_K, w_edit, u_edit, v_edit

sigma2pres(pres, hght, thta, T_iso, w_base, u_grid, v_grid)

thta_e = eq_pot_temp(pres_final[1], T_iso[1], w[1])
del pres, hght, thta, T_iso, w_base, u_grid, v_grid, w

dy, dx = create_deltas(FILE_LISTS+'r000_grid.nc', 111000*units.m) # 1 degree approx. 111 km
scale_y = 111000*units.m # constant spacing for 1 degree
scale_x = np.mean(dx[m1y,m1x-1:m1x+1]) # estimated constant spacing

fg_si = mpcalc.frontogenesis(thta[1], u_final[1], v_final[1], dx, dy, dim_order='yx')
ad850 = mpcalc.advection(T_iso[1], wind=(u_final[1], v_final[1]), deltas=(dx, dy), dim_order='yx') # deltas reversed
grad_vect = mpcalc.gradient(thta[1], deltas=(dy, dx))
gradVect_e = mpcalc.gradient(thta_e, deltas=(dy, dx))
grad_t = mpcalc.wind_speed(grad_vect[1], grad_vect[0])
grad_e = mpcalc.wind_speed(gradVect_e[1], gradVect_e[0])
del grad_vect, grad_vect_e

# constants
# degrees adjustment for run case
c_pd = 1005.7 * units.joule / (units.kg * units.K) # specific heat of dry air at const pres
L_v = 2406. * 1000 * units.joule / units.kg # latent heat of evap, at 40C for simplicity

# using simplified eq pot temp equation to re-calculate
def recalculate_theta_e(te_arr, t_arr, offset):
    # the = T + (Lv/c * r)(p0/p)^(R/c)
    p_part = (1000./850.)**(constants.dry_air_gas_constant / c_pd)
    L_part = L_v / c_pd
    r_est = ((te_arr / p_part) - t_arr) / L_part
    t_arr_new = t_arr + offset
    te_out = (t_arr + offset) + (L_part * r_est) * p_part
    return te_out

# climatological means
t850_m = (read_var('/home/py84/research_new/t_pert_era/C1225_12_T850.nc', 'temperature', None) + t_adj)[0,:2*ym+1,:2*xm+1] * units.K
fg_m = read_var('/home/py84/research_new/acm/C1225_12', 'FGEN850_AV', 0)[0,j1:j2+1,i1:i2+1] # check units, seems to be K/m/s
ad850_m = read_var('/home/py84/research_new/acm/C1225_12', 'TADV850_AV', 0)[0,j1:j2+1,i1:i2+1] * units.K / units.s
if t_adj == 0:
    t850_30day = read_var('/home/py84/research_new/t_pert_era/C1225_12_T30d.nc', 'temperature', None)[0,:2*ym+1,:2*xm+1] * units.K
    theta_e_m = read_var('/home/py84/research_new/acm/C1225_12', 'THE850_AV', 0)[0,j1:j2+1,i1:i2+1] * units.K
else:
    t850_30day = read_var('/home/py84/research_new/t_pert_era/C1225_12_T30d.nc', 'temperature', None)[0,:2*ym+1,:2*xm+1] * units.K
    theta_e_orig = read_var('/home/py84/research_new/acm/C1225_12', 'THE850_AV', 0)[0,j1:j2+1,i1:i2+1] * units.K
    theta_e_m = recalculate_theta_e(theta_e_orig, t850_30day, t_adj*units.K)
grad_t_m = read_var('/home/py84/research_new/acm/C1225_12', 'DT850_AV', 0)[0,j1:j2+1,i1:i2+1] # K/m
grad_e_m = read_var('/home/py84/research_new/acm/C1225_12', 'DTHE850_AV', 0)[0,j1:j2+1,i1:i2+1] # K/m
tp_m = read_var('/home/py84/research_new/acm/C1225_12', 'T_PERT_AV', 0)[0,j1:j2+1,i1:i2+1] * units.K

# standard deviations
fg_sd = read_var('/home/py84/research_new/acm/C1225_12', 'FGEN850_STD', 0)[0,j1:j2+1,i1:i2+1] # check units, if K/100km/3hr
ad850_sd = read_var('/home/py84/research_new/acm/C1225_12', 'TADV850_STD', 0)[0,j1:j2+1,i1:i2+1] * units.K / units.s
theta_e_sd = read_var('/home/py84/research_new/acm/C1225_12', 'THE850_STD', 0)[0,j1:j2+1,i1:i2+1] * units.K
grad_t_sd = read_var('/home/py84/research_new/acm/C1225_12', 'DT850_STD', 0)[0,j1:j2+1,i1:i2+1] # check units
grad_e_sd = read_var('/home/py84/research_new/acm/C1225_12', 'DTHE850_STD', 0)[0,j1:j2+1,i1:i2+1] # check units
tp_sd = read_var('/home/py84/research_new/acm/C1225_12', 'T_PERT_STD', 0)[j1:j2+1, i1:i2+1] * units.K

# temp anomaly
a10_850 = T_iso[1] - t850_m

# normalized values
tp_norm = (a10_850 - tp_m) / tp_sd
fg_norm = (fg_si.m - fg_m) / fg_sd
ad850_norm = (ad850 - ad850_m) / ad850_sd
thta_e_norm = (thta_e - thta_e_m) / thta_e_sd
grad_t_norm = (grad_t.m - grad_t_m) / grad_t_sd
grad_e_norm = (grad_e.m - grad_e_m) / grad_e_sd

# functions to calculate radial distances
def dist_from_center(sc_y, sc_x, y_len, x_len, ymid, xmid):
    dist2 = np.linalg.norm((sc_y*(y_dist-ymid), sc_x*(x_dist-xmid)),axis=0)
    return dist2

# sort normalized values within specific distance from center
def norm2med(arr, sy, sx, lim, use_top=True, log_trans=False):
    ay, ax = arr.shape
    b = arr.copy()
    d_arr = dist_from_center(sy, sx, ay, ax, m1y, m1x)
    m2 = np.zeros_like(arr, dtype=bool)
    m2[d_arr > lim] = 1.
    b = np.ma.array(b, mask=m2)
    a_out = b[b.mask==False]
    m_ind = len(a_out)//2
    a_out = np.sort(a_out)
    if use_top:  # top half of median
        if log_trans:
            a_up = np.log(a_out[m_ind:])
        else:
            a_up = a_out[m_ind:]
        pc_val = np.mean(a_up)
    else:  # bottom half of median
        pc_val = np.mean(a_out[0:m_ind+1])
    return pc_val

tp_sort = norm2med(tp_norm, scale_y, scale_x, 500000*units.m)
fg_sort = norm2med(fg_norm, scale_y, scale_x, 500000*units.m, log_trans=True)
ad850_sort = norm2med(ad850_norm, scale_y, scale_x, 500000*units.m)
th_e_sort = norm2med(thta_e_norm, scale_y, scale_x, 500000*units.m)
gradi_t_sort = norm2med(grad_t_norm, scale_y, scale_x, 500000*units.m)
gradi_e_sort = norm2med(grad_e_norm, scale_y, scale_x, 500000*units.m)

# print(tp_sort)
# print(fg_sort)
# print(ad850_sort)
# print(th_e_sort)
# print(grad_t_sort)
# print(grad_e_sort)
# adding normalized values to precursor list

```python
df = pd.read_csv('precursors_case0.csv', sep='\,', header=None, 
names=[None, 'T+0K', 'T-5K', 'T+1K', 'T+5K', 'T+10K'], index_col=0)
df.at['nstd_FGEN850.p5u.0',t_head] = fg_sort
df.at['nstd_T_PERT.p5u.0',t_head] = tp_sort
df.at['nstd_DT850.p5u.0',t_head] = grad_t_sort
df.at['nstd_DTHE850.p5u.0',t_head] = grad_e_sort
df.at['nstd_THE850.p5u.0',t_head] = th_e_sort
df.at['nstd_TADV850.p5u.0',t_head] = ad850_sort
print(df)
df.iloc[:,df.notnull().any().values].to_csv('precursors_case0.csv', 
header=False) # only includes columns if at least 1 value isn't NaN
```

# plotting

```python
# Use custom colormap function from Earle
def custom_div_cmap(numcolors=11, name='custom_div_cmap', mincol='blue', 
midcol='white', maxcol='red'):
    """ Create a custom diverging colormap with three colors
    Default is blue to white to red with 11 colors. Colors can be
    specified in any way understandable by
    matplotlib.colors.ColorConverter.to_rgb()
    """
    cmap_wanted = LinearSegmentedColormap.from_list(name=name, 
    colors = [mincol, midcol, maxcol], N=numcolors)
    return cmap_wanted

al = np.array((-10., -8., -6., -4., -2., -1., -0.5, -0.25, 0.25, 0.5, 
1., 2., 4., 6., 8., 10.))
N = len(al) - 1
bnorm = BoundaryNorm(al, N)
cmap2 = custom_div_cmap(N)
```

```python
n1 = np.arange(-3., 3.1, 0.2)
tc1 = np.arange(-14., 15., 1.)
tc2 = np.arange(-9., 10., 1.)
fgl = np.array((-0.5, -0.25, 0.25, 0.5, 1., 2., 4., 6., 8., 10.))
```

```python
if t_adj < 7:
    thel = np.arange(255., 331., 5.)
    grel = np.arange(0., 10.1, 0.5)
else:
    thel = np.arange(275., 351., 5.)
    grel = np.arange(0., 15.1, 0.5)
```

```python
def draw_map(dataset1, l, x_axis, y_axis, str_ver, cm, e, i1, i2, n, 
ms=True, Normalize=False):
    print(n)
    ax = plt.subplot(i1, i2, n)
    m = Basemap(llcrnrlon = x_axis[0], llcrnrlat = y_axis[0], 
    urcrnrlon = x_axis[-1], urcrnrlat = y_axis[-1], 
    lat_ts = y_axis[ym], resolution = 'l', area_thresh = 100., projection = 'merc')
    nx, ny = m.meshgrid(x_axis, y_axis)
    x, y = m(nx, ny)
```
# sfc center here
x_point, y_point = m(x_axis[xm], y_axis[ym1])
parallels = np.arange(10., 65., 2.)
meridians = np.arange(-125., -9., 4.)
m.drawparallels(parallels, labels = [1,0,0,0])
m.drawmeridians(meridians, labels = [0,0,1,1])
m.drawcoastlines()
m.drawcountries()
m.drawstates()
if ms:
    if Normalize:
        cf = plt.contourf(x, y, dataset1, l, norm=bnorm,
cmap=cm)
    else:
        cf = plt.contourf(x, y, dataset1, l, cmap=cm,
extend=e)
else:
    cf = plt.contourf(x, y, dataset1, l, colors=cm, extend=e)
plt.colorbar(cf, shrink = 0.6)
m.plot(x_point, y_point, marker='$\mathrm{L}$', color='r,
markersize=18)
ax.set_title(str_ver, fontsize = 11)
return ax

figw, figh = 9,7
plt.figure(figsize=[figw, figh])
map1 = draw_map(fg_si.m * 1.000e4 * 1.000e5, fgl, lons, lats,
'Frontogenesis (K/100km/3hr)', cm0, 'both', 1, 1, 1, False)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
top=1-0.4/figh)
plt.savefig('fin'+ra+'_02_02_fg850.png', dpi=60)

plt.figure(figsize=[figw, figh])
map2 = draw_map(ad850.to('K/hr'), al, lons, lats, 'Horiz. Temp. Adv. (K/hr)',
cmap2, 'neither', 1, 1, 1, True, True)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
top=1-0.4/figh)
plt.savefig('fin'+ra+'_06_03_tadv.png', dpi=60)

plt.figure(figsize=[figw, figh])
map2 = draw_map(grad_t * 1.e5, grtl, lons, lats, r'Horiz. |$\nabla\theta_{850}$| (K/100km)', cm1, 'max', 1, 1, 1,
False)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
top=1-0.4/figh)
plt.savefig('fin'+ra+'_05_04_gradt.png', dpi=60)

plt.figure(figsize=[figw, figh])
map3 = draw_map(thta_e, thel, lons, lats, "850 hPa Eq. Potential Temp (K)", 'plasma', 'both', 1, 1, 1)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
top=1-0.4/figh)
plt.savefig('fin'+ra+'_06_02_te.png', dpi=60)

plt.figure(figsize=[figw, figh])
map5 = draw_map(grad_e*1.e5, grel, lons, lats, r"|$\nabla\theta_{e850}$| (K/100km)'", 'plasma', 'neither', 1, 1, 1)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
top=1-0.4/figh)
plt.savefig('fin'+ra+'\_06\_01\_grad-te.png', dpi=60)

plt.figure(figsize=[figw, figh])
map4 = draw_map(a10_850, tc2, lons, lats, '850 hPa Temp. Anomaly (K)', 'bwr', 'both', 1, 1, 1)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh, top=1-0.4/figh)
plt.savefig('fin'+ra+'\_04\_03\_tp850.png', dpi=60)
# takes about 47 sec to run
# file center_pv.py
import sys
import numpy as np
import netCDF4 as nc
import pandas as pd
import metpy.calc as mpcalc
import metpy.constants as constants
from metpy.interpolate import log_interpolate_1d
from metpy.units import units
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
print('done import')

# array averaging function similar to numpy.diff function (arr[n] +
# arr[n+1])
def av_ar(a, n=1, axis=-1):
    a = np.asanyarray(a)
    nd = a.ndim
    axis = np.core.multiarray.normalize_axis_index(axis, nd)
    slice1 = [slice(None)] * nd
    slice2 = [slice(None)] * nd
    slice1[axis] = slice(1, None)
    slice2[axis] = slice(None, -1)
    slice1 = tuple(slice1)
    slice2 = tuple(slice2)
    op = not_equal if a.dtype == np.bool_ else np.add
    for _ in range(n):
        a = op(a[slice1], a[slice2]) / 2.
    return a

# finding location on global grid
def combine_arr(lat1, lat2, lon1, lon2):
    lat_bool = np.in1d(lat1, lat2)
    lon_bool = np.in1d(lon1, lon2)
    return np.nonzero(lat_bool)[0][0], np.nonzero(lat_bool)[0][-1],
    np.nonzero(lon_bool)[0][0], np.nonzero(lon_bool)[0][-1]

def read_var(NCfile, varname, hr):
    fid = nc.Dataset(NCfile, 'r')
    var_out = fid.variables[varname][:]
    var_time = var_out[hr]
    fid.close()
    return var_time

try:
    t_adj, ym1, xm = float(sys.argv[1]), int(sys.argv[2]),
    int(sys.argv[3]) # (6, 8) for all except +10 (4, 9)
    print(t_adj, ym1, xm)
except:
    print('error: missing inputs')
sys.exit()

FILE_LISTS = '/home/py84/research_new/'
if t_adj == -5.:
    ra = '105'
    fname_grid = FILE_LISTS+'r105_grid.nc'.format(t_adj)
    fname_uwnd = FILE_LISTS+'r105_uwnd.nc'.format(t_adj)
fname_vwnd = FILE_LISTS+'r105_vwnd.nc'.format(t_adj)
t_head = 'T-5K'
else:
    ra = '{:03}'.format(int(t_adj))
    fname_grid = FILE_LISTS+'r{}_grid.nc'.format(ra)
    fname_uwnd = FILE_LISTS+'r{}_uwnd.nc'.format(ra)
    fname_vwnd = FILE_LISTS+'r{}_vwnd.nc'.format(ra)
t_head = 'T+{}K'.format(int(t_adj))

# forecast hour 12
h_ind = 12
# use same size gridbox for all the runs, only modifying center marker
ym = 6

# on the regridded 1-deg data (lat is len 30, lon is len 49)
lat_tot = np.arange(-90., 91., 1.)
lon_tot = np.arange(-180., 181., 1.)

# i-10:i+11 is preferred if not out of bounds; otherwise, use :2*i+1
lats = read_var(FILE_LISTS+'r000_grid.nc', 'lat', None)[0,:2*ym+1]
lons = read_var(FILE_LISTS+'r000_grid.nc', 'lon', None)[0,:2*xm+1]
xm, ym = np.meshgrid(lons, lats)
# print(nx.shape, ny.shape)
# midpoints after slicing
# m1y, m1x = len(lats)//2, len(lons)//2 # use if i-10:i+11
m1y, m1x = ym1, xm # use if 0:2*i+1; ym/xm not necessarily midpoints
j1, j2, i1, i2 = combine_arr(lat_tot, lats, lon_tot, lons)
u_grid = read_var(fname_uwnd, 'U', h_ind)[0,:,:2*ym+1,:2*xm+1] * units.m / units.s
v_grid = read_var(fname_vwnd, 'V', h_ind)[0,:,:2*ym+1,:2*xm+1] * units.m / units.s
thta = (read_var(fname_grid, 'T', h_ind)[0,:,:2*ym+1,:2*xm+1] + 300.0) * units.K
print("u, v, theta", u_grid.shape, v_grid.shape, thta.shape)

# pressure
def calc_pres(fid_name, hr):
p0 = read_var(fid_name, 'PB', hr)[0,:,:2*ym+1,:2*xm+1] * units.Pa
p1 = read_var(fid_name, 'P', hr)[0,:,:2*ym+1,:2*xm+1] * units.Pa
pres_pa = p0 + p1
return pres_pa.to('hPa')
pres = calc_pres(fname_grid, h_ind)
print(pres.shape)

def log_av(arr):
    arr_log = np.log(arr)
a2 = np.mean(arr_log, axis=0)
arr_out = np.exp(a2)
return arr_out

# geopotential, staggered on edge of grid
def calc_hght(fid_name, hr):
gvty0 = read_var(fid_name, 'PHB', hr)[0,:,:2*ym+1,:2*xm+1]
gvty1 = read_var(fid_name, 'PH', hr)[0,:,:2*ym+1,:2*xm+1]
geopot0 = gvty0 + gvty1 # geopotential in m2/s2
ggeopot1 = av_ar(ggeopot0, axis=0) * units.m**2 / units.s**2
h = mpcalc.geopotential_to_height(ggeopot1)
return h
hght = calc_hght(fname_grid, h_ind)
print(hght.shape)
print("done base variables")

# variable grid spacing, using map factors
def create_deltas(fid_name, dx_in):
    # assume same grid spacing for both x and y directions
    m_file = read_var(fid_name, 'MAPFAC_M', 0)[2*ym+1,2*xm+1]
    #dx_out = np.empty((13,20)) # make sure array size is 1 less than data in dir of interest
    #dy_out = np.empty((12,21))
    map_x = av_ar(m_file, axis=1)
    map_y = av_ar(m_file, axis=0)
    print(map_x.shape, map_y.shape)
    dx_out = dx_in / map_x
    dy_out = dx_in / map_y
    return dy_out, dx_out # following same order convention as wrf output

# calculate B-V frequency sq and non-sq values for a layer, excluding stratosphere
def N_levels_sq(ht, tha, pv_ref):
    N_new = np.zeros_like(ht)
    N_2 = np.full_like(ht, float('nan'))
    for j in range(len(N_new[0])):
        for i in range(len(N_new[0,0])):
            N_column = mpcalc.brunt_vaisala_frequency(ht[:,j,i],
                                                         tha[:,j,i])
            N_new[:,j,i] = N_column
            k = len(pv_ref) - 1
            while pv_ref.m[k,j,i] > 2.00e-6: #loop from top, stopping when value first drops below 2PVU
                #print(k, j, i)
                #N_2[k,j,i] = float('nan')
                k += -1
            else:
                print(k, j, i)
                N_c2 = mpcalc.brunt_vaisala_frequency_squared(ht[:k+1,j,i], tha[:k+1,j,i])
                N_2[:k+1,j,i] = N_c2
    return N_new / units.s, N_2 / units.s**2

# calculate B-V frequency values for a layer
def N_levels(ht, tha):
    N_new = np.empty(ht.shape)
    for j in range(len(N_new[0])):
        for i in range(len(N_new[0,0])):
            N_column = mpcalc.brunt_vaisala_frequency(ht[:,j,i],
                                                         tha[:,j,i])
            N_new[:,j,i] = N_column
    return N_new / units.s

# eady growth rate function
def layer_eady(f_col, N, u1, u2, v1, v2, z1, z2):
    dudz = (u1 - u2) / (z1 - z2)
    dvdz = (v1 - v2) / (z1 - z2)
    eady_out = 0.31 * f_col * np.sqrt(dudz**2 + dvdz**2) / N
    return eady_out
# function to take a column average, not weighted at boundaries
def vertical_av_ma(f):
    m1 = (f.m > 1.0e-3) | np.isnan(f)
    f_cleaned = np.ma.masked_array(f, m1)
    f_out = np.ma.average(f_cleaned, axis=0)
    return f_out

# function to take a column average, weighted 1/2 at boundaries
def vertical_av_wt(f):
    f_cleaned = np.ma.masked_array(f, np.isnan(f))
    wt = np.ones(f_cleaned.shape)
    f_out = np.ma.average(f_cleaned, axis=0, weights = wt)
    return f_out

# function that converts from sigma to pressure coordinates, then calculate pv
def sigma2pv(p, h, th, u, v, dx, dy, l):
    plevs = np.arange(1000., 70., -25.) * units.hPa
    f = mpcalc.coriolis_parameter(np.deg2rad(l))
    h_iso, thta_int, u_int, v_int = log_interpolate_1d(plevs, p, h, th, u, v, axis=0)
    pv_full = mpcalc.potential_vorticity_baroclinic(thta_int, plevs[:,None,None,None], u_int, v_int, dx[None,:], dy[None,:], np.deg2rad(l))
    return plevs.m, pv_full, h_iso, thta_int, f, u_int, v_int

dy_var, dx_var = create_deltas(FILE_LISTS+'r000_grid.nc', 111000*units.m) # 1 degree approx. 111 km
scale_y = 111000*units.m # constant spacing for 1 degree
scale_x = np.mean(dx_var[m1y,m1x-1:m1x+1]) # estimated constant spacing

pres_final, pv_all, ht_all, thta_all, f_cor, u_all, v_all = sigma2pv(pres, hght, thta, u_grid, v_grid, dx_var, dy_var, ny)
del pres, hght, thta, u_grid, v_grid, dx_var, dy_var

# B-V frequency
N_all, N_sq = N_levels_sq(ht_all, thta_all, pv_all)
N_lo = vertical_av_wt(N_all[20:29]) #850-500 MB
N_up = vertical_av_wt(N_all[6:21]) #500-300 MB
N_col = vertical_av_ma(N_sq)

# Upper and Lower PV
pv_lower = vertical_av_wt(pv_all[:17]) #1000-600 MB
pv_upper = vertical_av_wt(pv_all[16:33]) #600-200 MB

# Upper and Lower EADY
e_lower = layer_eady(f_cor, N_lo, u_all[20], u_all[6], v_all[20], v_all[6], ht_all[20], ht_all[6])
e_upper = layer_eady(f_cor, N_up, u_all[28], u_all[20], v_all[28], v_all[20], ht_all[28], ht_all[20])

del pv_all, ht_all, thta_all, u_all, v_all

# climatological means
# re-calculate N2 using 850 mb theta as estimate of mean theta
def recalculate_N2(N_arr, T_arr0, offset):
    # N2 = (g/th) * dthdz
    th_arr0 = mpcalc.potential_temperature(850.*units.millibar, T_arr0)
    T_arr1 = T_arr0 + offset
    th_arr1 = mpcalc.potential_temperature(850.*units.millibar, T_arr1)
    gdthdz = N_arr * th_arr0 # the constant
dN2dth = -1. * (th_arr1**-2) * gdthdz # change in N2 wrt th
    return N_arr + (dN2dth * offset)

def recalculate_eady(E_arr, N_arr0, N_arr1):
    # EADY = (constants) / N not N2
    const = E_arr * np.sqrt(N_arr0)
dEdN = -1. * const / N_arr1
dE = dEdN * (np.sqrt(N_arr1) - np.sqrt(N_arr0))
    return E_arr + dE

if t_adj == 0:
    N_climo_m = read_var('/home/py84/research_new/acm/C1225_12', 'NSQ_TROPO_AV', 0)[0,j1:j2+1,i1:i2+1] / (units.s**2)
eal_climo_m = read_var('/home/py84/research_new/acm/C1225_12', 'EADY_AV', 0)[0,j1:j2+1,i1:i2+1]
eau_climo_m = read_var('/home/py84/research_new/acm/C1225_12', 'EADYup_AV', 0)[0,j1:j2+1,i1:i2+1]
else:
    t850_30day = read_var('/home/py84/research_new/t_pert_era/C1225_12_T30d.nc', 'temperature', None)[0,:2*ym+1,:2*xm+1] * units.K
    N_climo_orig = read_var('/home/py84/research_new/acm/C1225_12', 'NSQ_TROPO_AV', 0)[0,j1:j2+1,i1:i2+1] / (units.s**2)
    N_climo_m = recalculate_N2(N_climo_orig, t850_30day, 'EADY_AV', 0)[0,j1:j2+1,i1:i2+1]
eal_climo_m = read_var('/home/py84/research_new/acm/C1225_12', 'EADY_AV', 0)[0,j1:j2+1,i1:i2+1]
eau_climo_m = read_var('/home/py84/research_new/acm/C1225_12', 'EADYup_AV', 0)[0,j1:j2+1,i1:i2+1]
eau_climo_m = recalculate_eady(eal_climo_orig, N_climo_orig, N_climo_m)

# standard deviations
N_climo_sd = read_var('/home/py84/research_new/acm/C1225_12', 'NSQ_TROPO_STD', 0)[0,j1:j2+1,i1:i2+1]
eal_climo_sd = read_var('/home/py84/research_new/acm/C1225_12', 'EADY_STD', 0)[0,j1:j2+1,i1:i2+1]
eau_climo_sd = read_var('/home/py84/research_new/acm/C1225_12', 'EADYup_STD', 0)[0,j1:j2+1,i1:i2+1]
pv1_climo_sd = read_var('/home/py84/research_new/acm/C1225_12', 'PVlow_AV', 0)[0,j1:j2+1,i1:i2+1]
pvu_climo_sd = read_var('/home/py84/research_new/acm/C1225_12', 'PVup_AV', 0)[0,j1:j2+1,i1:i2+1]

# normalized values
\[ N_{\text{norm}} = \frac{(N_{\text{col}} - N_{\text{climo.m.m}})}{N_{\text{climo.sd}}} \]

\[ e_{\text{al.norm}} = \frac{(e_{\text{lower.m}} - e_{\text{al.climo.m}})}{e_{\text{al.climo.sd}}} \]

\[ e_{\text{au.norm}} = \frac{(e_{\text{upper.m}} - e_{\text{au.climo.m}})}{e_{\text{au.climo.sd}}} \]

\[ p_{\text{vl.norm}} = \frac{((p_{\text{v_lower}} \times 1.0e6) - p_{\text{vl.climo.m}})}{p_{\text{vl.climo.sd}}} \]

\[ p_{\text{vu.norm}} = \frac{((p_{\text{v_upper}} \times 1.0e6) - p_{\text{vu.climo.m}})}{p_{\text{vu.climo.sd}}} \]

# functions to calculate radial distances

def dist_from_center(sc_y, sc_x, y_len, x_len, ymid, xmid):
    y_dist, x_dist = np.ogrid[0:y_len,0:x_len]
    dist2 = np.linalg.norm((sc_y*(y_dist-ymid), sc_x*(x_dist-xmid)),axis=0)
    return dist2

# sort normalized values within specific distance from center

def norm2med(arr, sy, sx, lim, use_top = True):
    ay, ax = arr.shape
    b = arr.copy()
    d_arr = dist_from_center(sy, sx, ay, ax, m1y, m1x)
    m2 = np.zeros_like(arr, dtype=bool)
    m2[d_arr > lim] = 1.
    #print(m2)
    b = np.ma.array(b, mask=m2)
    a_out = b[b.mask==False]
    m_ind = len(a_out)//2
    #print(m_ind)
    a_out = np.sort(a_out)
    if use_top: # top half of median
        pc_val = np.mean(a_out[m_ind:]
    else: # bottom half of median
        pc_val = np.mean(a_out[0:m_ind+1])
    return pc_val

N_sort = norm2med(N_norm, scale_y, scale_x, 500000*units.m, False)

eal_sort = norm2med(eal_norm, scale_y, scale_x, 500000*units.m)

neau_sort = norm2med(eau_norm, scale_y, scale_x, 500000*units.m)
pvl_sort = norm2med(pvl_norm, scale_y, scale_x, 500000*units.m)
pvu_sort = norm2med(pvu_norm, scale_y, scale_x, 500000*units.m)

# adding normalized values to precursor list, use pandas

df = pd.read_csv('precursors_case0.csv', sep=',', header=None, names=[None, 'T+0K', 'T-5K', 'T+1K', 'T+5K', 'T+10K'], index_col=0)
df.at['nstd_PVup.p5u.1',t_head] = pvu_sort

df.at['nstd_PVlow.p5u.1',t_head] = pvl_sort

df.at['nstd_NSQ_TROPO.p5l.0',t_head] = N_sort

df.at['nstd_EADYup.p5u.1',t_head] = eau_sort

df.at['nstd_EADY.p5u.1',t_head] = eal_sort

print(df)

df.iloc[df.notnull().any().values].to_csv('precursors_case0.csv', header=False) # only includes columns if at least 1 value isn't NaN

# plotting
bvl = np.arange(0.3, 3.0, 0.1)
n1 = np.arange(-3., 3.1, 0.2)
efl = np.arange(0., 2.5, 0.1)
pvfl = np.arange(0., 3.6, 0.25)

def draw_map(dataset1, l, x_axis, y_axis, str_ver, cm, e, i1, i2, n):
    ax = plt.subplot(i1, i2, n)
    m = Basemap(llcrnrlon = x_axis[0], llcrnrlat = y_axis[0],
                urcrnrlon = x_axis[-1], urcrnrlat = y_axis[-1], \
                lat_ts = y_axis[ym], resolution = 'l', area_thresh = 100.,
                projection = 'merc')
    nx, ny = np.meshgrid(x_axis, y_axis)
    x, y = m(nx, ny)
    x_point, y_point = m(x_axis[xm], y_axis[ym1])
    parallels = np.arange(10., 65., 2.)
    meridians = np.arange(-125., -9., 4.)
    m.drawparallels(parallels, labels = [1,0,0,0])
    m.drawmeridians(meridians, labels = [0,0,0,1])
    m.drawcoastlines()
    m.drawcountries()
    m.drawstates()
    cf = plt.contourf(x, y, dataset1, l, cmap = cm, extend=e)
    plt.colorbar(cf, shrink = 0.6)
    m.plot(x_point, y_point, marker='$\mathrm{L}$', color='r',
           markersize=18)
    ax.set_title(str_ver, fontsize = 11)
    return ax

figw, figh = 9,7
plt.figure(figsize=[figw, figh])
map3 = draw_map(pv_lower*1.0e6, bvl, lons, lats, '1000-600MB PV (PVU)',
                'gist_ncar', 'max', 1, 1, 1)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
                    top=1-0.4/figh)
plt.savefig('fin'+ra+'_01_01_pv-low.png', dpi=60)

plt.figure(figsize=[figw, figh])
map3 = draw_map(pv_upper*1.0e6, bvl, lons, lats, '600-200MB PV (PVU)',
                'gist_ncar', 'max', 1, 1, 1)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
                    top=1-0.4/figh)
plt.savefig('fin'+ra+'_01_02_pv-up.png', dpi=60)

plt.figure(figsize=[figw, figh])
map2 = draw_map(e_lower*1.0e5, bvl, lons, lats, 'Lower EADY (10$^{-5}$s$^{-2}$)',
                'brg', 'max', 1, 1, 1)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
                    top=1-0.4/figh)
plt.savefig('fin'+ra+'_01_03_eady.png', dpi=60)

plt.figure(figsize=[figw, figh])
map2 = draw_map(e_upper*1.0e5, bvl, lons, lats, 'Upper EADY (10$^{-5}$s$^{-2}$)',
                'brg', 'max', 1, 1, 1)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
                    top=1-0.4/figh)
plt.savefig('fin'+ra+'_01_04_eady-up.png', dpi=60)

plt.figure(figsize=[figw, figh])
map1 = draw_map(N_col*1.0e4, bvl, lons, lats, 'Trop. $N^2$ ($10^{-4}$ s$^{-2}$)', 'viridis', 'both', 1, 1, 1)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh, top=1-0.4/figh)
plt.savefig('fin'+ra+'_02_01_n2.png', dpi=60)
# takes about 39 sec to run
# file: center_veljet.py
import sys
import numpy as np
import netCDF4 as nc
import pandas as pd
import metpy.calc as mpcalc
import metpy.constants as constants
from metpy.interpolate import log_interpolate_1d
from metpy.units import units
from metpy.plots import convert_gempak_color
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap

print("done import")

def read_var(NCfile, varname, hr):
    fid = nc.Dataset(NCfile, 'r')
    var_out = fid.variables[varname][:]
    var_time = var_out[hr]
    fid.close()
    return var_time

# array averaging function similar to numpy.diff function (arr[n] +
# arr[n+1])

def av_ar(a, n=1, axis=-1):
    a = np.asanyarray(a)
    nd = a.ndim
    axis = np.core.multiarray.normalize_axis_index(axis, nd)
    slice1 = [slice(None)] * nd
    slice2 = [slice(None)] * nd
    slice1[axis] = slice(1, None)
    slice2[axis] = slice(None, -1)
    slice1 = tuple(slice1)
    slice2 = tuple(slice2)
    op = not_equal if a.dtype == np.bool_ else np.add
    for _ in range(n):
        a = op(a[slice1], a[slice2]) / 2.
    return a

# finding location on global grid

def combine_arr(lat1, lat2, lon1, lon2):
    lat_bool = np.in1d(lat1, lat2)
    lon_bool = np.in1d(lon1, lon2)
    return np.nonzero(lat_bool)[0][0], np.nonzero(lat_bool)[0][-1],
    np.nonzero(lon_bool)[0][0], np.nonzero(lon_bool)[0][-1]

try:
    t_adj, ym1, xm = float(sys.argv[1]), int(sys.argv[2]),
    int(sys.argv[3])  # (6, 8) for all except +10 (4, 9)
    print(t_adj, ym1, xm)
except:
    print('error: missing inputs')
sysexit()

FILE_LISTS = '/home/py84/research_new/
if t_adj == -5.:
    ra = '105'
    fname_grid = FILE_LISTS+'r105_grid.nc'.format(t_adj)
fname_uwnd = FILE_LISTS+'r105_uwnd.nc'.format(t_adj)
fname_vwnd = FILE_LISTS+'r105_vwnd.nc'.format(t_adj)
t_head = 'T-5K'
else:
    ra = '{:03}'.format(int(t_adj))
fname_grid = FILE_LISTS+'r{}_grid.nc'.format(ra)
fname_uwnd = FILE_LISTS+'r{}_uwnd.nc'.format(ra)
fname_vwnd = FILE_LISTS+'r{}_vwnd.nc'.format(ra)
t_head = 'T+{}K'.format(int(t_adj))

# forecast hour 12
h_ind = 12
# use same size gridbox for all the runs, only modifying center marker
ym = 6

lat_tot = np.arange(-90., 91., 1.)
lon_tot = np.arange(-180., 181., 1.)
# on the regridded 1-deg data (lat is len 30, lon is len 49)
# i-10:i+11 is preferred if not out of bounds; otherwise, use :2*i+1
lats = read_var(FILE_LISTS+'r000_grid.nc', 'lat', None)[0,:2*ym+1]
lons = read_var(FILE_LISTS+'r000_grid.nc', 'lon', None)[0,:2*xm+1]
print(lats.shape, lons.shape)
# midpoints after slicing
m1y, m1x = len(lats)//2, len(lons)//2 # use if i-10:i+11
m1y, m1x = ym1, xm # use if 0:2*i+1; ym/xm not necessarily midpoints
j1, j2, i1, i2 = combine_arr(lat_tot, lats, lon_tot, lons)
u_grid = read_var(fname_uwnd, 'U', h_ind)[j1:j2, i1:i2] * units.m / units.s
v_grid = read_var(fname_vwnd, 'V', h_ind)[j1:j2, i1:i2] * units.m / units.s
print("u, v", u_grid.shape, v_grid.shape)

# pressure
def calc_pres(fid_name, hr):
p0 = read_var(fid_name, 'PB', hr)[j1:j2, i1:i2] * units.Pa
p1 = read_var(fid_name, 'P', hr)[j1:j2, i1:i2] * units.Pa
pres_pa = p0 + p1
return pres_pa.to('hPa')
pres = calc_pres(fname_grid, h_ind)
print(pres.shape)

# geopotential, staggered on edge of grid
def calc_hght(fid_name, hr):
gvty0 = read_var(fid_name, 'PHB', hr)[j1:j2, i1:i2] * units.m
gvty1 = read_var(fid_name, 'PH', hr)[j1:j2, i1:i2] * units.m
geopot0 = gvty0 + gvty1 # geopotential in m2/s2
geopot1 = av_ar(geopot0, axis=0) * units.m**2 / units.s**2
h = mpcalc.geopotential_to_height(geopot1)
return h
hght = calc_hght(fname_grid, h_ind)
print(hght.shape)
print("done base variables")

# do the interpolation
def sigma2pres(p, h, u, v):
    plevs = np.arange(500., 99., -50.) * units.hPa
heights, u_out, v_out = log_interpolate_1d(plevs, p, h, u, v, axis=0)
print(heights.shape, u_out.shape)
return plevs, heights, u_out, v_out

pres_final, hght_iso, u_final, v_final = sigma2pres(pres, hght, u_grid, v_grid)
del pres, hght, u_grid, v_grid
print("done interpolate")

sp_m = mpcalc.wind_speed(u_final, v_final)
print(sp_m.shape)
print(sp_m[-1,4:9,17:22])

# loop to calculate pressure-weighted mean for each column of data
def mpw(p, arr, h):
    print(arr.units)
    w_av = np.empty_like(arr[0])
    for j in range(len(w_av)):
        for i in range(len(w_av[0])):
            a = mpcalc.mean_pressure_weighted(p, arr[:,j,i], heights=h[:,j,i])[0]
            w_av[j,i] = a.m
    return w_av * arr.units

vj = mpw(pres_final, sp_m, hght_iso)

# climatological mean and standard deviation
vj_climo_m = read_var('/home/py84/research_new/acm/C1225_12', 'VELJET_AV', 0)[0,j1:j2+1,i1:i2+1] * units.m / units.s
vj_climo_sd = read_var('/home/py84/research_new/acm/C1225_12', 'VELJET_STD', 0)[0,j1:j2+1,i1:i2+1] * units.m / units.s

# normalized value
vj_norm = (vj - vj_climo_m) / vj_climo_sd

# variable grid spacing, using map factors
def create_deltas(fid_name, dx_in): # assume same grid spacing for both x and y directions
    m_file = read_var(fid_name, 'MAPFAC_M', 0)[:2*ym+1,:2*xm+1]
    map_x = av_ar(m_file, axis=1)
    map_y = av_ar(m_file, axis=0)
    print(map_x.shape, map_y.shape)
    dx_out = dx_in / map_x
    dy_out = dx_in / map_y
    return dy_out, dx_out # following same order convention as wrf

output

dy, dx = create_deltas(FILE_LISTS+'r000_grid.nc', 111000*units.m) # 1 degree approx. 111 km
scale_y = 111000*units.m # constant spacing for 1 degree
scale_x = np.mean(dx[m1y,m1x-1:m1x+1]) # estimated constant spacing

# functions to calculate radial distances
def dist_from_center(sc_y, sc_x, y_len, x_len, ymid, xmid):
    y_dist, x_dist = np.ogrid[0:y_len,0:x_len]
\[
dist2 = \text{linalg.norm}((\text{sc}_y \cdot (y_{\text{dist}} - y_{\text{mid}}), \text{sc}_x \cdot (x_{\text{dist}} - x_{\text{mid}})), \text{axis}=0)
\]
return dist2

# sort normalized values within specific distance from center
def norm2med(arr, sy, sx, lim, use_top = True):
    ay, ax = arr.shape
    b = arr.copy()
    d_arr = dist_from_center(sy, sx, ay, ax, m1y, m1x)
    m2 = np.zeros_like(arr, dtype=\text{bool})
    m2[d_arr > lim] = 1.
    #print(m2)
    b = np.ma.array(b, mask=m2)
    a_out = b[b.mask==\text{False}]
    m_ind = len(a_out)//2
    #print(m_ind)
    a_out = np.sort(a_out)
    if use_top: # top half of median
        pc_val = np.mean(a_out[m_ind:])
    else: # bottom half of median
        pc_val = np.mean(a_out[0:m_ind+1])
    return pc_val

vj_sort = norm2med(vj_norm, scale_y, scale_x, 500000*\text{units.m})
print(vj_sort)

# use pandas to read and open files
def read_csv('precursors_case0.csv', sep=',', header=None, names=[\text{None}, 'T+0K', 'T-5K', 'T+1K', 'T+5K', 'T+10K'], index_col=0):
df.at['nstd_VELJET.p5u.0',t_head] = vj_sort
print(df)

levs = np.arange(0., 46., 5.)
l2 = np.arange(-2., 2.1, 0.2)
cm1 = convert_gempak_color([32,21,22,23,6,25,4,28,29,30])

def draw_map(dataset1, x_axis, y_axis, str_ver, n):
    print(n)
    ax = plt.subplot(1, 1, n)
    m = Basemap(llcrnrlon = x_axis[0], llcrnrlat = y_axis[0],
                urcrnrlon = x_axis[-1], urcrnrlat = y_axis[-1],
                lat_ts = y_axis[ym], resolution = 'l', area_thresh = 100.,
                projection = 'merc')
    nx, ny = np.meshgrid(x_axis, y_axis)
    x, y = m(nx, ny)
    x_point, y_point = m(x_axis[xm], y_axis[ym1])
    parallels = np.arange(10., 65., 2.)
    meridians = np.arange(-125., -9., 4.)
    m.drawparallels(parallels, labels = [1,0,0,0])
    m.drawmeridians(meridians, labels = [0,0,0,1])
    m.drawcoastlines()
    m.drawcountries()
    m.drawstates()
    if n == 1:
        cs = plt.contourf(x, y, dataset1, levs, colors = cm1,
                          extend='max')
    else:
cs = plt.contourf(x, y, dataset1, l2, cmap = 'coolwarm',
    extend='both')
plt.colorbar(cs, shrink = 0.6)
m.plot(x_point, y_point, marker='$\text{L}$', color='r',
    markersize=18)
ax.set_title(str_ver, fontsize = 11)
return ax

figw, figh = 9,8
plt.figure(figsize=[figw, figh])
map1 = draw_map(vj, lons, lats, "500-100 hPa Mean Wind (m/s)", 1)
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
    top=1-0.4/figh)
plt.savefig('fin'+ra+'_06_04_jet.png',dpi=60)
# file: pwat_cape.py
#export PYTHONPATH=/home/py84/MetPy
import sys
import metpy
import itertools
import pandas as pd
print(metpy.__version__) # edited version to fix bug
import numpy as np
import netCDF4 as nc
import metpy.calc as mpcalc
import metpy.constants as constants
from metpy.units import units
import matplotlib as mpl
#mpl.use('Agg')
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
print("done import")

# array averaging function similar to numpy.diff function (arr[n] +
arr[n+1])
def av_ar(a, n=1, axis=-1):
    a = np.asanyarray(a)
    nd = a.ndim
    axis = np.core.multiarray.normalize_axis_index(axis, nd)
    slice1 = [slice(None)] * nd
    slice2 = [slice(None)] * nd
    slice1[axis] = slice(1, None)
    slice2[axis] = slice(None, -1)
    slice1 = tuple(slice1)
    slice2 = tuple(slice2)
    op = not_equal if a.dtype == np.bool_ else np.add
    for _ in range(n):
        a = op(a[slice1], a[slice2]) / 2.
    return a

def read_var(NCfile, varname, hr):
    fid = nc.Dataset(NCfile, 'r')
    var_out = fid.variables[varname][:]
    var_time = var_out[hr]
    fid.close()
    return var_time

# finding location on global grid
def combine_arr(lat1, lat2, lon1, lon2):
    lat_bool = np.in1d(lat1, lat2)
    lon_bool = np.in1d(lon1, lon2)
    return np.nonzero(lat_bool)[0][0], np.nonzero(lat_bool)[0][-1],
          np.nonzero(lon_bool)[0][0], np.nonzero(lon_bool)[0][-1]

try:
    t_adj, ym1, xm = float(sys.argv[1]), int(sys.argv[2]),
    int(sys.argv[3]) # (6, 8) for all except +10 (4, 9)
    print(t_adj, ym1, xm)
except:
    print('error: missing inputs')
sys.exit()
FILE_LISTS = '/home/py84/research_new/'
if t_adj == -5.:
    t_head = 'T-5K'
    ra = '105'
    f_input = FILE_LISTS+'r105_grid.nc'.format(t_adj)
else:
    ra = '{:03}'.format(int(t_adj))
    t_head = 'T+{}K'.format(int(t_adj))
    f_input = FILE_LISTS+'r{}_grid.nc'.format(ra)

# print(ra, t_head)
# forecast hour
h_ind = 12

# use same size gridbox for all the runs, only modifying center marker
ym = 6

lat_tot = np.arange(-90., 91., 1.)
lon_tot = np.arange(-180., 181., 1.)

# on the regridded 1-deg data (lat is len 30, lon is len 49)
# i-10:i+11 is preferred if not out of bounds; otherwise, use :2*i+1
lats = read_var(FILE_LISTS+'r000_grid.nc', 'lat', None)[0,:2*ym+1]

lons = read_var(FILE_LISTS+'r000_grid.nc', 'lon', None)[0,:2*xm+1]

print(lats.shape, lons.shape)

# midpoints after slicing
# m1y, m1x = len(lats)//2, len(lons)//2 # use if i-10:i+11
m1y, m1x = ym1, xm # use if 0:2*i+1; ym/xm not necessarily midpoints

j1, j2, i1, i2 = combine_arr(lat_tot, lats, lon_tot, lons)

# pressure

def calc_pres(fid_name, hr):
    p0 = read_var(fid_name, 'PB', hr)[::,2*ym+1,:2*xm+1] * units.Pa
    p1 = read_var(fid_name, 'P', hr)[::,2*ym+1,:2*xm+1] * units.Pa

    pres_pa = p0 + p1
    return pres_pa.to('hPa')

press = calc_pres(f_input, h_ind)

print(press.shape)

# temps (from pot temp)

def calc_temp(fid_name, p, hr):
    tmp0 = (read_var(fid_name, 'T', hr) + 300.0)[::,2*ym+1,:2*xm+1] * units.K

    T = mpcalc.temperature_from_potential_temperature(p, tmp0)
    return T

temps_K = calc_temp(f_input, press, h_ind)

print(temps_K.shape)

# dewpoint for moisture

def calc_dp(fid_name, p_wanted, hr):
    w_base = read_var(fid_name, 'QVAPOR', h_ind)[::,2*ym+1,:2*xm+1] * units.dimensionless
    e_part = mpcalc.vapor_pressure(p_wanted, w_base)
    return mpcalc.dewpoint(e_part)

dp = calc_dp(f_input, press, h_ind)

print("done base variables")

# precipitable water and cape functions for each column of data

def pw_full(p, T, m):
    pw_out = np.empty_like(m[0])
    cape_out = np.empty_like(m[0])
for j in range(len(pw_out)):
    for i in range(len(pw_out[0])):
        print(j, i)
        a = mpcalc.precipitable_water(m[:,j,i], pres[:,j,i])
        mp, mt, md = mpcalc.mixed_parcel(pres[:,j,i], T[:,j,i],
                                     m[:,j,i], interpolate=False)
        print(mt.to('degC'), md)
        pres[0,j,i], T[0,j,i], m[0,j,i] = mp, mt, md
        P2, T2, Td2, ml_profile =
            mpcalc.parcel_profile_with_lcl(pres[:,j,i], T[:,j,i], m[:,j,i])
        ca, ci = mpcalc.cape_cin(P2, T2, Td2, ml_profile)
        pw_out[j,i] = a.m
        cape_out[j,i] = ca.m
    return pw_out * units.mm, cape_out

pwat, cape = pw_full(pres, temps_K, dp)
print(cape[4:9,8:13])
print(pwat[4:9,8:13])

ca_climo_m = read_var('/home/py84/research_new/acm/C1225_12',
                      'ML_CAPE_AV', 0)[0,j1:j2+1,i1:i2+1]
pw_anom_m = read_var('/home/py84/research_new/acm/C1225_12', 'Qint_AV',
                     0)[0,j1:j2+1,i1:i2+1] * units.mm

ca_climo_sd = read_var('/home/py84/research_new/acm/C1225_12',
                       'ML_CAPE_STD', 0)[0,j1:j2+1,i1:i2+1]
pw_anom_sd = read_var('/home/py84/research_new/acm/C1225_12',
                      'Qint_STD', 0)[0,j1:j2+1,i1:i2+1] * units.mm

cape_norm = (cape - ca_climo_m) / ca_climo_sd
pwat_norm = (pwat - pw_anom_m) / pw_anom_sd

# normalized values
ca_climo_m = read_var(fid_name, 'MAPFAC_M', 0)[0,j1:j2+1,i1:i2+1]
map_x = av_ar(m_file, axis=1)
map_y = av_ar(m_file, axis=0)
print(map_x.shape, map_y.shape)
dx_out = dx_in / map_x
dy_out = dx_in / map_y
return dy_out, dx_out

def create_deltas(fid_name, dx_in): # assume same grid spacing for both
    m_file = read_var(fid_name, 'MAPFAC_M', 0)[2*ym+1,2*xm+1]
    map_x = av_ar(m_file, axis=1)
    map_y = av_ar(m_file, axis=0)
    print(map_x.shape, map_y.shape)
dx_out = dx_in / map_x
    dy_out = dx_in / map_y
    return dy_out, dx_out

scale_y = 111000*units.m # constant spacing for 1 degree
scale_x = np.mean(dx[m1y,m1x-1:m1x+1]) # estimated constant spacing
# functions to calculate radial distances

def dist_from_center(sc_y, sc_x, y_len, x_len, ymid, xmid):
    y_dist, x_dist = np.ogrid[0:y_len,0:x_len]
    dist2 = np.linalg.norm((sc_y*(y_dist-ymid), sc_x*(x_dist-xmid)),axis=0)
    return dist2

# sort normalized values within specific distance from center

def norm2med(arr, sy, sx, lim, use_top = True, log_trans = False):
    ay, ax = arr.shape
    d_arr = dist_from_center(sy, sx, ay, ax, m1y, m1x)
    m2[d_arr > lim] = 1.
    #print(m2)
    b = np.ma.array(b, mask=m2)
    a_out = b[b.mask==False]
    m_ind = len(a_out)//2
    #print(m_ind)
    a_out = np.sort(a_out)
    if use_top: # top half of median
        if log_trans:
            a_up = np.log(a_out[m_ind:])
        else:
            a_up = a_out[m_ind:]
        pc_val = np.mean(a_up)
    else: # bottom half of median
        pc_val = np.mean(a_out[0:m_ind+1])
    return pc_val

cape_sort = norm2med(cape_norm, scale_y, scale_x, 500000*units.m,
                      log_trans=True)
pwat_sort = norm2med(pwat_norm, scale_y, scale_x, 500000*units.m)

# use pandas to open and assign values

df = pd.read_csv('precursors_case0.csv', sep=',', header=None,
names=[None, 'T+0K', 'T-5K', 'T+1K', 'T+5K', 'T+10K'], index_col=0)
df.at['nstd_ML_CAPE.p5u.0',t_head] = cape_sort
df.at['nstd_Qint.p5u.0',t_head] = pwat_sort
print(df)
df.iloc[:,df.notnull().any().values].to_csv('precursors_case0.csv',
header=False) # only includes columns if at least 1 value isn't NaN

if t_adj < 5:
    l1 = np.arange(0., 41., 2.)
    l2 = [0.,10.,20.,30.,40.,50.,60.,70.,80.,90.,100.,1500.]
elif t_adj == 5:
    l1 = np.arange(0., 51., 2.)
    l2 = [0.,10.,20.,30.,40.,50.,60.,70.,80.,90.,100.,1500.]
else:
    l1 = np.arange(0., 51., 2.)
l2 = [0., 5., 10., 20., 50., 100., 200., 300., 400., 500., 600., 700., 800., 900., 1000., 1500., 2000.]

def draw_map(dataset1, l, x_axis, y_axis, str_ver, cm):
    # Basemap setup
    ax = plt.subplot(1, 1, 1)
    m = Basemap(llcrnrlon = x_axis[0], llcrnrlat = y_axis[0],
               urcrnrlon = x_axis[-1], urcrnrlat = y_axis[-1],
               lat_ts = y_axis[ym], resolution = 'l',
               area_thresh = 100., projection = 'merc')
    nx, ny = np.meshgrid(x_axis, y_axis)
    x, y = m(nx, ny)
    x_point, y_point = m(x_axis[xm], y_axis[ym1])
    parallels = np.arange(10., 65., 2.)
    meridians = np.arange(-125., -9., 4.)
    m.drawparallels(parallels, labels = [1,0,0,0])
    m.drawmeridians(meridians, labels = [0,0,0,1])
    m.drawcoastlines()
    m.drawcountries()
    m.drawstates()
    cf = plt.contourf(x, y, dataset1, l, cmap = cm, extend='max')
    plt.colorbar(cf, shrink = 0.6)
    m.plot(x_point, y_point, marker='$\mathrm{L}$', color='r',
           markersize=18)
    ax.set_title(str_ver, fontsize = 11)
    return ax

figw, figh = 9, 7
plt.figure(figsize=[figw, figh])
map1 = draw_map(pwat, l1, lons, lats, 'precipitable water (mm)', 'jet')
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
                    top=1-0.4/figh)
plt.savefig('fin'+ra+'_02_04_qint.png',dpi=60)

plt.figure(figsize=[figw, figh])
map2 = draw_map(cape, l2, lons, lats, 'ML-CAPE (J/kg)', 'jet')
plt.subplots_adjust(left=0.8/figw, right=1-0.2/figw, bottom=0.4/figh,
                    top=1-0.4/figh)
plt.savefig('fin'+ra+'_03_01_mlcape.png',dpi=60)
**Part 2**

List of precursors and the calculated normalized anomalies for each case at forecast hour 12 (12Z on December 25).

<table>
<thead>
<tr>
<th>Precursor Name</th>
<th>control</th>
<th>T-5K</th>
<th>T+1K</th>
<th>T+5K</th>
<th>T+10K</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PV_{up}$</td>
<td>1.81563</td>
<td>1.70276</td>
<td>1.84718</td>
<td>2.08527</td>
<td>1.61391</td>
</tr>
<tr>
<td>$PV_{low}$</td>
<td>1.96405</td>
<td>1.44148</td>
<td>1.99675</td>
<td>2.30815</td>
<td>1.33912</td>
</tr>
<tr>
<td>EADY</td>
<td>1.20275</td>
<td>1.04345</td>
<td>1.16558</td>
<td>0.95750</td>
<td>0.74268</td>
</tr>
<tr>
<td>EADY$_{up}$</td>
<td>0.04327</td>
<td>0.38271</td>
<td>0.03191</td>
<td>-0.25530</td>
<td>-0.39499</td>
</tr>
<tr>
<td>$N^2_{TROPO}$</td>
<td>-1.76502</td>
<td>-1.99672</td>
<td>-1.73994</td>
<td>-1.51326</td>
<td>-1.25832</td>
</tr>
<tr>
<td>$FGEN_{850}$</td>
<td>-0.11562</td>
<td>-0.13761</td>
<td>-0.15057</td>
<td>-0.20587</td>
<td>-0.86353</td>
</tr>
<tr>
<td>DEF</td>
<td>3.47411</td>
<td>2.54813</td>
<td>3.77073</td>
<td>3.41439</td>
<td>3.58668</td>
</tr>
<tr>
<td>$Q_{int}$</td>
<td>0.54851</td>
<td>-0.21413</td>
<td>0.76794</td>
<td>1.64717</td>
<td>2.95346</td>
</tr>
<tr>
<td>$ML_CAPE$</td>
<td>-1.50509</td>
<td>-0.17954</td>
<td>-0.86679</td>
<td>0.85893</td>
<td>2.00694</td>
</tr>
<tr>
<td>RTOT</td>
<td>0.24484</td>
<td>-0.65315</td>
<td>0.19386</td>
<td>0.69502</td>
<td>0.28750</td>
</tr>
<tr>
<td>SKT</td>
<td>-0.70518</td>
<td>-0.55654</td>
<td>-0.73027</td>
<td>-0.84444</td>
<td>-1.01311</td>
</tr>
<tr>
<td>SLHF</td>
<td>-0.35361</td>
<td>-0.60142</td>
<td>-0.29951</td>
<td>-0.09128</td>
<td>0.24947</td>
</tr>
<tr>
<td>SSHF</td>
<td>-0.00173</td>
<td>0.04895</td>
<td>0.00284</td>
<td>-0.03172</td>
<td>-0.22957</td>
</tr>
<tr>
<td>THDIFF</td>
<td>-1.22266</td>
<td>-1.50595</td>
<td>-1.23285</td>
<td>-1.22899</td>
<td>-1.29282</td>
</tr>
<tr>
<td>TPERT</td>
<td>0.23029</td>
<td>0.25568</td>
<td>0.22455</td>
<td>0.13166</td>
<td>0.16754</td>
</tr>
<tr>
<td>$QG\omega_{BOT3}$</td>
<td>-0.76644</td>
<td>-0.66028</td>
<td>-0.77549</td>
<td>-0.74699</td>
<td>-0.60734</td>
</tr>
<tr>
<td>$QG\omega_{TOP3}$</td>
<td>-1.02528</td>
<td>-1.07753</td>
<td>-0.98355</td>
<td>-0.94466</td>
<td>-0.74756</td>
</tr>
<tr>
<td>$Z_{ANOM}$</td>
<td>-5.70471</td>
<td>-6.02584</td>
<td>-5.65117</td>
<td>-5.52138</td>
<td>-5.50930</td>
</tr>
<tr>
<td>$Z_{ORIG}$</td>
<td>-1.49764</td>
<td>-1.56810</td>
<td>-1.49278</td>
<td>-1.45794</td>
<td>-1.37146</td>
</tr>
<tr>
<td>$\nabla\theta_{850}$</td>
<td>1.38794</td>
<td>1.41518</td>
<td>1.41556</td>
<td>1.32420</td>
<td>0.48979</td>
</tr>
<tr>
<td>$\nabla\theta_{e850}$</td>
<td>0.62003</td>
<td>0.27094</td>
<td>0.73416</td>
<td>1.17955</td>
<td>1.60641</td>
</tr>
<tr>
<td>$\theta_{850}$</td>
<td>0.35092</td>
<td>0.81916</td>
<td>1.30650</td>
<td>1.73753</td>
<td>2.35000</td>
</tr>
<tr>
<td>$TADV_{850}$</td>
<td>0.40744</td>
<td>0.37639</td>
<td>0.42445</td>
<td>0.50417</td>
<td>0.36355</td>
</tr>
<tr>
<td>VELJET</td>
<td>-0.75127</td>
<td>-0.71818</td>
<td>-0.74713</td>
<td>-0.66817</td>
<td>-0.46924</td>
</tr>
</tbody>
</table>
Part 3: The 24 Precursors for all the experimental runs.

Figure A1: As in (Figure 12), but for the T-5K case.
Figure A2: As in (Figure 12), but for the T+1K case.
Figure A3: As in (Figure 12), but for the T+5K case.
Figure A4: As in (Figure 12), but for the T+10K case. Since the cyclone center is at a different point from the cyclone center in all the other runs, the “L” is not centered in the plots.
REFERENCES


