MetPy: Python Tools for Meteorological Data Analysis and Visualization

Ryan May
UCP/Unidata
10 June 2019
Unidata

• NSF-funded center for developing tools, data access, and community support

• Tools:
  • netCDF
  • LDM
  • IDV
  • THREDDS
  • Rosetta
  • AWIPS/GEMPAK
  • Python Training, MetPy, Siphon
What is MetPy?

- Toolkit for meteorological applications in Python
- Provide GEMPAK-like functionality in Python
Foundation

• Builds on many other Python tools:
  • NumPy
  • Matplotlib
  • SciPy
  • Pint
  • Xarray
  • Cartopy
• Test **everything**
• Automate as much as possible
Broad Goals

• Intended to be general purpose:
  • Education
  • Research
  • Other applications
• Be a community resource to find useful pieces
• Allow easily combining with other applications
Hurricane Florence

GOES-16 Visible, SST (contours), NDBC Buoy Observations, NHC Best and Forecast Track

Created: 2018-09-12T22:45:53Z
Hurricane Michael
Cross-Sections

NARR Cross-Section - (37.0, -105.0) to (35.5, -65.0) - Valid: 1987-04-04 18:00Z
Potential Temperature (K), Tangential/Normal Winds (knots), Relative Humidity (dimensionless)
Inset: Cross-Section Path and 500 hPa Geopotential Height
Isentropic Interpolation
Gridding

- Nearest Neighbor
- Barnes
- Cressman
- Linear
- Natural Neighbor
Interpolation & Derivatives

- Linear interpolation
- Log interpolation
- Derivative
- Gradient
- Laplacian
Kinematic Calculations

- Vorticity
- Divergence
- Deformation
- Frontogenesis
- Q-Vector
- Advection
Skew-T and Calculations

- Hodograph
- Sounding Calcs
  - CAPE/CIN
  - Storm Motion
  - LCL/LFC
  - Mixed Parcel
  - Supercell
  - Composite
More Features

- Xarray integration and decoding CF-convention metadata
- Reading NEXRAD and GINI file formats
- US counties and matching state borders for Cartopy
Replacing GEMPAK

source /Users/gempak/GEMPAK6.3.0/Gemenviron
SET CURDAY = `date -u +%Y%m%d`
set FRUN = 12

gdcntr <<EOF1
GDFILE = gfs/`{CURDAY}`/`{FRUN}`_gfs003.gem
GDATTIM = 'f012'
GLEVEL = 700
GVCORD = pres
CTYPE = f
GFUNC = avor(wnd)
CONTRUR = 2
CINT = 2
LINE = 1/1
TITLE = 31/-2/GFS -
GAREA = us
PROJ = 'str/90;-100;0'
DEVICE = 'gif|us.gif|1024;768'
EOF1

from datetime import datetime
import cartopy.crs as ccrs
import cartopy.feature as feature
import matplotlib.pyplot as plt
import metpy.calc as mpmcalc
from metpy.units import units
from netCDF4 import numdate
import numpy as np
import scipy.ndimage as ndimage
from aphis.nco.nco import NCO

now = NCO('https://www.nco.ncep.noaa.gov/thredds/grid/namrl/180604/18060416_1800_000.grib')
now = datetime.strptime(now['date'][0].strftime('%Y%m%d'), '%Y%m%d')

bft = now.query(time=[datetime(2016, 4, 16, 18)], accept='netcdf')
bft.variables['u-component_of_wind_isobaric'] = ndimage.gaussian_filter(bft.variables['u-component_of_wind_isobaric'], 10)

ds = bft.variables['time']
lats = ds.variables['lat'][:]
lons = ds.variables['lon'][:]
times = ds['time']
time = na_time = num2date(times[::], units='time.units')
lev_500 = np.where(ds.variables['isobaric'][:, 0] == 500, 0, 0)
winds_500 = bft.variables['u-component_of_wind_isobaric'][:, 0, lev_500, :, :]

dx, dy = np.gradient(winds_500

fig = plt.figure(figsize=(14., 12.))
plt.subplot(2, 1, 1)
plt.contourf(gf, elevation='sph', projection=ccrs.PlateCarree(), contornorm=norm, cmap='RdYlGn', levels=50)
plt.title('AVOSP-15.55 (fs=-19)', loc='left')
ax.set_extent([235., 290., 20., 58.], ccrs.PlateCarree())
ax.coastlines()
ax.add_features(feature.FRTF, linewidth=0.5)

fig2 = plt.subplots(2, 2, figsize=10, inline=1, inline_spacing=10, figsize=4)
plt.title('AVOSP-15.55 (fs=-19)', loc='left')
ax.set_extent([235., 290., 20., 58.], ccrs.PlateCarree())
ax.coastlines()
ax.add_features(feature.FRTF, linewidth=0.5)

clevs = [10000, 9000, 8000, 7000, 6000, 5000, 4000, 3000, 2000, 1000, 850, 700, 600, 500, 400, 300, 200, 100, 50, 30, 20, 10, 5, 3, 2, 1, 0.5, 0.25, 0.125, 0.0625]
cs = ax.contourf(lon, lat, winds_500, colors='grey', levels=clevs, linewidths=0.5, transform=ccrs.PlateCarree())
plt.title('AVOSP-15.55 (fs=-19)', loc='left')
ax.set_extent([235., 290., 20., 58.], ccrs.PlateCarree())
ax.coastlines()
ax.add_features(feature.FRTF, linewidth=0.5)

clevs = [10000, 9000, 8000, 7000, 6000, 5000, 4000, 3000, 2000, 1000, 850, 700, 600, 500, 400, 300, 200, 100, 50, 30, 20, 10, 5, 3, 2, 1, 0.5, 0.25, 0.125, 0.0625]
cs = ax.contourf(lon, lat, winds_500, colors='grey', levels=clevs, linewidths=0.5, transform=ccrs.PlateCarree())
plt.title('AVOSP-15.55 (fs=-19)', loc='left')
ax.set_extent([235., 290., 20., 58.], ccrs.PlateCarree())
ax.coastlines()
ax.add_features(feature.FRTF, linewidth=0.5)
MetPy Simplified Plotting

```python
grb_cat = TDSCatalog('http://thredds.ucar.edu/thredds/
  [catalog/satellite/ Goes16/GRB16/ABI/
  'FullDisk/Channel11/current/catalog.xml'])
grb_dat = grb_cat.datasets[0].remote_access(use_xarray=True)

img = ImagePlot()
img.data = grb_dat
img.field = 'Rad'
img.colormap = 'Greys'

m = MapPanel()
m.projection = 'data'
m.plots = [img]

c = PanelContainer()
c.size = (10, 10)
c.panels = [m]
c.draw()
```
Design Philosophy

- Fit well with scientific Python ecosystem
- Simple to use with your own data
- Unit-correctness built-in (using pint)
- Good online documentation
virtual_temperature

`metpy.calc.virtual_temperature(temperature, mixing, molecular_weight_ratio=\<Quantity(0.6219800858985514, 'dimensionless')\>)` [source]

Calculate virtual temperature.

This calculation must be given an air parcel's temperature and mixing ratio. The implementation uses the formula outlined in [Hobbs2006] pg.80.

Parameters:

- `temperature (pint.Quantity)` – The temperature
- `mixing (pint.Quantity)` – dimensionless mass mixing ratio
- `molecular_weight_ratio (pint.Quantity or float, optional)` – The ratio of the molecular weight of the constituent gas to that assumed for air. Defaults to the ratio for water vapor to dry air. ($\epsilon \approx 0.622$).

Returns: `pint.Quantity` – The corresponding virtual temperature of the parcel

Notes

\[ T_v = T \frac{w + \epsilon}{\epsilon (1 + w)} \]
Community-Driven Development

• Actively soliciting community contributions and involvement
• Open development on GitHub
• Roadmap available in the web documentation
  • We welcome input on what to add!
How can you contribute?

• Write code, docs, or an example
  • See our “Good First Issues” for inspiration
• Open a Pull Request
Contributing (cont.)

- Tests run automatically
- Get feedback

Welcome, and thanks for the contribution.

Overall this looks great and gets us closer to the 0.7.0 milestone, so thanks for the help. Just a few minor things to fix up and this will be ready.
Contributing (cont.)

• It gets merged!

**dopplershift** approved these changes on Nov 21, 2017

> Looks good to me, pending tests all passing. We can safely ignore Codacy here.

**dopplershift** commented on Nov 21, 2017

> 🎉 Congratulations on your first contribution to MetPy! Hopefully this is the first of many!
Everything Open

- All contributions go through this process
- Feel free to ask questions or comment on issues
- Or chat on Gitter
- Also can ask questions using the "metpy" tag on StackOverflow
Coming Attractions

- MetPy 1.0 release anticipated Fall 2019
- DROPPING Python 2.7 support
- Expand simplified plotting interface
- More calculations (e.g. dynamic tropopause)
- More file format support (e.g. METAR, BUFR)
- See roadmap for more information
Resources

• GitHub: https://github.com/Unidata/MetPy
• Documentation: https://unidata.github.io/MetPy
• Follow @metpy on Twitter