PyAPS Documentation

Release 1.0

Romain Jolivet, Piyush Agram, Cong Liu

June 13, 2012

CONTENTS

PyAPS module to compute InSAR phase delay maps from weather models.

Written by Romain Jolivet <rjolivet@gps.caltech.edu> and Piyush Agram <piyush@gps.caltech.edu>. The original Fortran package was written by Romain Jolivet and the Python version including support for different models was written by Piyush Agram.

Note: Details of the python module can be obtained here.

CHAPTER

ONE

PRE-REQUISITES

PyAPS assumes that the following packages are properly installed. Most of the packages can be obtained through ‘’macports” or ‘’apt-get” like utilities.

- Python with numpy and scipy (atleast 0.9).
- the grib_api library
- pygrib
- the hdf4 library
- pyhdf

CHAPTER

TWO

DOWNLOADING DATA

We provide an automatic script to get ERA-Interim data from the ECMWF website and from the UCAR website, the NARR data from the NOAA website and the MERRA data from the NASA's website. This is driven by the `autoget.py` file. However, users need to specify their passwords and login into the `model.cfg` file. We provide an example called `model.cfg.template`. Login and password can be obtain at:

- `http://data-portal.ecmwf.int/` for ECMWF
- `http://rda.ucar.edu/` for UCAR.

CHAPTER

THREE

PYAPS_GEO

class PyAPS.**PyAPS_geo**(*gribfile*, *demfile*, *grib='ECMWF'*, *humidity='Q'*, *demtype=<type 'numpy.int16'>*, *demfmt='HGT'*)

Class for dealing with Atmospheric phase corrections in geo-coded space. Operates on one weather model file and one Geo.rsc file at a time.

This use the relative humidity information from the specified ERA file over the area specified by the dem rsc file.

Default Usage:

```
myaps = PyAPS.PyAPS_geo('ecmwfile','demfile')
phs = np.zeros((myaps.ny,myaps.nx))
myaps.getdelay(phs,inc=23.,wvl=0.05656)
```

(Default) This uses the specific humidity information from the specified ECMWF file (using getecmwf.py) over the area specified by the dem rsc file. The incidence angle and wavelength is passed in as keywork inputs. The array (phs) could also have been a h5py object of the correct size.

Detailed Example:

```
myaps = PyAPS.PyAPS_geo('erafile','demfile',grib='ERA',humidity='R',demtype=np.float32,demfmt='R
myaps.getdelay('outfile',inc=23.,wvl=0.05656)
```

This use the relative humidity information from the specified ERA file over the area specified by the dem rsc file. The DEM file in this case is also in float32 RMG format. The output is directly written to file.

__init__(*gribfile*, *demfile*, *grib='ECMWF'*, *humidity='Q'*, *demtype=<type 'numpy.int16'>*, *demfmt='HGT'*)

Initiates the data structure for atmos corrections in geocoded domain.

Args:

- gribfile (str) : Path to the weather model file.
- demfile (str) : Path to the SIM_nRLKS.hgt file.

Kwargs:

- grib (str) : Can be ECMWF, ERA, NARR or MERRA depending on the model.
- humidity (str) : Can be 'Q' or 'R' depdending on specific or relative humidity.
- demtype (dtype) : Data type of the DEM File.
- demfmt (str) : Can be 'HGT' or 'RMG' depending on type of file. If RMG, altitude is assumed to be second channel.

Note: The DEMFile is only used to set up the geographic boundaries of the problem and the size of the output matrix during init. The GRIBfile is read and the 3D interpolation function is prepared.

getdelay (*dataobj*, *inc=0.0*, *wvl=12.566370614359172*)
Write delay to a matrix / HDF5 object or a file directly.

Args:

- dataobj (str or HDF5 or np.array): Final output. If str, output is written to file.

Kwargs:

- inc (np.float): Incidence angle in degrees. Default is vertical.
- wvl (np.float): Wavelength in meters. Default output results in delay in meters.

Note: If dataobj is string, output is written to the file. If np.array or HDF5 object, it should be of size (ny,nx).

CHAPTER

FOUR

PYAPS_RDR

class PyAPS.**PyAPS_rdr**(*gribfile*, *demfile*, *grib='ECMWF'*, *humidity='Q'*, *demtype=<type 'numpy.float32'>*, *demfmt='RMG'*)

Class for dealing with Atmospheric phase corrections in radar coordinates. Operates on one weather model file and one Geo.rsc file at a time.

Default Usage:

```
myaps = PyAPS.PyAPS_rdr('ecmwfile','simfile')
phs = np.zeros((myaps.ny,myaps.nx))
myaps.getdelay(phs,inc=23.,wvl=0.05656)
```

(Default) This uses the specific humidity information from the specified ECMWF file (using getecmwf.py) over the area specified by the sim rsc file. The incidence angle and wavelength is passed in as keywork inputs. The array (phs) could also have been a h5py object of the correct size. The simfile by default is assumed to be a float32 RMG file (ROI-PAC).

Detailed Example:

```
myaps = PyAPS.PyAPS_rdr('erafile','simfile',grib='ERA',humidity='R',demtype=np.float32,demfmt='H
myaps.getdelay('outfile',inc=23.,wvl=0.05656)
```

This use the relative humidity information from the specified ERA file over the area specified by the sim rsc file. The simulation file in this case is also in float32 binary format. The output is directly written to file.

Mapping Example:

```
myaps = PyAPS.PyAPS_rdr('erafile','simfile',grib='ERA')
myaps.getgeodelay('outfile',lat='latfile',lon='lonfile',inc='incfile',wvl=0.05656)
```

This use the relative humidity information from the specified ERA file over the area specified by the sim rsc file. The simulation file in this case is also in float32 RMG format (ROI-PAC). The output is directly written to file. latfile,lonfile and incfile are all the same size as simfile. These are products typically generated for processing InSAR stacks.

__init__(*gribfile*, *demfile*, *grib='ECMWF'*, *humidity='Q'*, *demtype=<type 'numpy.float32'>*, *demfmt='RMG'*)

Initiates the data structure for atmos corrections in geocoded domain.

getdelay(*dataobj*, *inc=0.0*, *wvl=12.566370614359172*)

Write delay to a matrix / HDF5 object or a file directly.

Args:

- dataobj (str or HDF5 or np.array): Final output. If str, output is written to file.

Kwargs:

- inc (np.float): Incidence angle in degrees. Default is vertical.
- wvl (np.float): Wavelength in meters. Default output results in delay in meters.

Note: If dataobj is string, output is written to the file. If np.array or HDF5 object, it should be of size (ny,nx).

---

getgeodelay(*dataobj*, *lat=None*, *lon=None*, *inc=None*, *wvl=12.566370614359172*)

Write delay to a matrix / HDF5 object or a file directly. This is used when the latitude and longitude values are available for each radar pixel. Incidence angle can be a constant or a file name.

Args:

- dataobj (str or HDF5 or np.array): Final output. If str, output is written to file.

Kwargs:

- lat (str) : Path to the latitude file (np.float32).
- lon (str) : Path to the longitude file (np.float32).
- inc (str or np.float): Path to incidence angle file in degrees (str) or a constant float.
- wvl (np.float) : Wavelength in meters. Default output results in delay in meters.

Note: If dataobj is string, output is written to the file. If np.array or HDF5 object, it should be of size (ny,nx).

CHAPTER

FIVE

PROCESSOR

processor.**initconst**()

Initialization of various constants needed for computing delay.

Args:

- None

Returns:

- constdict (dict): Dictionary of constants

processor.**intP2H**(*lvls*, *hgt*, *gph*, *tmp*, *vpr*, *cdic*)

Interpolates the pressure level data to altitude.

Args:

- lvls (np.array) : Pressure levels.
- hgt (np.array) : Height values for interpolation.
- gph (np.array) : Geopotential height.
- tmp (np.array) : Temperature.
- vpr (np.array) : Vapor pressure.
- cdic (dict) : Dictionary of constants.

Note: gph,tmp,vpr are of size (nstn,nlvls).

Returns:

- Presi (np.array) : Interpolated pressure.
- Tempi (np.array) : Interpolated temperature.
- Vpri (np.array) : Interpolated vapor pressure.

Note: Cubic splines are used to convert pressure level data to height level data.

---

processor.**PTV2del**(*Presi*, *Tempi*, *Vpri*, *hgt*, *cdict*)

Computes the delay function given Pressure, Temperature and Vapor pressure.

Args:

- Presi (np.array) : Pressure at height levels.

- Tempi (np.array) : Temperature at height levels.
- Vpri (np.array) : Vapor pressure at height levels.
- hgt (np.array) : Height levels.
- cdict (np.array) : Dictionary of constants.

Returns:

- DDry2 (np.array) : Dry component of atmospheric delay.
- DWet2 (np.array) : Wet component of atmospheric delay.

Note: Computes refractive index at each altitude and integrates the delay using cumtrapz.

---

processor.**make3dintp**(*Delfn*, *lonlist*, *latlist*, *hgt*, *hgtscale*)

Returns a 3D interpolation function that can be used to interpolate using llh coordinates.

Args:

- Delfn (np.array) : Array of delay values.
- lonlist (np.array) : Array of station longitudes.
- latlist (np.array) : Array of station latitudes.
- hgt (np.array) : Array of height levels.
- hgtscale (np.float) : Height scale factor for interpolator.

Returns:

- fnc (function) : 3D interpolation function.

Note: We currently use the LinearNDInterpolator from scipy.

CHAPTER

SIX

WEATHER MODELS

6.1 ERA-Interim Reanalysis

era.**cc_era**(*tmp*, *cdic*)

Clausius Clayperon law used by ERA Interim.

Args:

- tmp (np.array) : Temperature.
- cdic (dict) : Dictionary of constants

Returns:

- esat (np.array) : Water vapor saturation partial pressure.

era.**get_era**(*fname*, *minlat*, *maxlat*, *minlon*, *maxlon*, *cdic*, *humidity='Q'*)

Read data from ERA interim grib file. Note that the Lon values should be between [0-360]. GRB file with weather model data can be downloaded from http://dss.ucar.edu/datasets/ds627.0/

Args:

- fname (str): Path to the grib file
- minlat (np.float): Minimum latitude
- maxlat (np.float): Maximum latitude
- minlon (np.float): Minimum longitude
- maxlon (np.float): Maximum longitude
- cdic (np.float): Dictionary of constants

Kwargs:

- humidity (str): Specific ('Q') or relative humidity ('R').

Returns:

- lvls (np.array): Pressure levels
- latlist(np.array): Latitudes of the stations
- lonlist(np.array): Longitudes of the stations
- gph (np.array): Geopotential height
- tmp (np.array): Temperature
- vpr (np.array): Vapor pressure

Note: Uses cc_era by default.

era.**get_ecmwf** (*fname*, *minlat*, *maxlat*, *minlon*, *maxlon*, *cdic*, *humidity='Q'*)
Read data from ERA Interim grib file. Note that Lon values should be between [0-360]. GRB file with weather model data can be downloaded from http://www.ecmwf.int using getecmwf.py.

Args:

- fname (str): Path to the grib file
- minlat (np.float): Minimum latitude
- maxlat (np.float): Maximum latitude
- minlon (np.float): Minimum longitude
- maxlon (np.float): Maximum longitude
- cdic (np.float): Dictionary of constants

Kwargs:

- humidity (str): Specific ('Q') or relative humidity ('R').

Returns:

- lvls (np.array): Pressure levels
- latlist(np.array): Latitudes of the stations
- lonlist(np.array): Longitudes of the stations
- gph (np.array): Geopotential height
- tmp (np.array): Temperature
- vpr (np.array): Vapor pressure

Note: Uses cc_era by default.

6.2 On Clausius-Clapeyron Laws for ERA-Interim

We propose here three different methods to estimate the water vapor partial pressure from the outputs given by ERA-Interim: (1) from relative humidity with a mixed Clausius-Clapeyron law (default), (2) from relative humidity with a simple Clausius-Clapeyron law (i.e. as in Jolivet et al. 2011.), and (3) from specific humidity. Choice can be made between (1) and (3) in the PyAPS_rdr and PyAPS_geo functions. (2) can be activated inside the library and is hardcoded (non-recommended). These laws are valid only for the ERA-Interim re-analysis, and therefore only concern the routines get_era and get_ecmwf (hereafter named get_(eralecmwf).

(1) The get_(eralecmwf) routine reads the relative humidity, $R_e(z)$ as a function of elevation z, in the GRIB file. Water vapor partial pressure, $P_{vpr}(z)$ as a function of elevation z, is given by,

$$P_{vpr}(z) = P^{\star}(z)\frac{R_e(z)}{100}, \tag{6.1}$$

where $P^\star(z)$, the saturation water vapor partial pressure, given by,

$$\begin{aligned} P^\star(z) &= P^\star_w(z) = a1e^{a3_w \frac{T(z)-T_o}{T(z)-a4_w}} && \text{, for } T(z) >= T_o \\ P^\star(z) &= P^\star_i(z) = a1e^{a3_i \frac{T(z)-T_o}{T(z)-a4_i}} && \text{, for } T(z) <= T_i \\ P^\star(z) &= P^\star_i(z) + (P^\star_w(z) - P^\star_i(z)) \frac{T(z)-T_i}{T_o-T_i}^2 && \text{, for } T_o < T(z) < T_i. \end{aligned}$$

$P^\star_w(z)$ gives the Clausius-Clapeyron law "over water", $P^\star_i(z)$ gives the Clausius-Clapeyron law "over ice". This way, over $T_o = 273.16$ K, $P^\star_w$ is used, and under $T_i = 250.16$, $P^\star_i$ is used while a weighted average of both is used in between. The parameter values are: $a1 = 611.21$ hPa, $a3_w = 17.502$, $a4_w = 32.19$ K, $a3_i = 22.587$ and $a4_i = -0.7$ K.

(2) The get_(eralecmwf) routine reads the relative humidity, $R_e(z)$ as a function of elevation z, in the GRIB file. Water vapor partial pressure, $P_{vpr}(z)$ as a function of elevation z, is given by,

$$P_{vpr}(z) = P^\star(z) \frac{R_e(z)}{100}, \tag{6.2}$$

where $P^\star(z)$, the saturation water vapor partial pressure, given by,

$$P^\star(z) = a1e^{\frac{2.5e6}{R_v}(\frac{1}{T_o} - \frac{1}{T(z)}}, \tag{6.3}$$

where, $R_v = 461.495$ J.Kg^{-1}.K^{-1} is the water vapor specific constant.

(3) The get_era routine reads the specific humidity, $S_h(z)$ as a function of elevation z. Water vapor partial pressure $P_{vpr}(z)$ is given by,

$$P_{vpr}(z) = S_h(z)P(z)\frac{\alpha}{1+(\alpha-1)S_h(z)} \text{ and } \alpha = \frac{R_v}{R_d}, \tag{6.4}$$

where, $P(z)$ is the pressure at elevation z, $R_v = 461.495$ J.Kg^{-1}.K^{-1} is the water vapor specific constant and $R_d = 287.05$ J.Kg^{-1}.K^{-1} is the dry air specific constant.

Although no strong discrepancies are shown between the three methods, variations are expected at low altitude. Methods (1) and (3) provide the exact same results. These laws are in the routines called *cc_⋆* and any supplementary model you would like to add should include it own cc routine.

6.3 Delay Package

`delay.`**`cc_eraorig`**(*tmp*, *cdic*)

This routine takes temperature profiles and returns Saturation water vapor partial pressure using the Clausius-Clapeyron law applied in Jolivet et al. 2011,GRL, doi:10.1029/2011GL048757. It can be used in case you are using Relative Humidity from ECMWF models

Args:

- tmp (np.array) : Temperature
- cdic (dict) : Dictionnary of constants

Returns:

- esat (np.array) : Saturation water vapor partial pressure.

`delay.`**`read_eratxt`** (*fname*, *cdic*)

Read ECMWF files from 0.75 degree grid similar to Romain Jolivet's Delay Package.

Args:

- fname (str): Path to the delay file
- cdic (np.float): Dictionary of constants

Kwargs:

- humidity (str): Specific ('Q') or relative humidity ('R').

Returns:

- lvls (np.array): Pressure levels
- latlist(np.array): Latitudes of the stations
- lonlist(np.array): Longitudes of the stations
- gph (np.array): Geopotential height
- tmp (np.array): Temperature
- vpr (np.array): Vapor pressure

Note: Uses cc_eraorig by default.

CHAPTER

SEVEN

ROI-PAC UTILITIES

utils.**geo_rsc**(*inname*, *full=False*)

Reading a ROI-PAC style geocoded rsc file.

Args:

- inname (str): Path to the RSC file.

Returns:

- lon (np.array) : Array of min and max lon values.
- lat (np.array) : Array of min and max lat values.
- nx (np.int) : Number of lon bins.
- ny (np.int) : Number of lat bins.

Note: Currently set up to work with dem.rsc file from ROI-PAC.

---

utils.**rd_rsc**(*inname*, *full=False*)

Reading a ROI-PAC style RSC file.

Args:

- inname (str): Path to the RSC file.

Returns:

- lat (np.array) : Array of lat of the 4 corners.
- lon (np.array) : Array of lon of the 4 corners.
- nx (np.int) : Number of range bins.
- ny (np.int) : Number of azimuth lines.
- dpix (np.float): Average pixel spacing.

Note: Currently set up to work with SIM_nrlks.hgt from ROI-PAC.

---

utils.**glob2rdr**(*nx*, *ny*, *lat*, *lon*, *latl*, *lonl*, *plotflag='n'*)

Transfert these latlon coordinates into radar geometry (xi,yi) with a simple linear transformation given the first pixel and the pixel spacing of the simulation.

Args:

- nx (np.int) : Number of range bins.

- ny (np.int) : Number of azimuth lines.
- lat (np.array) : Array of latitudes of the corners
- lon (np.array) : Array of longitudes of the corners
- latl (np.array) : Latitudes of the stations.
- lonl (np.array) : Longitudes of the stations.

Kwargs:

- plotflag (bool) : Plot the stations distribution.

Returns:

- xi (np.array) : Position of stations in range.
- yi (np.array) : Position of stations in azimuth.

Note: Mapping function is : * range = a1*lat+b1*lon+c1 * azimu = a2*lat+b2*lon+c2 * First point is (1,1) i.e. Near Range, First Lane <==> Lat[1],Lon[1] * Second point is (nx,1) i.e. Far Range, First Lane <==> Lat[2],Lon[2] * Third point is (1,ny) i.e. Near Range, Last Lane <==> Lat[3],Lon[3] * Fourth point is (nx,ny) i.e. Far Range, Last Lane <==> Lat[4],Lon[4]

CHAPTER

EIGHT

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

INDEX