Toward an Open-Source, Python-Powered, Multi-Doppler Radar Analysis Suite

Timothy J. Lang, Christopher J. Schultz

Corey K. Potvin

Robert Jackson, Scott Collis, Jonathan Helmus

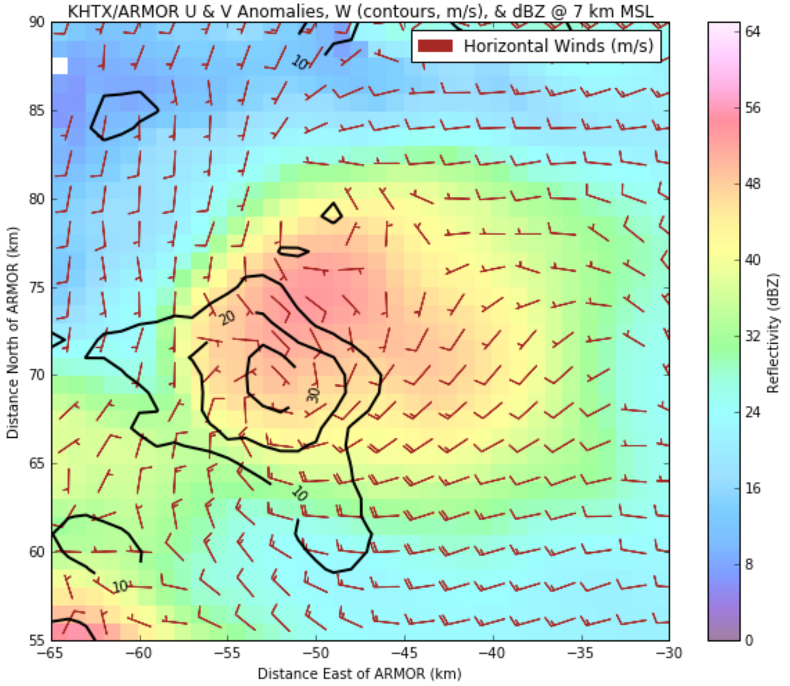
Brenda Dolan

NASA

CIMMS
Cooperative Institute for Mesoscale Meteorological Studies

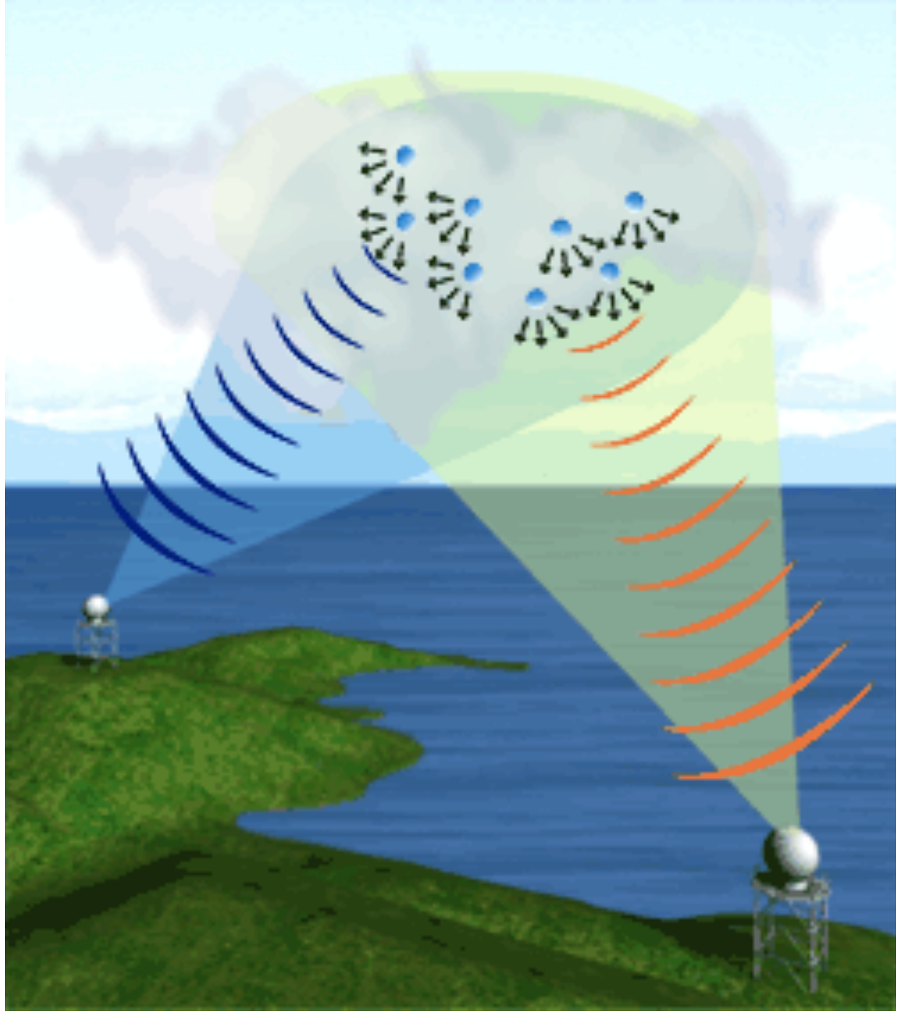
Argonne
NATIONAL LABORATORY

Colorado State University



Multi-Doppler Radar Concept

- Measure radial Doppler velocities with more than one radar, spaced appropriately (typically ~10s km)
- Coordinated spatial and temporal scanning
- Subtract hydrometeor fallspeed, remap to obtain horizontal U and V wind components
- Integrate mass-continuity equation to recover vertical velocity W



NICT Okinawa

The Context

- NASA Weather program (under Tsengdar Lee) seeks to improve NASA severe weather observational and modeling capabilities - NASA STORM project, FY 2016

- Independent but parallel effort to VORTEX-Southeast

- Three Main Goals

  1. Expansion of North Alabama Lightning Mapping Array (NALMA)

  2. Advanced ensemble model severe weather forecasting

  3. Expand open-source tools for severe weather analysis

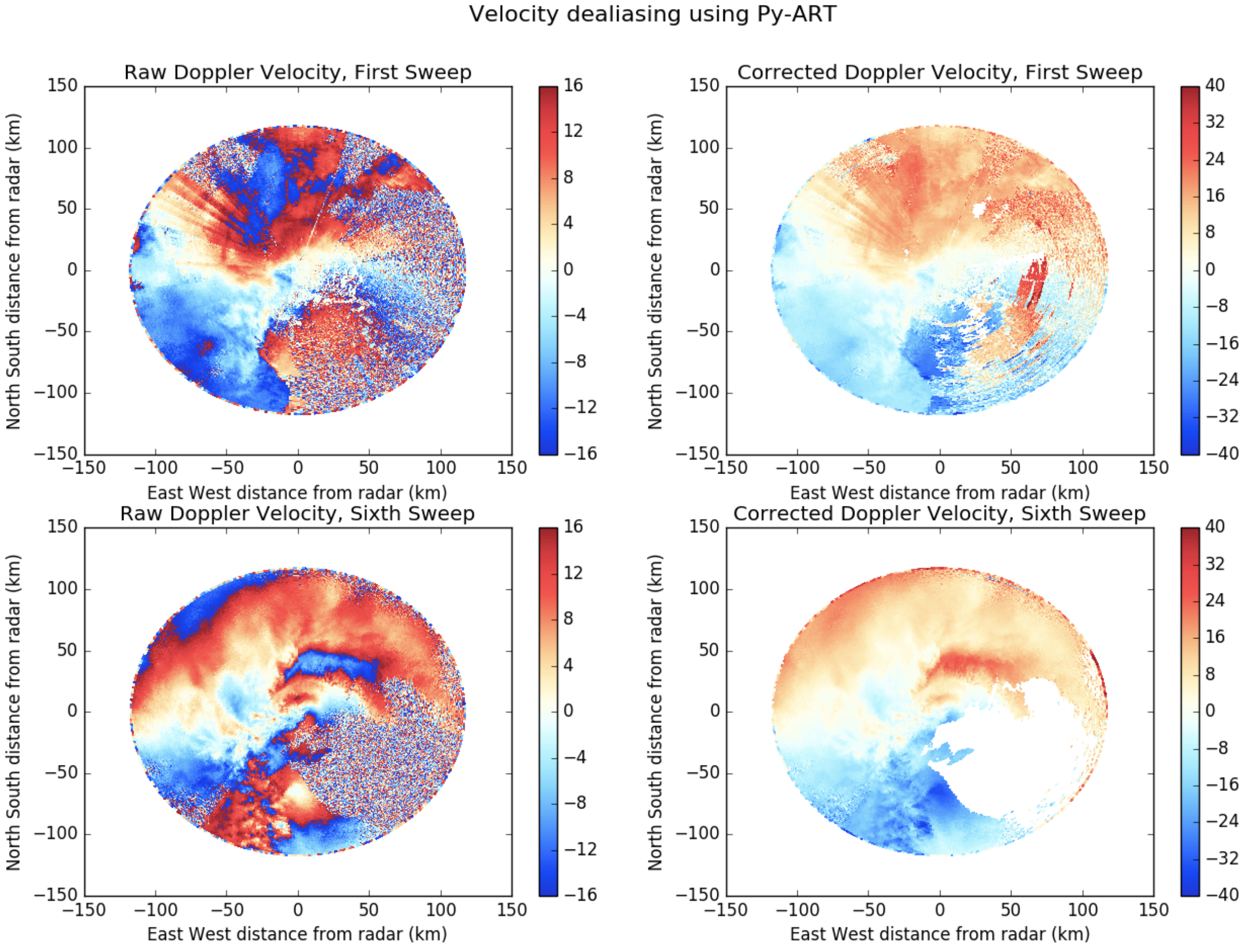
The Dream

Wouldn’t it be nice to have an open-source, Python-based toolkit for multi-Doppler wind syntheses?

- Three-dimensional winds from arbitrary radar networks
- Enable community-supported severe weather analyses
- Significantly lower barrier to entry for new users

Realizing the Dream, Part I - Python ARM Radar Toolkit (Py-ART)

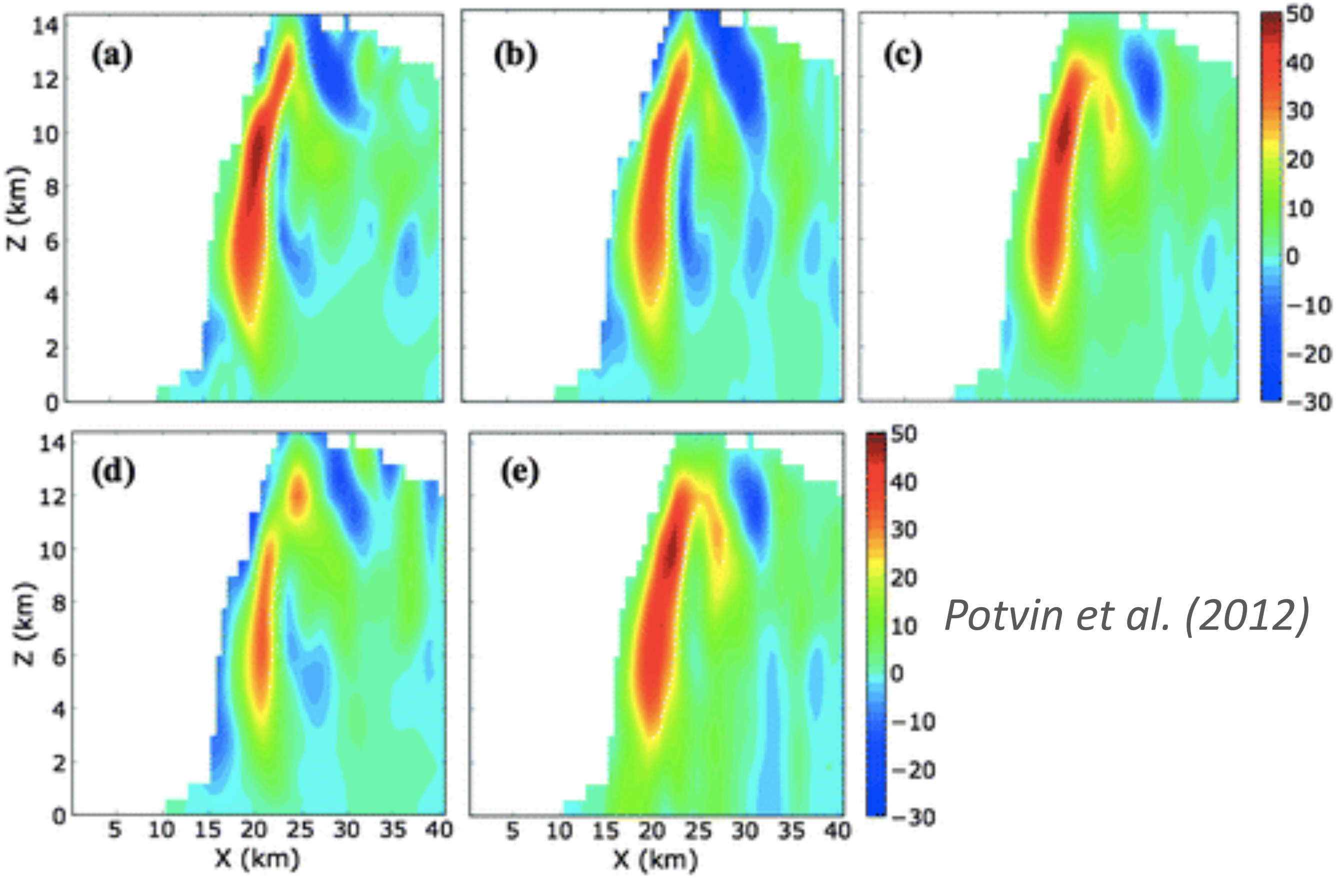
- Developed at Argonne National Lab
- Simplified File I/O
- Facilitates filtering via GateFilter object
- Automated Doppler velocity dealiasing
- Interpolating to a Cartesian grid
- Display of spherical and gridded data
- Advection correction under development



https://arm-doe.github.io/pyart/dev/auto_examples/index.html

Realizing the Dream, Part II - DDA C Application

- "Dual-Doppler Analysis" Developed at OU/CIMMS
- Based on 3D Variational Analysis (3DVAR)
- Mass conservation constraint becomes a tunable parameter
- Also tunable: Vorticity, Smoothness, Sounding weights



Potvin et al. (2012)

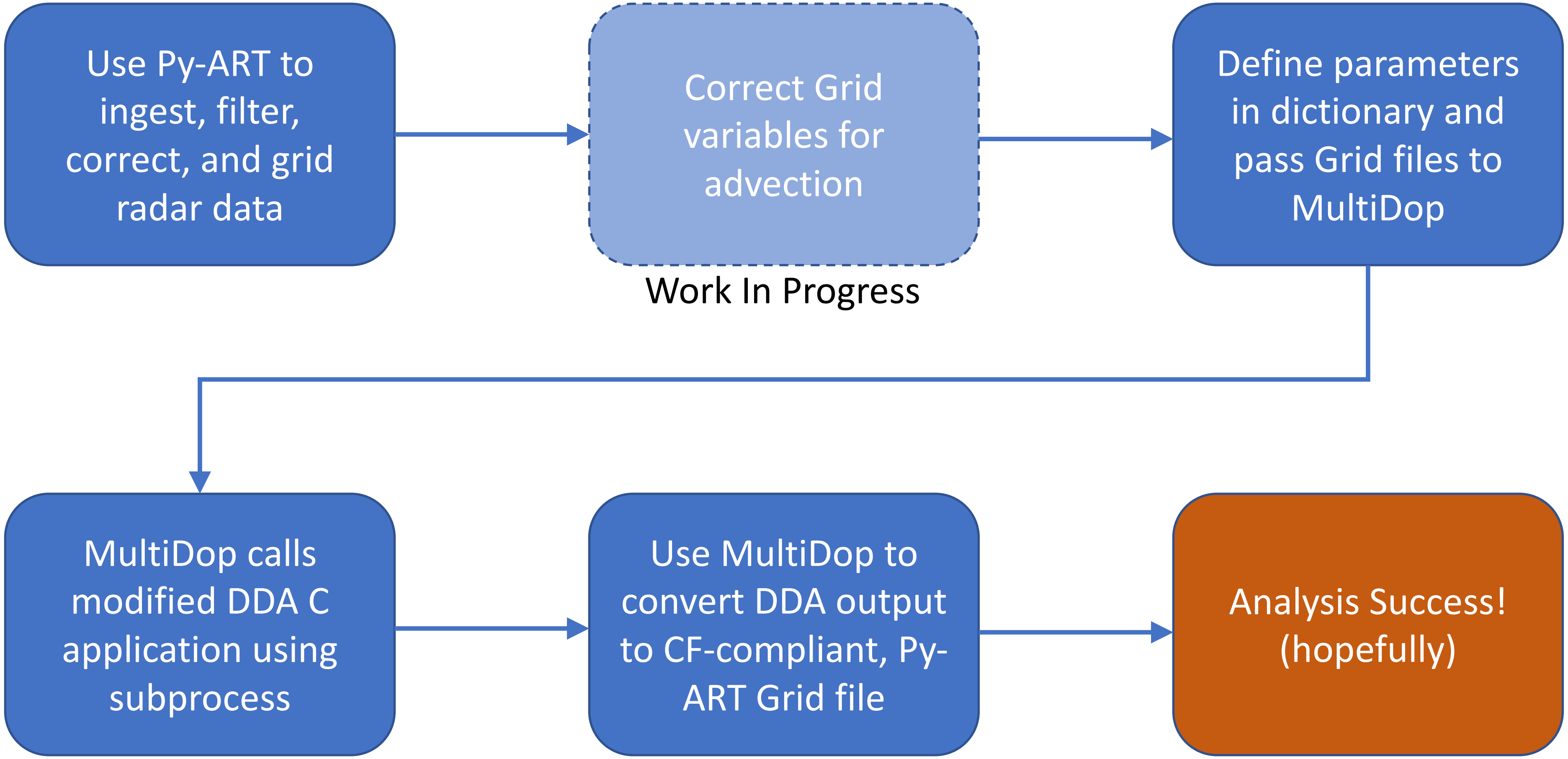
Total Cost Function $J = \underset{\text{Obs}}{J_O} + \underset{\text{Mass Cont.}}{J_M} + \underset{\text{Vort.}}{J_V} + \underset{\text{Smooth}}{J_S},$

The Culmination of the Dream - MultiDop

- Developed at NASA Marshall Space Flight Center
- Python wrapper for DDA C-based application
- Python classes to bridge Py-ART and DDA
- DDA updated to accept Py-ART grid format
- Python install script for compiling both C and Python components

How Does It All Work, Then?

- MultiDop makes Py-ART and DDA work together
- A sample workflow is available as a Jupyter notebook



Use Py-ART to ingest, filter, correct, and grid radar data

Correct Grid variables for advection

Work In Progress

Define parameters in dictionary and pass Grid files to MultiDop

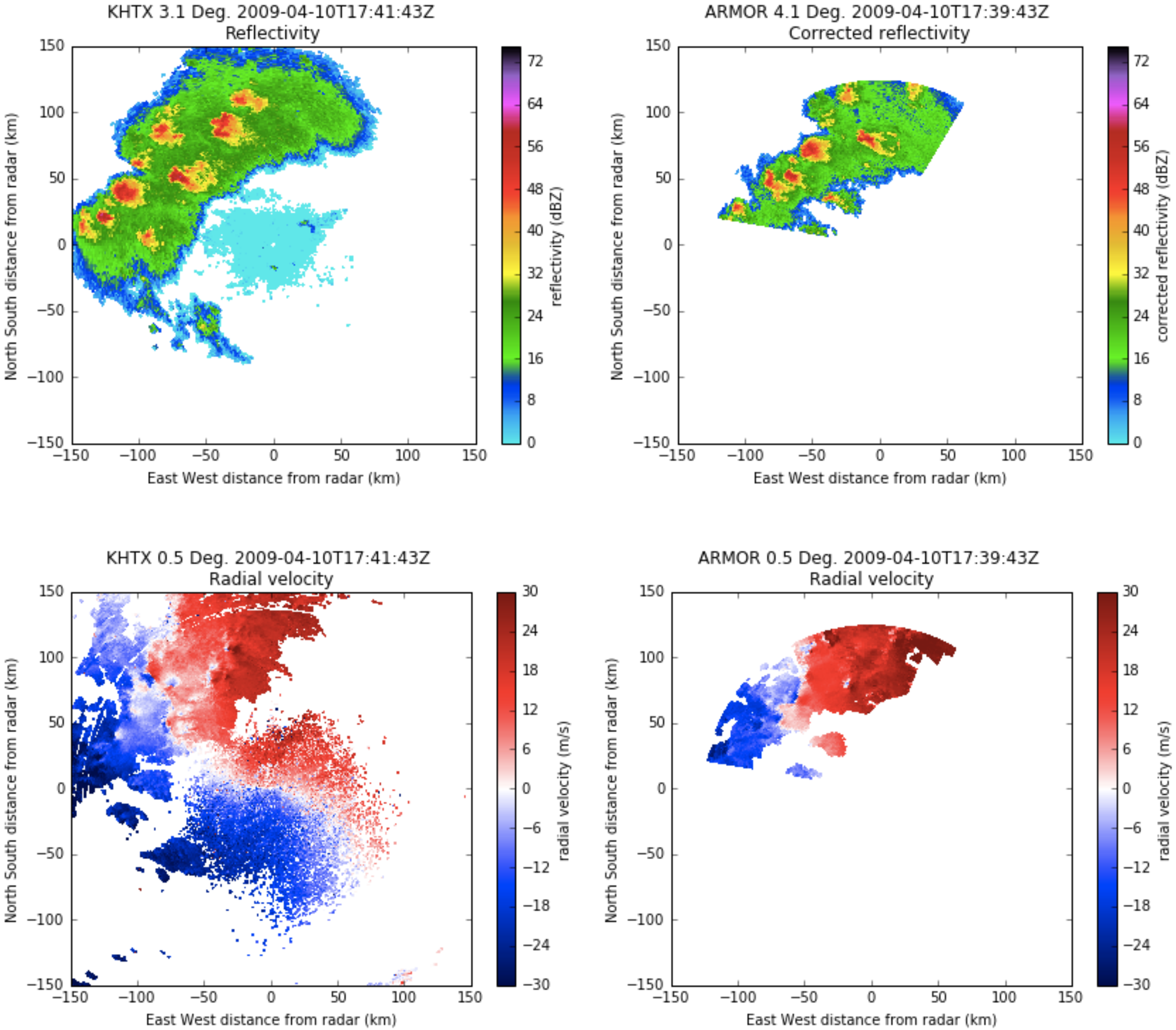
MultiDop calls modified DDA C application using subprocess

Use MultiDop to convert DDA output to CF-compliant, Py-ART Grid file

Analysis Success! (hopefully)

Py-ART Steps

- Use Py-ART to create Radar objects from volume files
- Use GateFilter or other editing on data if necessary
- Dealias velocity data if necessary
- Rename fields to match names among all volumes to be synthesized
- Create Py-ART Grid objects



Py-ART Advection Correction

- For radars that are non-synchronized or we need to determine and correct for advection of radial velocity patterns.
- We have implemented a image shift detection technique to get X/Y advection between volumes using cross correlation (same as in image stabilization).
- We have implemented an image shifter using NDImage
- To Do: Combine forward and backward projected images, “Advective averaging”

$$R(t+\Delta t, z, y, x) = \left(1 - \frac{t+\Delta t - t_1}{t_2 - t_1}\right) R_{t1}(t_1, z, y + v\Delta t, x + u\Delta t) + \frac{t+\Delta t - t_1}{t_2 - t_1} R_{t2}(t_2, z, y - v\Delta t, x - u\Delta t)$$

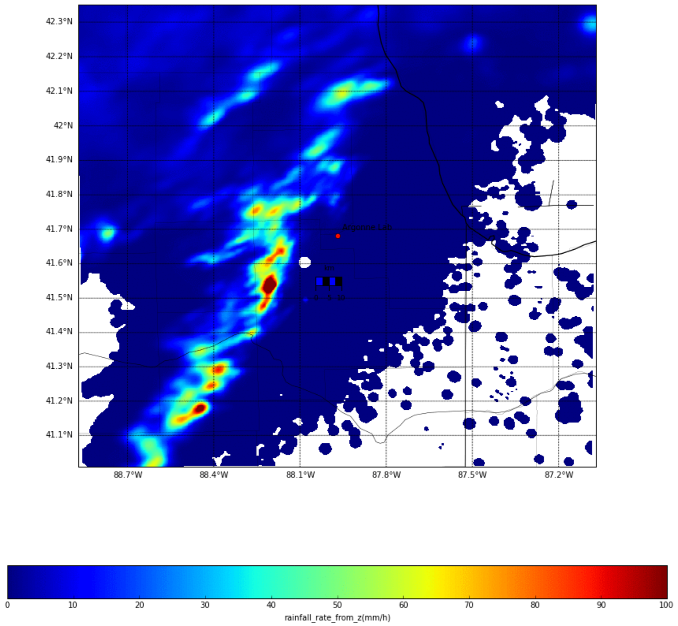
$$\mathbf{G}_{t1} = \mathcal{F}\left\{R_{t1}\right\}, \mathbf{G}_{t2} = \mathcal{F}\left\{R_{t2}\right\}$$

$$C = \frac{\mathbf{G}_{t1} \circ \mathbf{G}^*_{t2}}{\left|\mathbf{G}_{t1} \circ \mathbf{G}^*_{t2}\right|}$$

$$r = \mathcal{F}^{-1}\left\{C\right\}$$

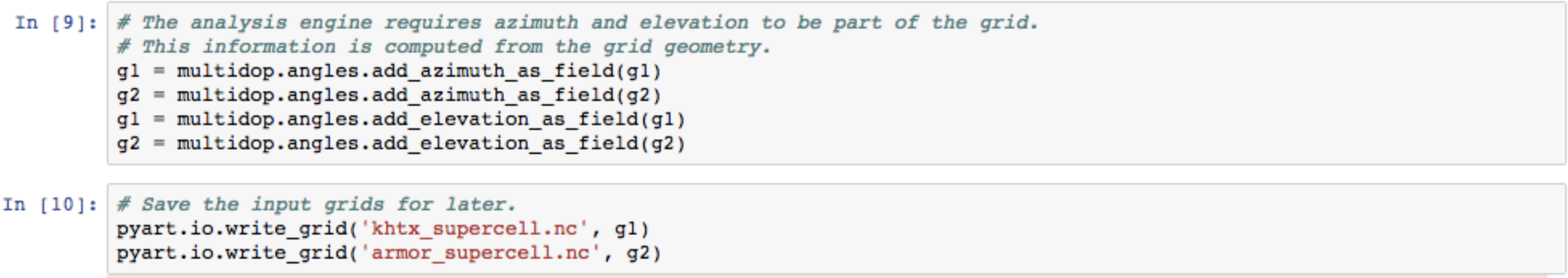
$$\Delta x, \Delta y = \operatorname{argmax}\left\{r\right\}$$

where F is the Fourier transform, $*$ is the complex conjugate and $\circ$ represents element wise multiplication.



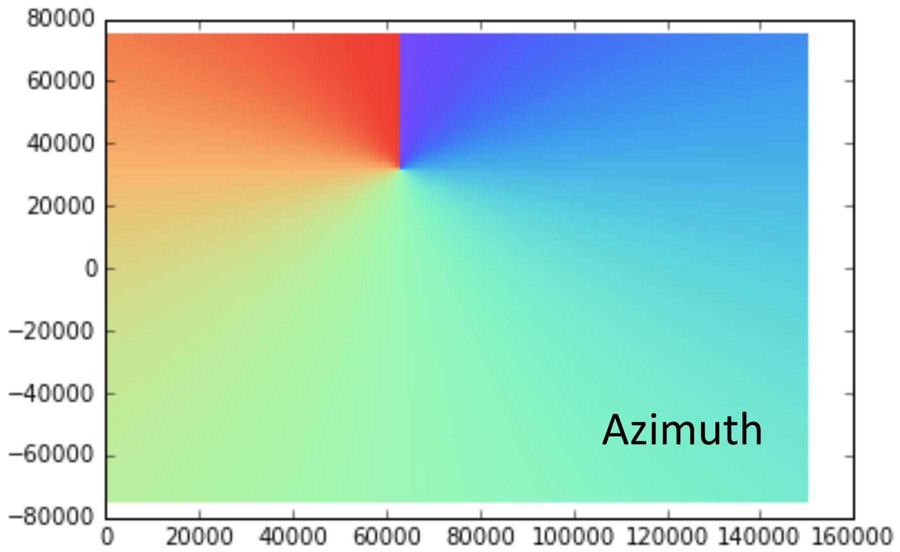
Other Special Py-ART Steps

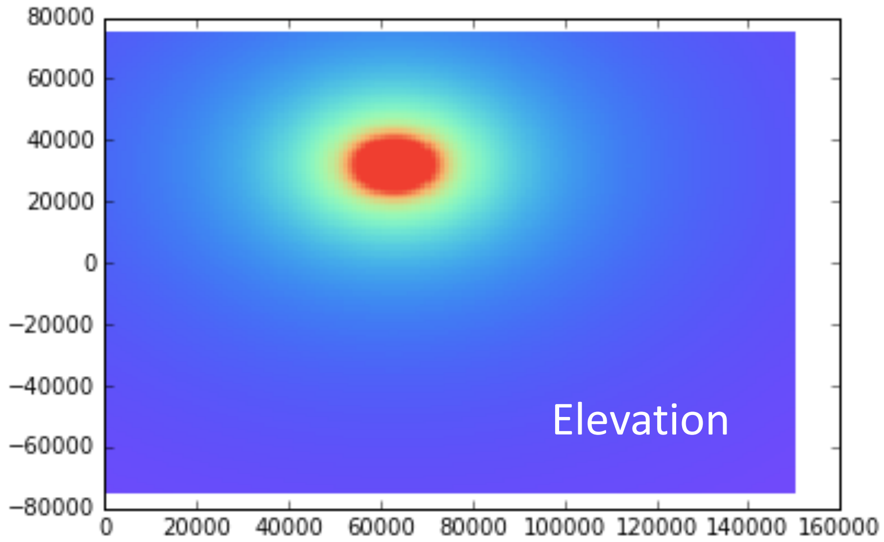
- Azimuth and Elevation must be added as fields to the Py-ART Grid objects
- MultiDop has functions to do this automatically using Grid attributes



```
In [9]: # The analysis engine requires azimuth and elevation to be part of the grid.
        # This information is computed from the grid geometry.
        g1 = multidop.angles.add_azimuth_as_field(g1)
        g2 = multidop.angles.add_azimuth_as_field(g2)
        g1 = multidop.angles.add_elevation_as_field(g1)
        g2 = multidop.angles.add_elevation_as_field(g2)

In [10]: # Save the input grids for later.
         pyart.io.write_grid('khtx_supercell.nc', g1)
         pyart.io.write_grid('armor_supercell.nc', g2)
```





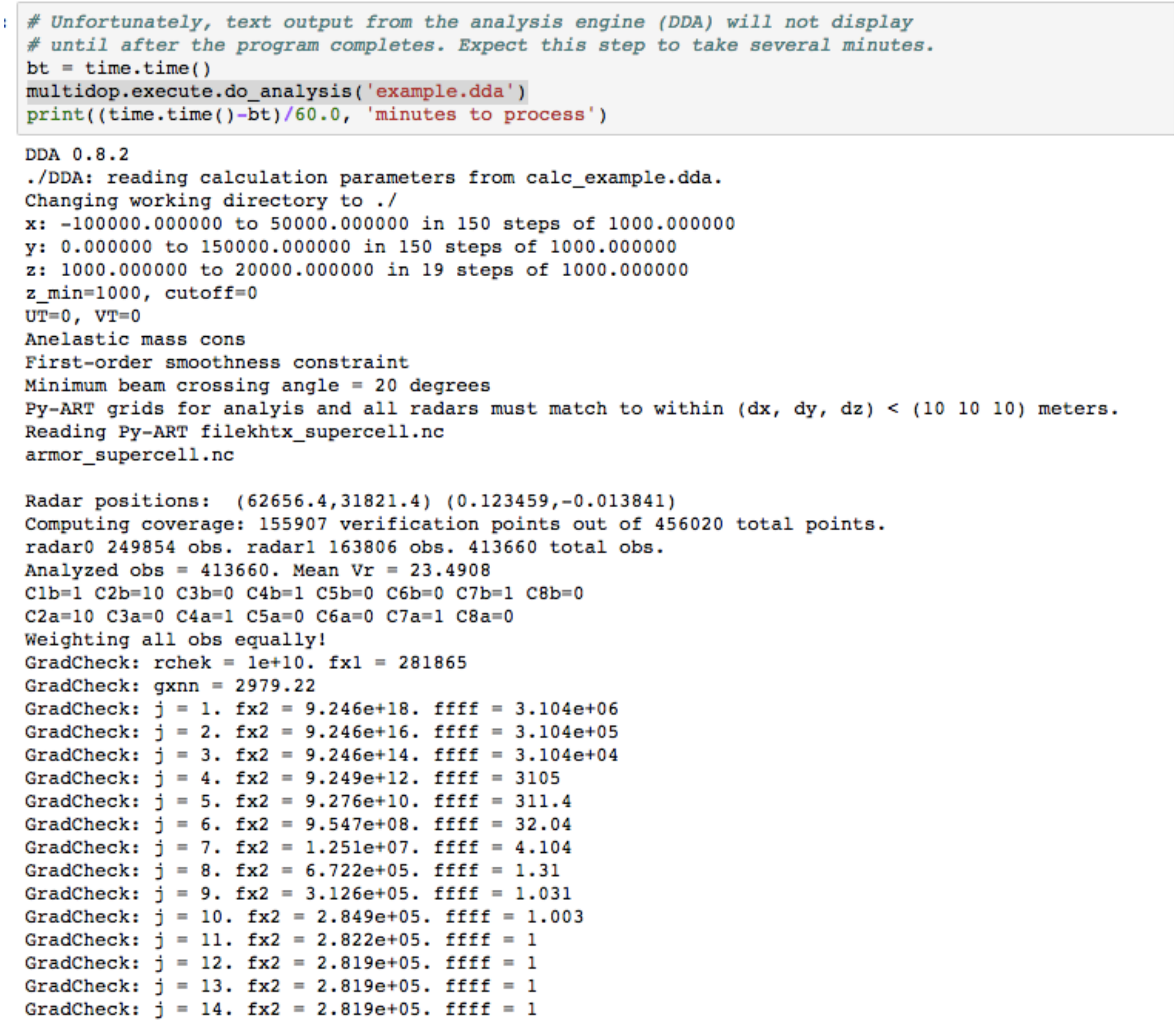
Define Parameters Step

- Tunable and user-defined parameters are handled via a dictionary
- ParamFile and CalcParamFile objects use this dictionary to create input scripts used by the DDA application
- Default values are used to fill in what end user does not provide

```
localfile = tempfile.NamedTemporaryFile()
pd = {'dir': './',
      'x': [-100000.0, 1000.0, 151],
      'y': [0.0, 1000.0, 151],
      'z': [1000.0, 1000.0,  20],
      'grid': [gl.origin_longitude['data'][0], gl.origin_latitude['data'][0], 0.0],
      'files': ['khtx_supercell.nc',
                'armor_supercell.nc'],
      'radar_names': ['KHTX', 'ARMOR'],
      'refl': 'DT',  # Name of reflectivity field. Must be common between radars.
      'vt': 'VT',  # Name of velocity field. Must be common between radars.
      'bgfile': None,
      'writeout': localfile.name,
      'min_cba': 20.0,  # Minimum beam-crossing angle
      'calc_params': 'calc_example.dda',
      'anel': 1,
      'laplace': 0,
      'read_dataweights': 2,
      'max_dist': 10.0,
      'cutoff': 0.0,
      'UT': 0.0,
      'VT': 0.0,
      'output_error': 0,
      'weak_height': -1,
      'upper_bc': 1,
      'itmax_frprmn': [200, 10],
      'itmax_dbrent': 200,
      'C1b': 1.0,  # Data weighting factor
      'C2b': 10.0,  # Mass continuity weighting factor
      'C3b': 0,  # Vorticity weighting factor
      'C4b': 1.0,  # Horizontal smoothing factor
      'C5b': 0.0,  # Vertical smoothing factor
      'C8b': 0.0,  # Sounding factor
      'vary_weights': 0,
      'filter': ['none', '', ''],
      'cvg_opt_bg': [1, 1, 1],
      'cvg_sub_bg': [0, 0, 0],
      'cvg_opt_fil': [0, 1, 1],
      'cvg_sub_fil': [0, 0, 0],
      'cvg_bg': [0, 0, 0],
      'cvg_fil': [0, 0, 0],
      'sseq_trip': [1.0, 1.0, 0.0]
     }
pf = multidop.parameters.ParamFile(pd, 'example.dda')
pf = multidop.parameters.CalcParamFile(pd, 'calc_example.dda')
```

DDA Application Step

- MultiDop calls the DDA C application via the subprocess module
- Text output from the application is captured, but not displayed until after application completes
- Entire process usually takes a few minutes
- Future work – Refactor DDA into a library so that it can be accessed via ctypes for improved Python integration



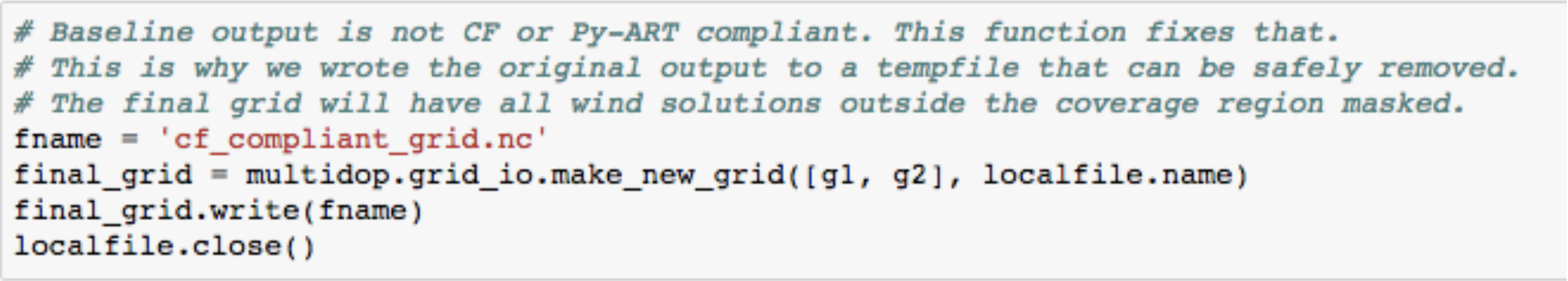
```python
# Unfortunately, text output from the analysis engine (DDA) will not display
# until after the program completes. Expect this step to take several minutes.
bt = time.time()
multidop.execute.do_analysis('example.dda')
print((time.time()-bt)/60.0, 'minutes to process')
```

```
DDA 0.8.2
./DDA: reading calculation parameters from calc_example.dda.
Changing working directory to ./
x: -100000.000000 to 50000.000000 in 150 steps of 1000.000000
y: 0.000000 to 150000.000000 in 150 steps of 1000.000000
z: 1000.000000 to 20000.000000 in 19 steps of 1000.000000
z_min=1000, cutoff=0
UT=0, VT=0
Anelastic mass cons
First-order smoothness constraint
Minimum beam crossing angle = 20 degrees
Py-ART grids for analyis and all radars must match to within (dx, dy, dz) < (10 10 10) meters.
Reading Py-ART filekhtx_supercell.nc
armor_supercell.nc

Radar positions:  (62656.4,31821.4) (0.123459,-0.013841)
Computing coverage: 155907 verification points out of 456020 total points.
radar0 249854 obs. radar1 163806 obs. 413660 total obs.
Analyzed obs = 413660. Mean Vr = 23.4908
C1b=1 C2b=10 C3b=0 C4b=1 C5b=0 C6b=0 C7b=1 C8b=0
C2a=10 C3a=0 C4a=1 C5a=0 C6a=0 C7a=1 C8a=0
Weighting all obs equally!
GradCheck: rchek = 1e+10. fx1 = 281865
GradCheck: gxnn = 2979.22
GradCheck: j = 1. fx2 = 9.246e+18. ffff = 3.104e+06
GradCheck: j = 2. fx2 = 9.246e+16. ffff = 3.104e+05
GradCheck: j = 3. fx2 = 9.246e+14. ffff = 3.104e+04
GradCheck: j = 4. fx2 = 9.249e+12. ffff = 3105
GradCheck: j = 5. fx2 = 9.276e+10. ffff = 311.4
GradCheck: j = 6. fx2 = 9.547e+08. ffff = 32.04
GradCheck: j = 7. fx2 = 1.251e+07. ffff = 4.104
GradCheck: j = 8. fx2 = 6.722e+05. ffff = 1.31
GradCheck: j = 9. fx2 = 3.126e+05. ffff = 1.031
GradCheck: j = 10. fx2 = 2.849e+05. ffff = 1.003
GradCheck: j = 11. fx2 = 2.822e+05. ffff = 1
GradCheck: j = 12. fx2 = 2.819e+05. ffff = 1
GradCheck: j = 13. fx2 = 2.819e+05. ffff = 1
GradCheck: j = 14. fx2 = 2.819e+05. ffff = 1
```

Last Step

- Nominal DDA output is not CF-compliant or Py-ART friendly
- MultiDop has function to convert output to Py-ART Grid file
- Option to mask winds outside Doppler lobes



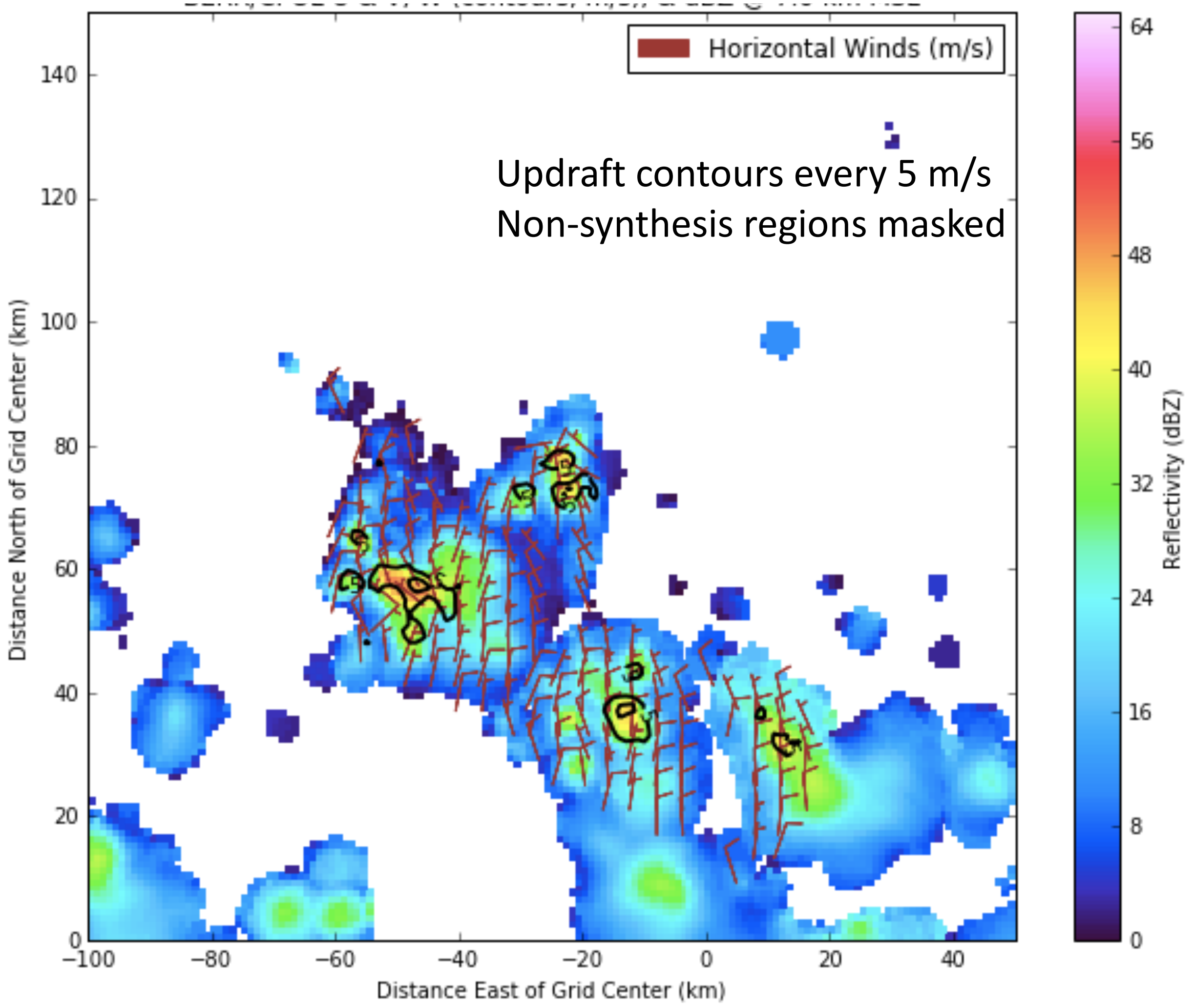
```
# Baseline output is not CF or Py-ART compliant. This function fixes that.
# This is why we wrote the original output to a tempfile that can be safely removed.
# The final grid will have all wind solutions outside the coverage region masked.
fname = 'cf_compliant_grid.nc'
final_grid = multidop.grid_io.make_new_grid([g1, g2], localfile.name)
final_grid.write(fname)
localfile.close()
```

MultiDop Checkout

North Alabama convection
- Supercell
- Multicell
- QLCS

<u>Lessons Learned</u>
- CEDRIC/MultiDop updraft locations and magnitudes qualitatively match
- MultiDop tunable parameters can greatly modify results
- Pay special attention to horizontal and vertical smoothing

ARMOR and KHTX Radars



Updraft contours every 5 m/s
Non-synthesis regions masked

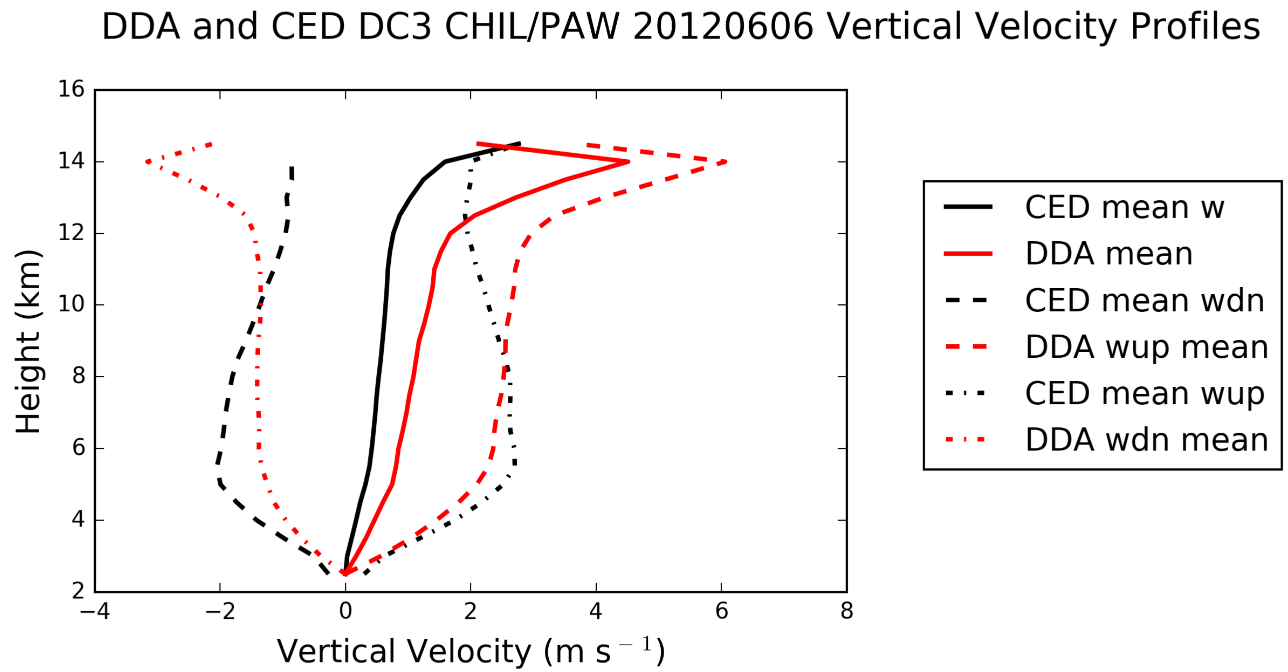
MultiDop Checkout (cont.)

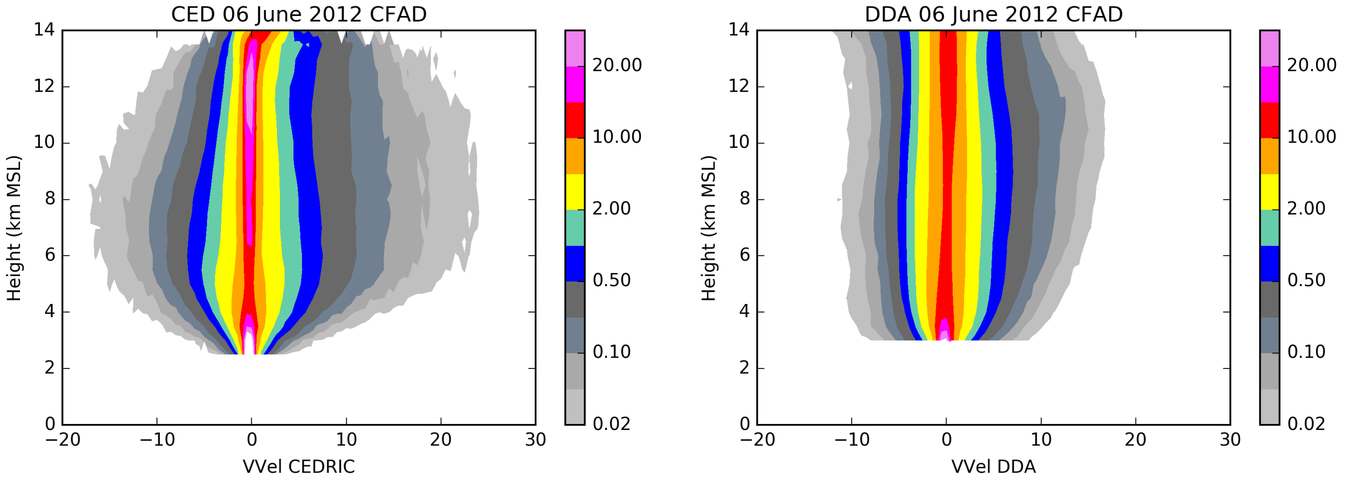
Northern Colorado DC3 Cases

- CSU-CHILL and CSU-Pawnee
- Volumes from 5 & 6 June 2012
- Multicellular convection

<u>Lessons Learned</u>

- MultiDop w/in ~1 m/s of CEDRIC
- Good spatial correspondence
- MultiDop ~10x slower than CEDRIC, but many times easier to use!
- Pay special attention to Py-ART gridding





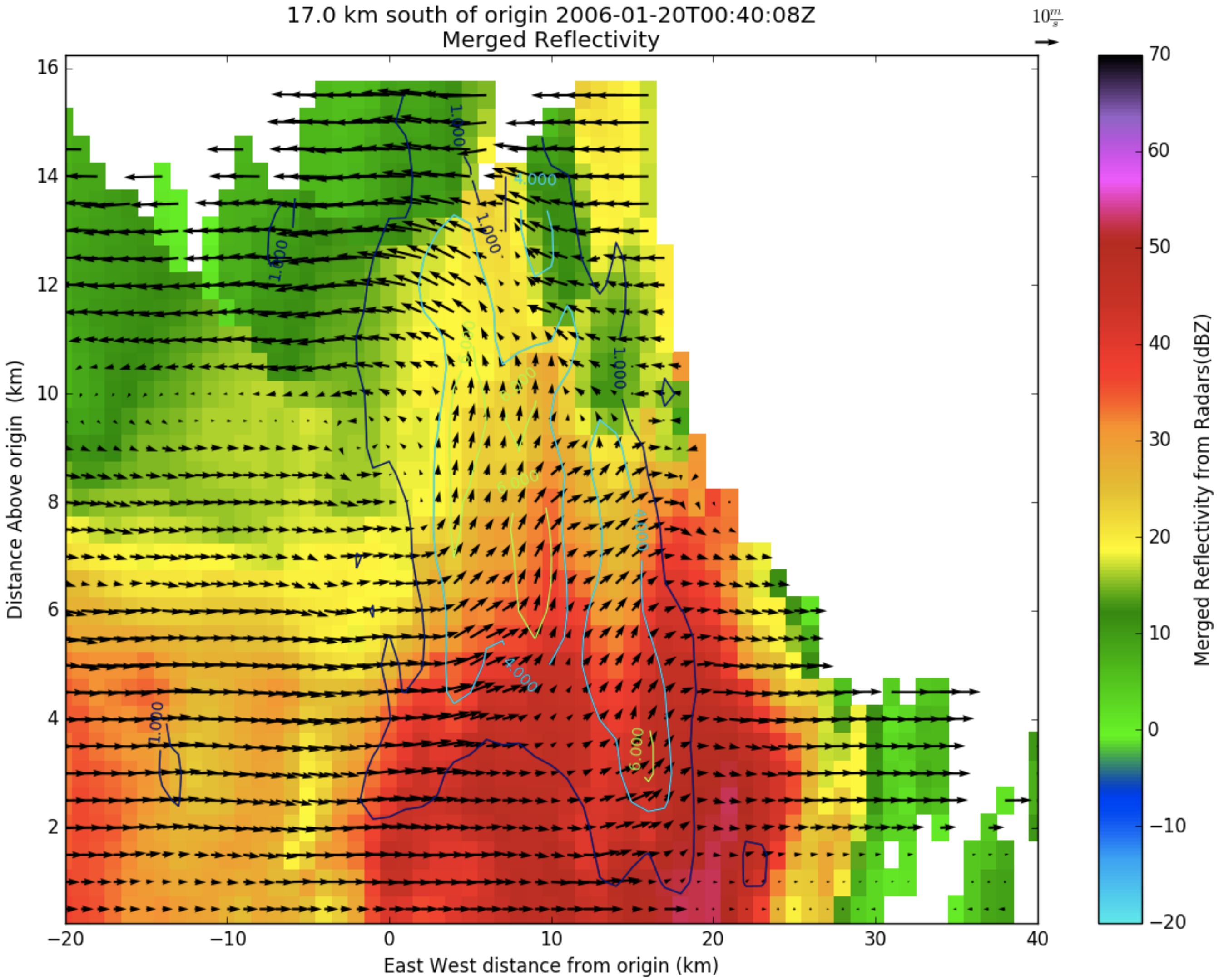
MultiDop Checkout (cont.)

Northern Australia Convection

- CPOL (Darwin) & Berrima S-band
- 19-22 January 2006, ~400 volumes
- Cluster: 1 instance MultiDop/core

<u>Lessons Learned</u>

- Needed strong mass continuity constraint (C2b = 1500) to suppress high-altitude noise in W
- Used Leise filter and strong horizontal smoothing to remove artifacts near edge of lobes
- Took advantage of 4/day soundings to help the retrieval



MultiDop will become open source in Spring 2017

If you use MultiDop, you **MUST** cite the following papers:

Shapiro, A., C. Potvin, and J. Gao, 2009: Use of a Vertical Vorticity Equation in Variational Dual-Doppler Wind Analysis. J. Atmos. Oceanic Technol., 26, 2089–2106, doi: 10.1175/2009JTECHA1256.1.

Potvin, C., A. Shapiro, and M. Xue, 2012: Impact of a Vertical Vorticity Constraint in Variational Dual-Doppler Wind Analysis: Tests with Real and Simulated Supercell Data. J. Atmos. Oceanic Technol., 29, 32–49, doi: 10.1175/JTECH-D-11-00019.1.

Contact timothy.j.lang@nasa.gov if you just can't wait!