The application of modern optimization packages in multisensor data assimilation

Robert Jackson[1], Scott Collis[1], Timothy Lang[2], Corey Potvin[3], Todd Munson[1]

1. Argonne National Laboratory, Argonne, IL, United States, 2. NASA Marshall Space Flight Center, Huntsville, AL, 3. Cooperateive Institute for Mesoscale Meteorological Studies/National Severe Storms Laboratory, Norman, OK

1. Motivation

Easy to use, efficient, and open source frameworks for assimilating data from multiple sensors currently do not exist.

PyDDA is an expandable framework that integrates data from weather radars and forecasting models using SciPy's optimization package to create meteorological fields.

2. Variational framework

Data from varying sensors can be assimilated using the 3D variational framework [1,2]. A cost function related to the difference between the retrieved field and that from a sensor *j* is minimized:

$$J(\mathbf{v}) = c_1 J_1(\mathbf{v}) + c_2 J_2(\mathbf{v}) + \ldots + c_n J_n(\mathbf{v})$$

Since an inital state of the meteorological field must be prescribed, this is commonly taken from model/rawinsonde data, or set to zero. In addition, a J_n can also represent a physical constraint, such as conservation of mass. The c_n represent the relative importance of each sensor or constraint to *J*.

3.

PyDDA (Pythonic Dual Doppler Analysis) is a Python package that implements the 3D variational framework built on the Python ARM Radar Toolkit (Py-ART) [3]. Currently features:

- do multiple doppler analysis of an arbitrary number of radars using mass continuity, vorticity equations
- use Weather Research and Forecasting (WRF) data as initial state
- use rawinsonde data as initial state
- visualize wind fields w/wind barbs + streamlines

The limited memory Broyden–Fletcher–Goldfarb–Shanno - Box optimization technique from SciPy used for optimization:

```
V = fmin_l_bfgs_b(J, winds, args=(...),
                  maxiter=10, pgtol=1e-3, bounds=bounds,
                  fprime=grad_J, disp=1, iprint=-1)
```

PyDDA available for public use and contribution here:

https://github.com/rcjackson/PyDDA

Example: Assimilating data from 2 radars + WRF model and visualize

Load two gridded radar files:

```
import pydda
import pyart

grid1 = pyart.io.read_grid(grid_name1)
grid2 = pyart.io.read_grid(grid_name2)
```

Then, prescribe the initial state from a WRF run:

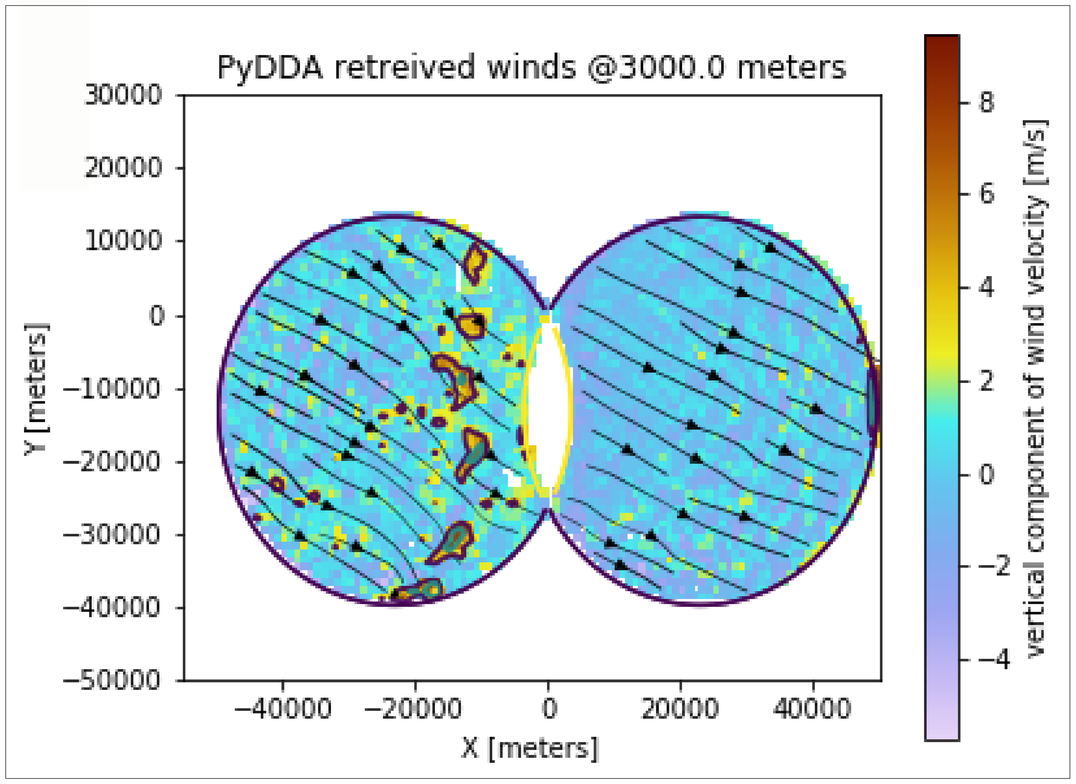
```
u1, v1, w1 = pydda.initialization.make_background_from_wrf(
    grid1, wrf_run_path, your_time, radar_loc, vel_field)
```

Let PyDDA retrieve the wind field:

```
Grids = pydda.retrieval.get_dd_wind_field(
    [grid1, grid2], u1, v1, w1, Co=100.0, Cm=1500.0)
```

Then, plot the result!

```
fig, ax = plt.subplots(1)
pydda.vis.plot_horiz_xsection_streamlines(
    Grids, ax, 'w', level=6, w_vel_contours=[3, 6, 9])
```



References

[1] Potvin, C.K., 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, https://doi.org/10.1175/JTECH-D-11-00019.1

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

[3] Helmus, J.J. and Collis, S.M., 2016: The Python ARM Radar Toolkit (Py-ART), a Library for Working with Weather Radar Data in the Python Programming Language. Journal of Open Research Software. 4(1), p.e25. DOI: http://doi.org/10.5334/jors.119

We are looking for more collaborators to add more capabilities! If interested contact Bobby Jackson at rjackson@anl.gov.

This poster has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. This research was supported by the Climate Model Development and Validation activity funded by the Office of Biological and Environmental Research in the US Department of Energy Office of Science. Computing resources were provided by the Laboratory Computing Resource Center of Argonne National Laboratory.