epygram

Enhanced PYthon for GRaphics and Analysis of Meteorological fields

Alexandre Mary [1], *Sébastien Riette* [2]

1. Météo France : CNRM/GMAP/COOPE
2. Météo France : CNRM/GMME/MESO-NH

Aladin/Hirlam joint 26th Workshop All-Staff Meeting 2016, Lisbon

Outline

Handling NWP/Climate models output

- closest to the model (*historical* files, model variable & geometry) and/or post-processed output
- proxy for data processing (plots, extractions, time series...)

Handling NWP/Climate models output

- closest to the model (*historical* files, model variable & geometry) and/or post-processed output
- proxy for data processing (plots, extractions, time series...)

A variety of data formats

- LFI/FA : historical, proper to our models
- GRIB 1&2 : WMO norm, MF Forecast dept., ECMWF
- others : netCDF (scientifical community), LFA, GeoPoints(ASCII)...

Handling NWP/Climate models output

- closest to the model (*historical* files, model variable & geometry) and/or post-processed output
- proxy for data processing (plots, extractions, time series...)

A variety of data formats

- LFI/FA : historical, proper to our models
- GRIB 1&2 : WMO norm, MF Forecast dept., ECMWF
- others : netCDF (scientifical community), LFA, GeoPoints(ASCII)...

Meta-data handling

- geometry : 3D localization of gridpoints
- time validity, incl. start/term and cumulative processes
- nature of data

⇒ **Goal :** gather these aspects into *one* library, with eased file/data, file/metadata, data/metadata interactions.
The whole in a language that enables rich downstream applications...

⇒ **Goal :** gather these aspects into *one* library, with eased file/data, file/metadata, data/metadata interactions.
The whole in a language that enables rich downstream applications...

Context in research dept. at MF

- many different tools, uses and tips, from one person to another
- large heterogeneity in languages, graphical tools and so on...
- redundancy, cleanness of code, modularity ? and *maintenance*...

→ by the end of 2013, will to converge to a common tool/library

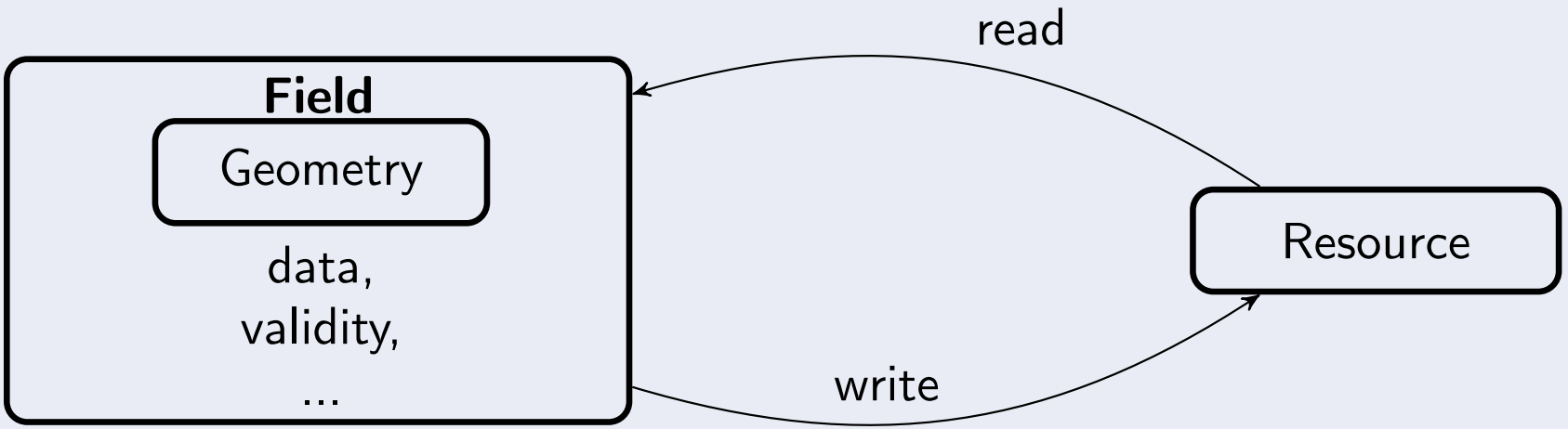
⇒ **epygram**

Python : not only a fashion victim ?

- hundreds of applicative libraries avalaible (graphical, scientific, system, interfacing with Fortran/C, IO & data formats...)
- object-oriented : mandatory for file/data/metadata interactions
- free, with an expanding community
- interactive/script duality
- interoperability with the *Vortex* at Meteo France (scripting system for operational & research model-running)
- *simplicity (...)*

3 main concepts

- **Resource** : container of *fields* encoded in a *format*, possibly ± compressed. Generaly, a *file*.
- **Field** : collection of values along a given *geometry*, with a given temporal validity. Superstar field = horizontal 2D field. (but also : vertical profile or section, transect, 3D field, local point(s)...)
- **Geometry** : collection of spatial positions in 3D (a "grid") on which is colocalized a *field*



(In-)dependances

Each object is self-sufficient :

- Fields and Resources are *independant* $\neq$ *incoherent* : no need of each other to fully exist, but ability to interact with each other !
- A field has a geometry, that can exist without reference to a field !
- *What about* field nature identification ? (each *Resource* format having often its own physical parameter designation...).

one exemple

ex : U-component of wind at (hybrid-pression) level 58

- FA : `S058WIND.U.PHYS`
- GRIB (GRIB_API syntax) :
  - GRIB1 :
    ```
    indicatorOfParameter=33, # U-wind
    indicatorOfTypeOfLevel=109, # hybrid-pressure levels
    level=58, # level 58
    table2Version=1, # necessary for local tables
    ```
  - GRIB2 :
    ```
    discipline=0, # meteorological products
    parameterCategory=3, # momentum
    parameterNumber=2, # U-component of wind
    typeofFirstFixedSurface=119, # hybrid-pressure levels
    level=58, # level 58
    ```

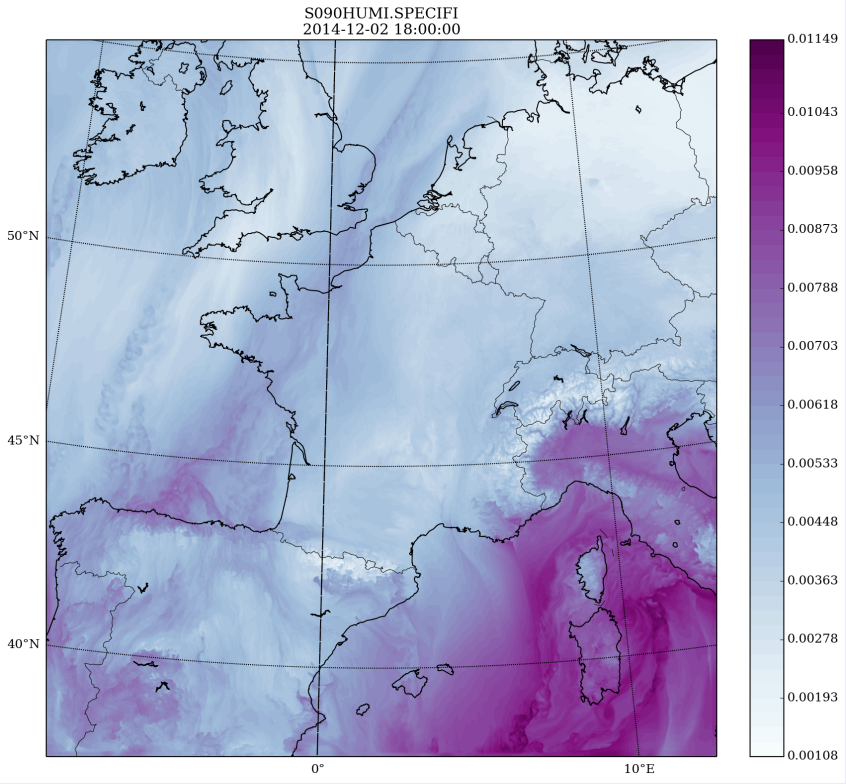
Field identifier (*fid*)

- reference to a format
- may be multiple in one single object
- often a reference to vertical position of a horizontal field (e.g. Z500, SST, ...)
- GRIB2 is potentially the most complete
  ⇒ `fid["generic"] = fid["GRIB2"]`

```
a_fid = {"FA":"L***PARAMETER",
         "LFI":(level, PARAMETER),
         "generic":{..., ...},
         ...}
```

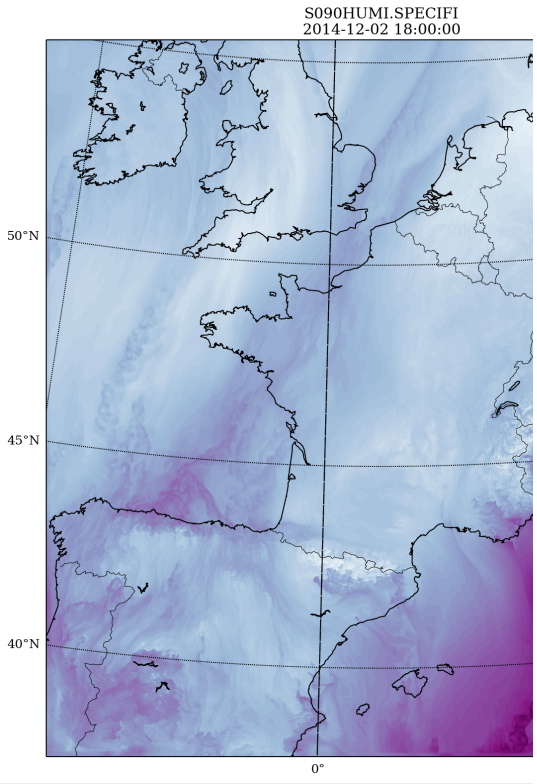
Fields

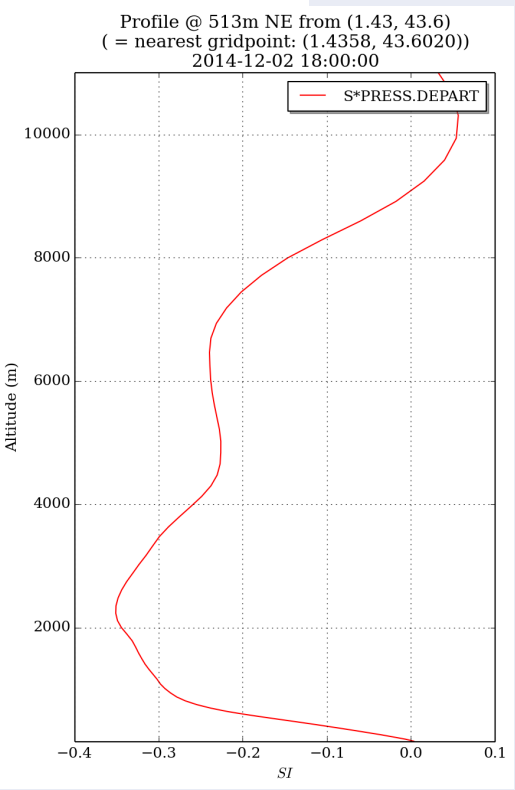
H2D,


S090HUMI.SPECIFI
2014-12-02 18:00:00
50°N
45°N
40°N
0°
10°E
0.01149
0.01043
0.00958
0.00873
0.00788
0.00703
0.00618
0.00533
0.00448
0.00363
0.00278
0.00193
0.00108

Fields

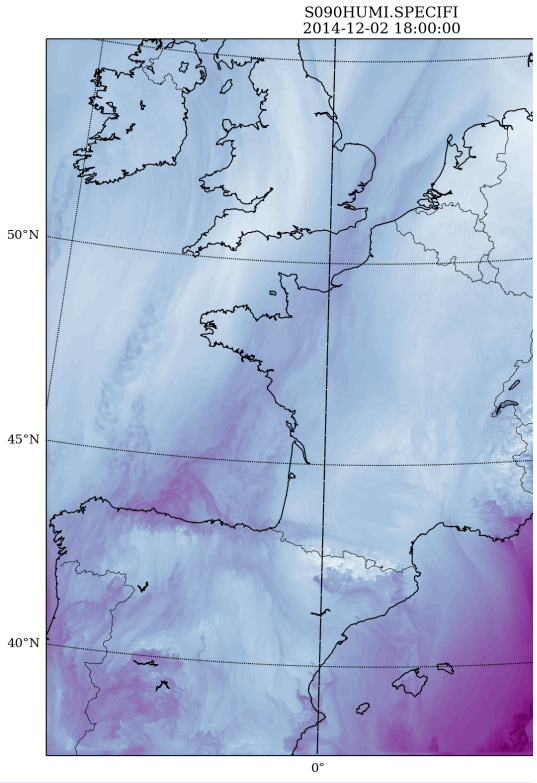
H2D, V1D,

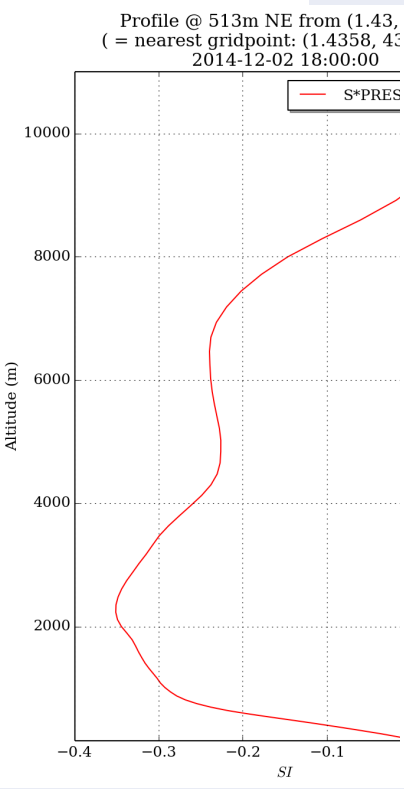


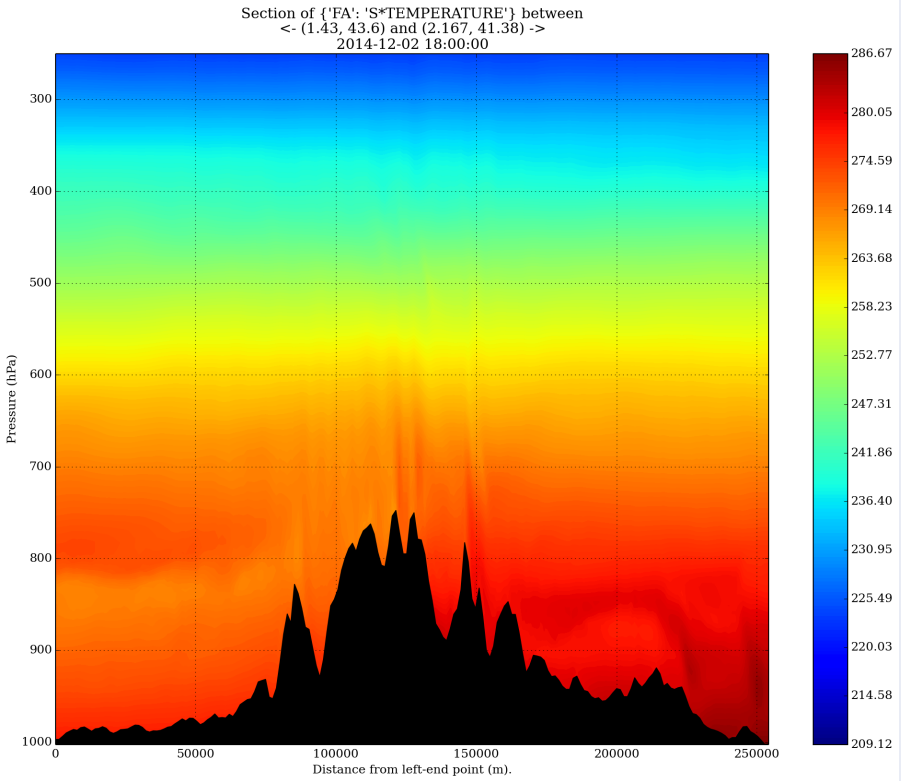


Fields

H2D, V1D, V2D...

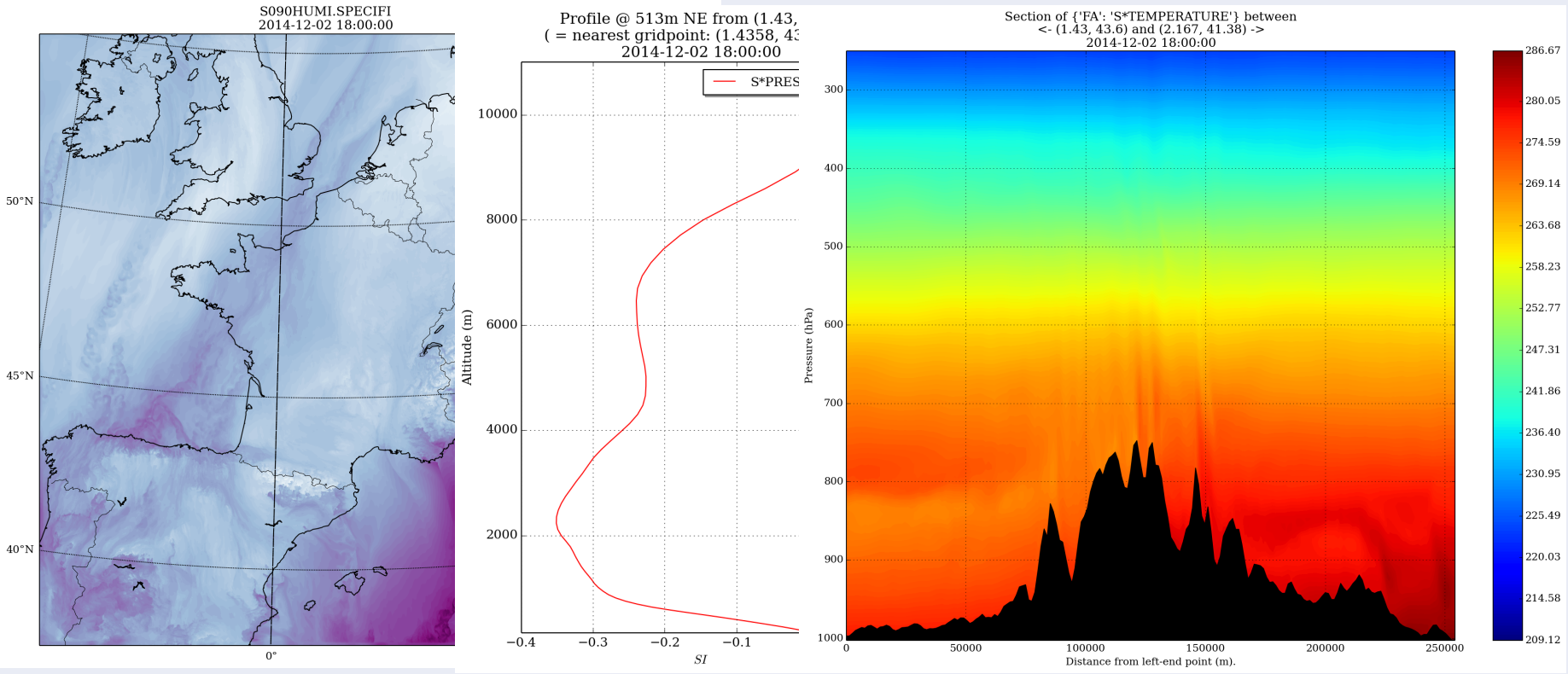






Fields

H2D, V1D, V2D... H1D, 3D, 0D.



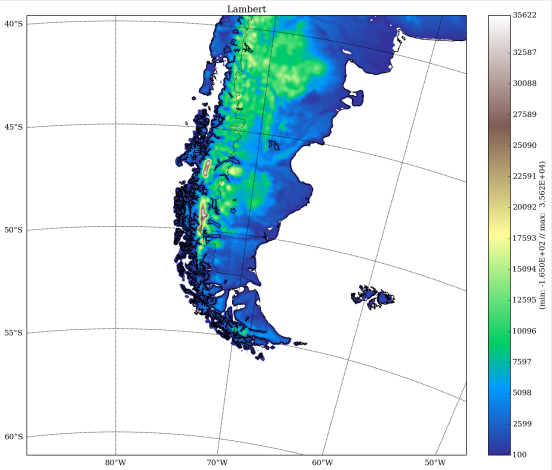
A certain level of service

- ease access to data/metadata, *avoid user to need to know how it works internally*
- integration of useful methods : may be useful to somebody else
- ex :
  - `a2Dfield.sp2gp()` ⇒ spectral → gridpoint transformation
  - `a2Dfield.getvalue_ll(1.44, 43.6)` ⇒ field value in Toulouse
  - `a2Dfield.plotfield(**kwargs)` ⇒ plot of the field, many options available in `kwargs` arguments
  - `a3Dfield.extract_subdomain(another_geometry, ...)` ⇒ extract a field in another geometry (e.g. a vertical profile)

Multiple geometries

epygram geometry objects implement :

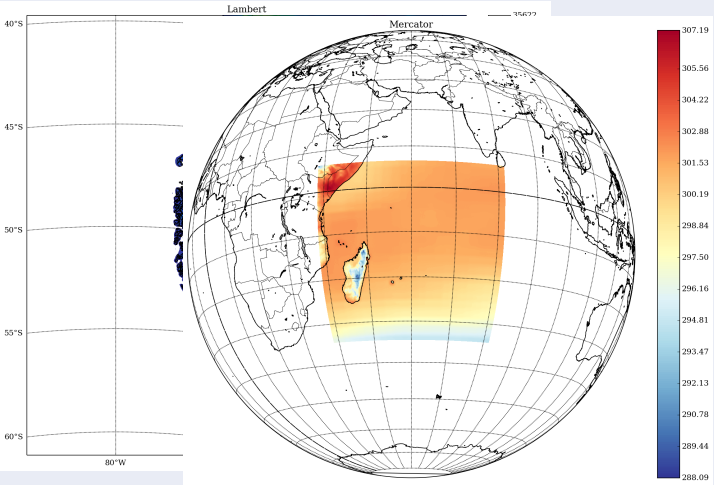
- on the horizontal : (secant/tangent) projections
  Lambert



Multiple geometries

epygram geometry objects implement :

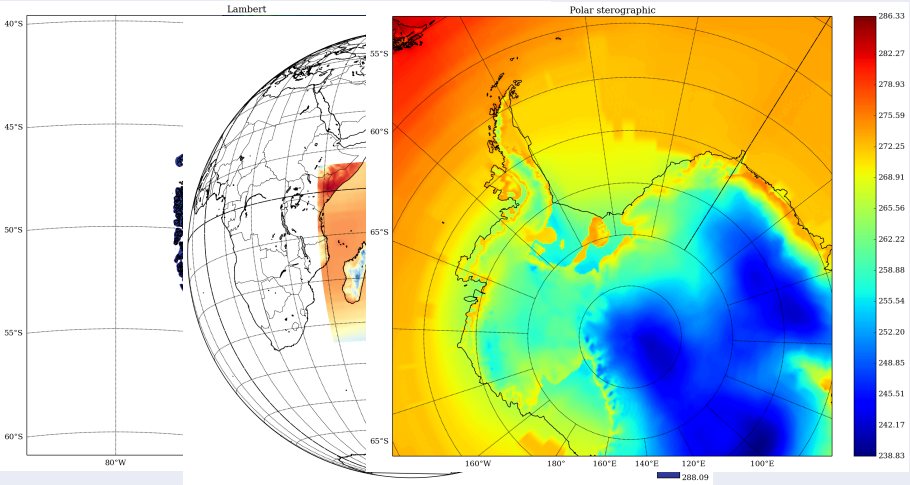
- on the horizontal : (secant/tangent) projections Lambert/Mercator



Multiple geometries

epygram geometry objects implement :

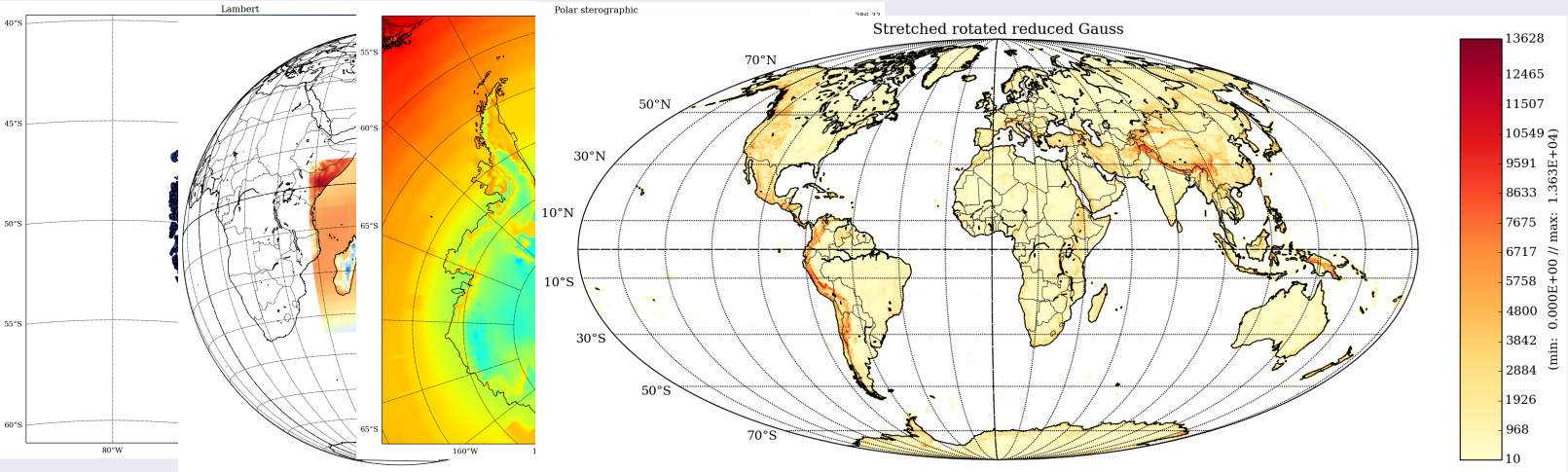
- on the horizontal : (secant/tangent) projections Lambert/Mercator/Polar-Stereographic,



Multiple geometries

epygram geometry objects implement :

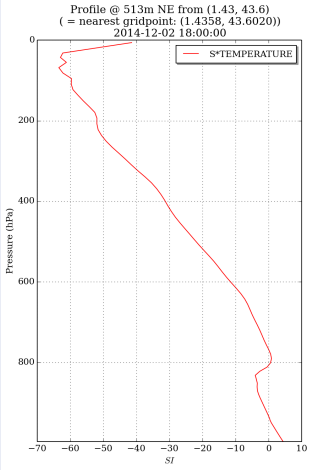
- on the horizontal : (secant/tangent) projections Lambert/Mercator/Polar-Stereographic, regular Lon/Lat, (reduced/stretched/rotated) Gauss grids, academic geometries & different geoids



Multiple geometries

epygram geometry objects implement :

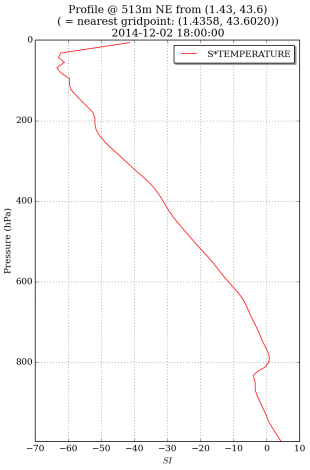
- on the horizontal : (secant/tangent) projections Lambert/Mercator/Polar-Stereographic, regular Lon/Lat, (reduced/stretched/rotated) Gauss grids, academic geometries & different geoids
- on the vertical : pressure,

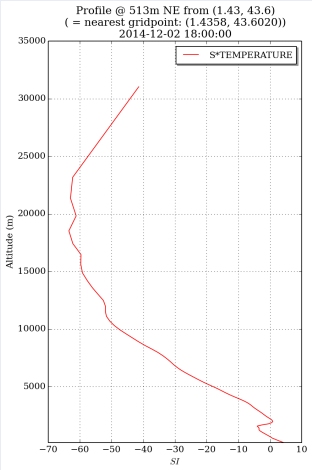


Multiple geometries

epygram geometry objects implement :

- on the horizontal : (secant/tangent) projections Lambert/Mercator/Polar-Stereographic, regular Lon/Lat, (reduced/stretched/rotated) Gauss grids, academic geometries & different geoids
- on the vertical : pressure, height/altitude,

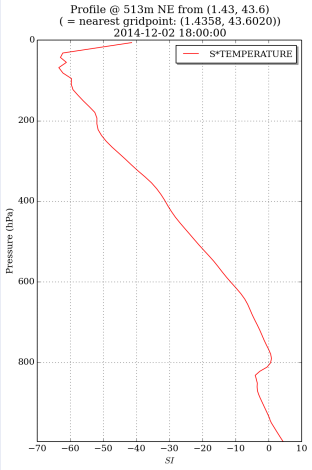


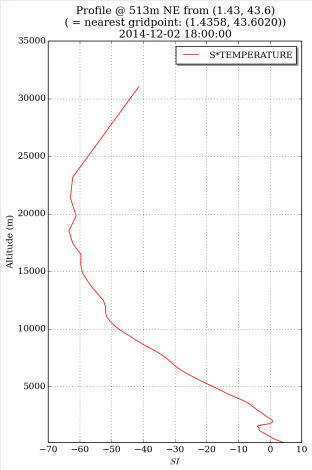


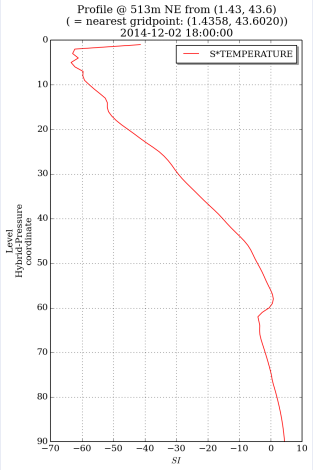
Multiple geometries

epygram geometry objects implement :

- on the horizontal : (secant/tangent) projections Lambert/Mercator/Polar-Stereographic, regular Lon/Lat, (reduced/stretched/rotated) Gauss grids, academic geometries & different geoids
- on the vertical : pressure, height/altitude, hybrid-pressure, hybrid-height coordinates







A certain level of service

- lon/lat ↔ gridpoint index :
  `aH2Dgeom.ij2ll(i, j) // aH2Dgeom.ll2ij(lon, lat)`
- LAM models : handling of C+I+E zones
- get the whole model grid `aH2Dgeom.get_lonlat_grid()`
- towards **matplotlib** : `aH2Dgeom.make_basemap(...)`
- vertical coordinate conversion functions, e.g. :
  `epygram.geometries.Vgeometry.hybridP2pressure(hybridP_geometry, Psurf=101500, vertical_mean="geometric")`
- others : `aH2Dgeom.distance(pt1, pt2)`, ...

Resource = **object** interface between the **epygram** `field` objects and IO read/write libraries of various formats.

- 3 opening modes `r, a, w`
- 3 superstar methods :
  - `r.listfields()`
  - `r.readfield(fid)`
  - `r.writefield(a_field)`
- some methods are specific to formats (encoding, metadata...)
- a *proxy* to open a resource whatever its format : `epygram.formats.resource()`

Resource = **object** interface between the **epygram** `field` objects and IO read/write libraries of various formats.

- 3 opening modes `r, a, w`
- 3 superstar methods :
  - `r.listfields()`
  - `r.readfield(fid)`
  - `r.writefield(a_field)`
- some methods are specific to formats (encoding, metadata...)
- a *proxy* to open a resource whatever its format : `epygram.formats.resource()`

tenuous bound between field and format : example

```
filein = epygram.formats.resource("ICMSHAROM+0042", "r")
fileout = epygram.formats.resource("ICMSHAROM+0042.nc", "w", fmt="netCDF")
ts = filein.readfield("SURFTEMPERATURE")
ts.fid["netCDF"] = "ts"
fileout.writefield(ts)
```

Resource = **object** interface between the **epygram** `field` objects and IO read/write libraries of various formats.

- 3 opening modes `r`, `a`, `w`
- 3 superstar methods :

Currently implemented formats

FA, LFI, LFA (+ DDH wrapping), GRIB1/2, netCDF, TIFFMF, GeoPoints

- some methods are specific to formats (encoding, metadata...)
- a *proxy* to open a resource whatever its format : `epygram.formats.resource()`

tenuous bound between field and format : example

```
filein = epygram.formats.resource("ICMSHAROM+0042", "r")
fileout = epygram.formats.resource("ICMSHAROM+0042.nc", "w", fmt="netCDF")
ts = filein.readfield("SURFTEMPERATURE")
ts.fid["netCDF"] = "ts"
fileout.writefield(ts)
```

Various other useful classes and functions :

- classes `FieldValidity, Angle, Spectrum`
- various functions : `epygram.util`
- default values & options ⇒ `epygram.config`
  "customizable" (userconfig)
- arguments catalog for `argparse` : `epygram.args_catalog`
- externalisable sub-modules : `epygram.untied`
  - Python interfaces to subroutines from IFS-ARPEGE-AROME code (FA, LFI, LFA, spectral transforms : `arpifs4py`),
  - *framework* for parallelisation of tasks on several cores ("threads") : `taylorism`
- ! underlying mechanics : package `footprints`

simplicity : plot a field from a FA file

```
> import epygram
> epygram.init_env()
```

simplicity : plot a field from a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
```

simplicity : plot a field from a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> ts = f.readfield("SURFTEMPERATURE")
```

simplicity : plot a field from a FA file

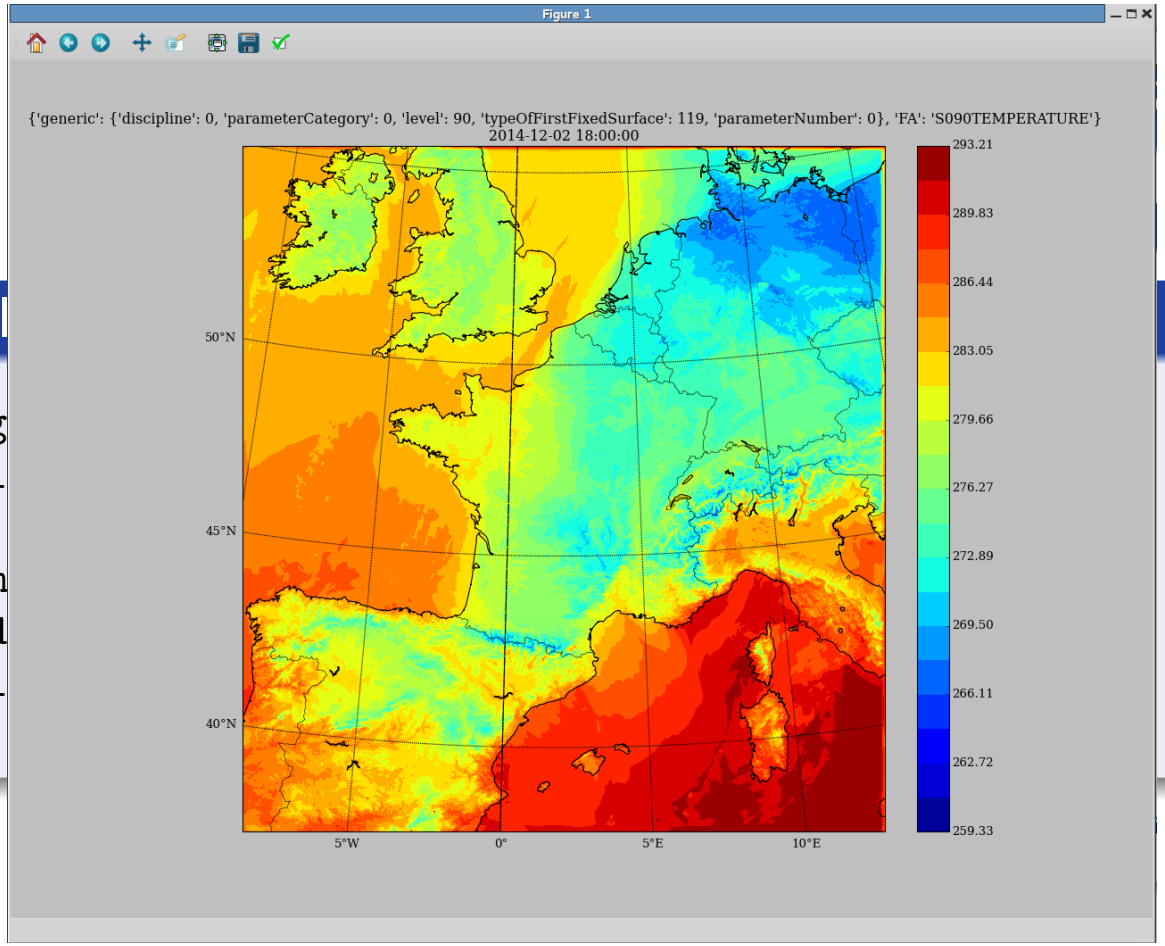
```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> ts = f.readfield("SURFTEMPERATURE")
> fig = ts.plotfield(... many plot options possible ...)
```

simplicity : pl

```
> import epyg
> epygram.ini

> f = epygram
> ts = f.read
> fig = ts.pl
> fig.show()
```



explore a FA file

```
> import epygram
> epygram.init_env()
```

explore a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
```

explore a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> f.isopen
True
```

explore a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> f.isopen
True
> f.format
"FA"
```

explore a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> f.isopen
True
> f.format
"FA"
> f.listfields()
["SURFTEMPERATURE", ... , "S090WIND.U.PHYS", ...]
```

explore a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> f.isopen
True
> f.format
"FA"
> f.listfields()
["SURFTEMPERATURE", ... , "S090WIND.U.PHYS", ...]
> print f.geometry
```

prints info about dimensions, projection, vertical levels, geoid... info contained or inferred from the FA header

explore a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> f.isopen
True
> f.format
"FA"
> f.listfields()
["SURFTEMPERATURE", ... , "S090WIND.U.PHYS", ...]
> print f.geometry
prints info about dimensions, projection, vertical levels, geoid... info
contained or inferred from the FA header
> f.geometry.dimensions["X"], f.geometry.dimensions["X_Iwidth"]
(1440, 16)
```

explore a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> f.isopen
True
> f.format
"FA"
> f.listfields()
["SURFTEMPERATURE", ... , "S090WIND.U.PHYS", ...]
> print f.geometry
prints info about dimensions, projection, vertical levels, geoid... info
contained or inferred from the FA header
> f.geometry.dimensions["X"], f.geometry.dimensions["X_Iwidth"]
(1440, 16)
> print f.validity.get()
2014-12-02 18:00:00
```

explore a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> f.isopen
True
> f.format
"FA"
> f.listfields()
["SURFTEMPERATURE", ... , "S090WIND.U.PHYS", ...]
> print f.geometry
prints info about dimensions, projection, vertical levels, geoid... info
contained or inferred from the FA header
> f.geometry.dimensions["X"], f.geometry.dimensions["X_Iwidth"]
(1440, 16)
> print f.validity.get()
2014-12-02 18:00:00
> f.validity.term()
datetime.timedelta(1, 64800)
```

play with a spectral field from a FA file

```
> import epygram
> epygram.init_env()
```

play with a spectral field from a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
```

play with a spectral field from a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> ps = f.readfield("SURFPRESSION")
```

play with a spectral field from a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> ps = f.readfield("SURFPRESSION")
> ps.spectral
True
```

play with a spectral field from a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> ps = f.readfield("SURFPRESSION")
> ps.spectral
True
> print ps.spectral_geometry
SpectralGeometry containing:
    truncation:
        in_X: 719
        in_Y: 767
        shape: elliptic
    space: bi-fourier
```

play with a spectral field from a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> ps = f.readfield("SURFPRESSION")
> ps.spectral
True
> print ps.spectral_geometry
SpectralGeometry containing:
    truncation:
        in_X: 719
        in_Y: 767
        shape: elliptic
    space: bi-fourier
> ps.sp2gp()
```

play with a spectral field from a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> ps = f.readfield("SURFPRESSION")
> ps.spectral
True
> print ps.spectral_geometry
SpectralGeometry containing:
    truncation:
        in_X: 719
        in_Y: 767
        shape: elliptic
    space: bi-fourier
> ps.sp2gp()
> ps.spectral
False
```

play with a spectral field from a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> ps = f.readfield("SURFPRESSION")
> ps.spectral
True
> print ps.spectral_geometry
SpectralGeometry containing:
    truncation:
        in_X: 719
        in_Y: 767
        shape: elliptic
    space: bi-fourier
> ps.sp2gp()
> ps.spectral
False
> ps.mean()
11.497678426124414
```

play with a spectral field from a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> ps = f.readfield("SURFPRESSION")
> ps.spectral
True
> print ps.spectral_geometry
SpectralGeometry containing:
    truncation:
        in_X: 719
        in_Y: 767
        shape: elliptic
    space: bi-fourier
> ps.sp2gp()
> ps.spectral
False
> ps.mean()
11.497678426124414
> ps.operation("exp")
```

play with a spectral field from a FA file

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> ps = f.readfield("SURFPRESSION")
> ps.spectral
True
> print ps.spectral_geometry
SpectralGeometry containing:
    truncation:
        in_X: 719
        in_Y: 767
        shape: elliptic
    space: bi-fourier
> ps.sp2gp()
> ps.spectral
False
> ps.mean()
11.497678426124414
> ps.operation("exp")
> ps.min(), ps.max()
(58666.409396722229, 103358.69845728)
```

plot a composition of fields from a GRIB file

```
> import epygram
> epygram.init_env()
```

plot a composition of fields from a GRIB file

```
> import epygram
> epygram.init_env()

> g22 = epygram.formats.resource("GRIDFRANGP0025r0_0022", "r")
> g24 = epygram.formats.resource("GRIDFRANGP0025r0_0024", "r")
```

plot a composition of fields from a GRIB file

```
> import epygram
> epygram.init_env()

> g22 = epygram.formats.resource("GRIDFRANGP0025r0_0022", "r")
> g24 = epygram.formats.resource("GRIDFRANGP0025r0_0024", "r")
> t2md = g24.readfield({"shortName":"2t"}) - g22.readfield({"shortName":"2t"})
```

plot a composition of fields from a GRIB file

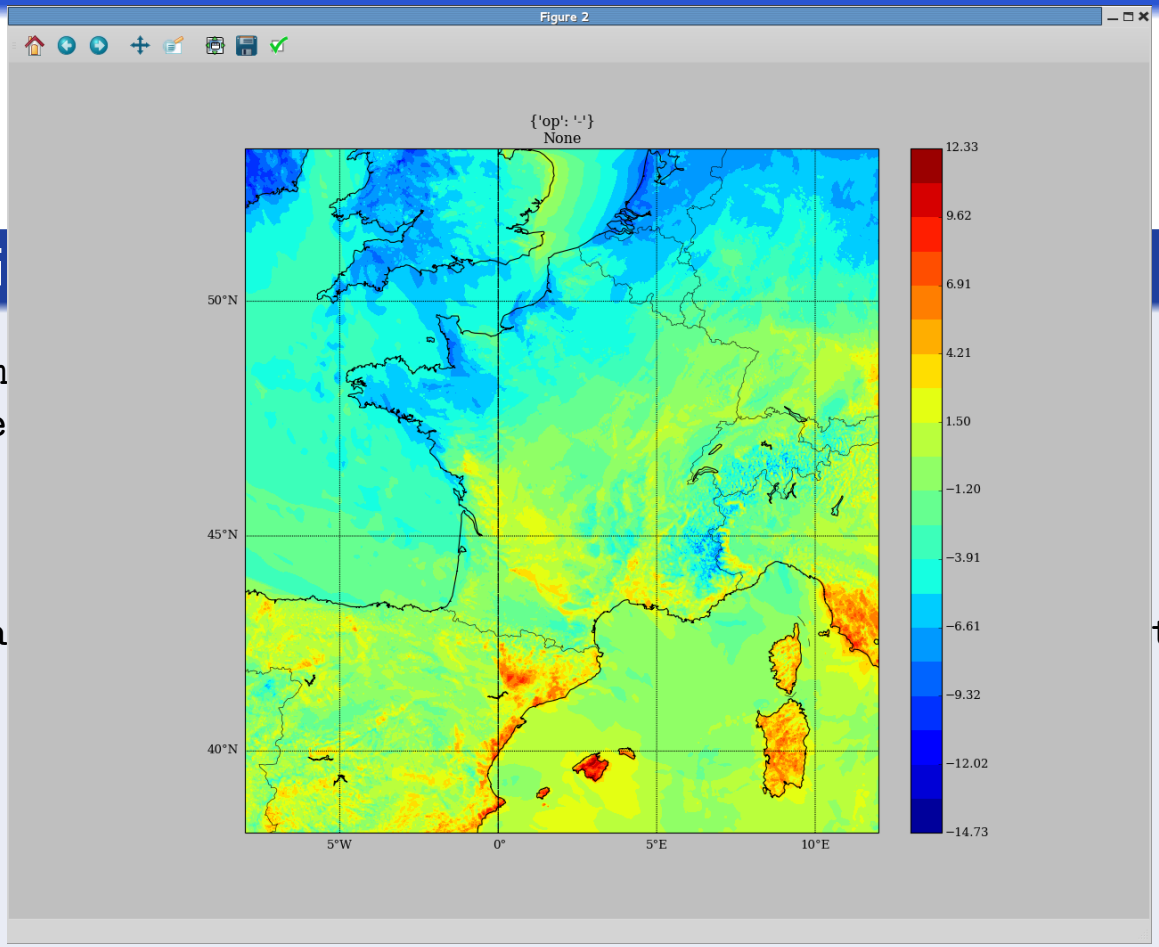
```
> import epygram
> epygram.init_env()

> g22 = epygram.formats.resource("GRIDFRANGP0025r0_0022", "r")
> g24 = epygram.formats.resource("GRIDFRANGP0025r0_0024", "r")
> t2md = g24.readfield({"shortName":"2t"}) - g22.readfield({"shortName":"2t"})
> figure = t2md.plotfield()
```

plot a compositi

```
> import epygram
> epygram.init_e

> g22 = epygram.
> g24 = epygram.
> t2md = g24.rea                                  tName":"2t"})
> figure = t2md.
> figure.show()
```



plot a composition of fields from a GRIB file

```
> import epygram
> epygram.init_env()

> g22 = epygram.formats.resource("GRIDFRANGP0025r0_0022", "r")
> g24 = epygram.formats.resource("GRIDFRANGP0025r0_0024", "r")
> t2md = g24.readfield({"shortName":"2t"}) - g22.readfield({"shortName":"2t"})
> figure = t2md.plotfield()
> figure.show()
> u = g22.readfield({"shortName":"u", "level":850})
> v = g22.readfield({"shortName":"v", "level":850})
```

plot a composition of fields from a GRIB file

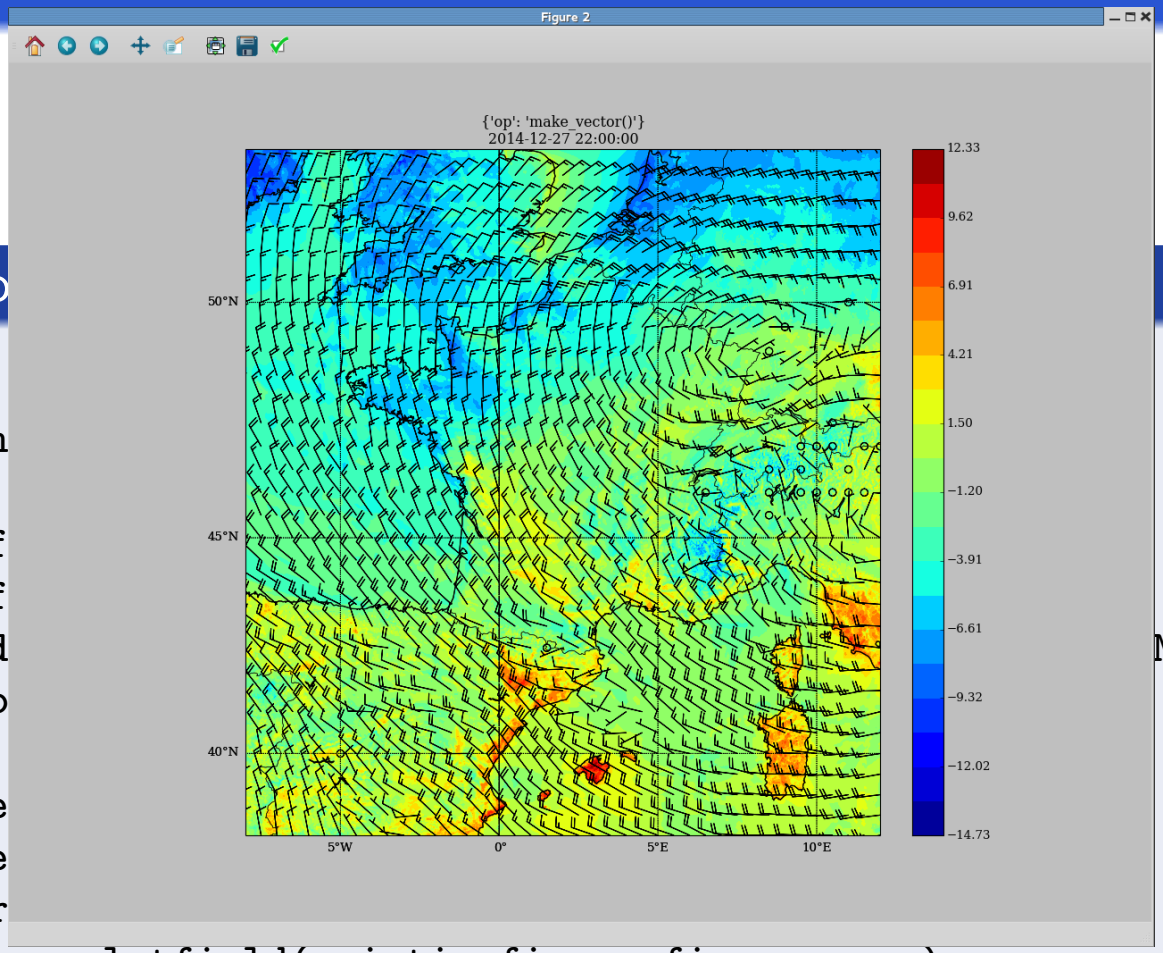
```
> import epygram
> epygram.init_env()

> g22 = epygram.formats.resource("GRIDFRANGP0025r0_0022", "r")
> g24 = epygram.formats.resource("GRIDFRANGP0025r0_0024", "r")
> t2md = g24.readfield({"shortName":"2t"}) - g22.readfield({"shortName":"2t"})
> figure = t2md.plotfield()
> figure.show()
> u = g22.readfield({"shortName":"u", "level":850})
> v = g22.readfield({"shortName":"v", "level":850})
> vect_uv = epygram.fields.make_vector_field(u, v)
```

plot a composition of fields from a GRIB file

```
> import epygram
> epygram.init_env()

> g22 = epygram.formats.resource("GRIDFRANGP0025r0_0022", "r")
> g24 = epygram.formats.resource("GRIDFRANGP0025r0_0024", "r")
> t2md = g24.readfield({"shortName":"2t"}) - g22.readfield({"shortName":"2t"})
> figure = t2md.plotfield()
> figure.show()
> u = g22.readfield({"shortName":"u", "level":850})
> v = g22.readfield({"shortName":"v", "level":850})
> vect_uv = epygram.fields.make_vector_field(u, v)
> figure2 = vect_uv.plotfield(existingfigure=figure, ...)
```



plot a compositio

```
> import epygram
> epygram.init_en

> g22 = epygram.f
> g24 = epygram.f
> t2md = g24.read                                    Name":"2t"})
> figure = t2md.p
> figure.show()
> u = g22.readfie
> v = g22.readfie
> vect_uv = epygr
> figure2 = vect_uv.plotfield(existingfigure=figure, ...)
> figure2.show()
```

play with geometry

```
> import epygram
> epygram.init_env()
```

play with geometry

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
```

play with geometry

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> t2m = f.readfield("CLSTEMPERATURE")
```

play with geometry

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> t2m = f.readfield("CLSTEMPERATURE")
> t2m.geometry.get_lonlat_grid()
(array_of_lon, array_of_lats)
```

play with geometry

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> t2m = f.readfield("CLSTEMPERATURE")
> t2m.geometry.get_lonlat_grid()
(array_of_lon, array_of_lats)
> t2m.geometry.distance((0.0, 45.1), (-2.1, 46.))
191712.97640887747
```

play with geometry

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> t2m = f.readfield("CLSTEMPERATURE")
> t2m.geometry.get_lonlat_grid()
(array_of_lon, array_of_lats)
> t2m.geometry.distance((0.0, 45.1), (-2.1, 46.))
191712.97640887747
> t2m.geometry.nearest_points(0., 45., interpolation="nearest")
(593, 618)
```

play with geometry

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> t2m = f.readfield("CLSTEMPERATURE")
> t2m.geometry.get_lonlat_grid()
(array_of_lon, array_of_lats)
> t2m.geometry.distance((0.0, 45.1), (-2.1, 46.))
191712.97640887747
> t2m.geometry.nearest_points(0., 45., interpolation="nearest")
(593, 618)
> t2m.geometry.gimme_corners_ll(subzone="C")
{"ul": (-12.039823, 54.664923), "ll": (-8.186537, 37.488412),
 "lr": (12.186537, 37.488412), "ur": (16.039823, 54.664923)}
```

play with geometry

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> t2m = f.readfield("CLSTEMPERATURE")
> t2m.geometry.get_lonlat_grid()
(array_of_lon, array_of_lats)
> t2m.geometry.distance((0.0, 45.1), (-2.1, 46.))
191712.97640887747
> t2m.geometry.nearest_points(0., 45., interpolation="nearest")
(593, 618)
> t2m.geometry.gimme_corners_ll(subzone="C")
{"ul": (-12.039823, 54.664923), "ll": (-8.186537, 37.488412),
 "lr": (12.186537, 37.488412), "ur": (16.039823, 54.664923)}
> t2m.getvalue_ll(0., 45.6)
277.4228471331433
```

play with geometry

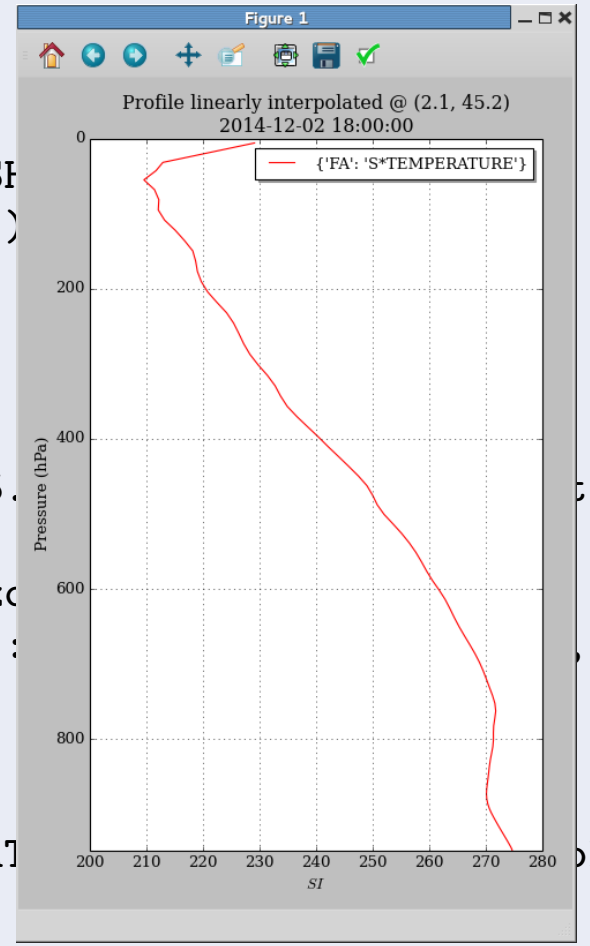
```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSHAROM+0042", "r")
> t2m = f.readfield("CLSTEMPERATURE")
> t2m.geometry.get_lonlat_grid()
(array_of_lon, array_of_lats)
> t2m.geometry.distance((0.0, 45.1), (-2.1, 46.))
191712.97640887747
> t2m.geometry.nearest_points(0., 45., interpolation="nearest")
(593, 618)
> t2m.geometry.gimme_corners_ll(subzone="C")
{"ul": (-12.039823, 54.664923), "ll": (-8.186537, 37.488412),
 "lr": (12.186537, 37.488412), "ur": (16.039823, 54.664923)}
> t2m.getvalue_ll(0., 45.6)
277.4228471331433
> prof = f.extractprofile("S*TEMPERATURE", 2.1, 45.2, interpolation="linear",
vertical_coordinate=100)
> fig = prof.plotfield()
```

play with geometry

```
> import epygram
> epygram.init_env()

> f = epygram.formats.resource("ICMSH
> t2m = f.readfield("CLSTEMPERATURE")
> t2m.geometry.get_lonlat_grid()
(array_of_lon, array_of_lats)
> t2m.geometry.distance((0.0, 45.1),
191712.97640087747
> t2m.geometry.nearest_points(0., 45.                    ")
(593, 618)
> t2m.geometry.gimme_corners_ll(subzo
{"ul": (-12.039823, 54.664923), "ll":
 "lr": (12.186537, 37.488412), "ur":
> t2m.getvalue_ll(0., 45.6)
277.4228471331433
> prof = f.extractprofile("S*TEMPERAT                    olation="linear",
vertical_coordinate=100)
> fig = prof.plotfield()
> fig.show()
```

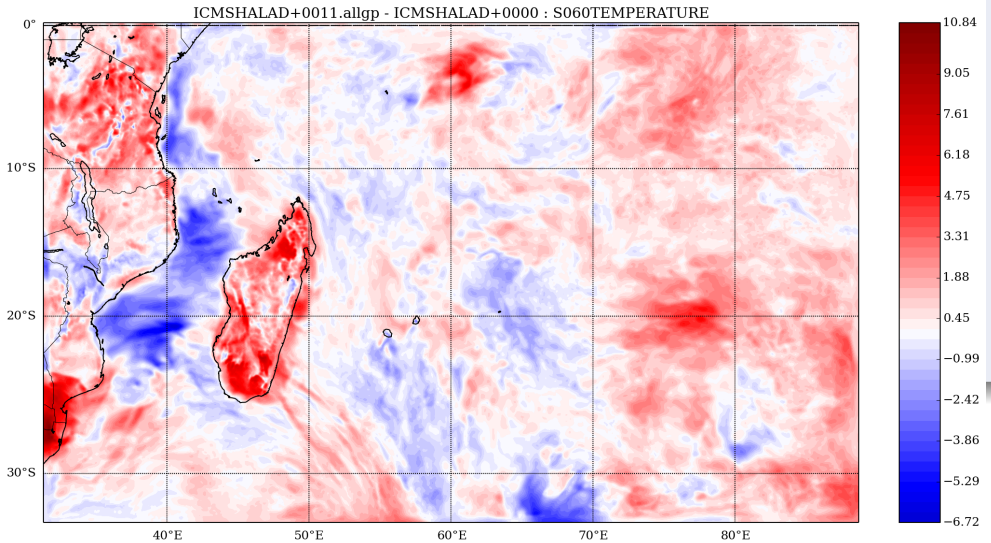


apptools

Some command line tools :

- applicative : `epy_plot.py`,

```
epy_plot.py -f S060TEMPERATURE ICMSHALAD+0011.allgp -D ICMSHALAD+0000
```



apptools

Some command line tools :

- applicative : `epy_plot.py`, `epy_stats.py`,

```
epy_stats.py ICMSHALAD+0011.allgp -d ICMSHALAD+0000 -f "*FTEMP*" -o
```

# PROFTEMPERATURE	min	max	mean	std	bias	rmsd	nonzero
ICMSHALAD+0011.allgp	2.764161E+02	3.104771E+02	2.980260E+02	3.431848E+00	-	-	3.907110E+05
ICMSHALAD+0000	2.762592E+02	3.104822E+02	2.981174E+02	3.398140E+00	-	-	3.907110E+05
-diff-	-4.261960E+00	3.791870E+00	-	-	-9.148226E-02	3.781067E-01	3.907110E+05
# SURFTEMPERATURE	min	max	mean	std	bias	rmsd	nonzero
ICMSHALAD+0011.allgp	2.796484E+02	3.359692E+02	2.995723E+02	6.556749E+00	-	-	3.907110E+05
ICMSHALAD+0000	2.716789E+02	3.028491E+02	2.970935E+02	3.658381E+00	-	-	3.907110E+05
-diff-	-1.230347E+00	4.428470E+01	-	-	2.478827E+00	7.546655E+00	3.907110E+05

apptools

Some command line tools :

- applicative : `epy_plot.py`, `epy_stats.py`, `epy_profile.py`, `epy_spectrum.py`, `epy_section.py`, `epy_point.py`

apptools

Some command line tools :

- applicative : `epy_plot.py`, `epy_stats.py`, `epy_profile.py`, `epy_spectrum.py`, `epy_section.py`, `epy_point.py`
- question a resource : `epy_what.py`

apptools

Some command line tools :

- applicative : `epy_plot.py`, `epy_stats.py`, `epy_profile.py`, `epy_spectrum.py`, `epy_section.py`, `epy_point.py`
- question a resource : `epy_what.py`
- fields handling : `epy_delfield.py`, `epy_movefield.py`, `fa_sp2gp.py`

apptools

Some command line tools :

- applicative : `epy_plot.py`, `epy_stats.py`, `epy_profile.py`, `epy_spectrum.py`, `epy_section.py`, `epy_point.py`
- question a resource : `epy_what.py`
- fields handling : `epy_delfield.py`, `epy_movefield.py`, `fa_sp2gp.py`
- format conversion : `epy_conv.py -o [grb|nc|geo]`

apptools

Some command line tools :

- applicative : `epy_plot.py`, `epy_stats.py`, `epy_profile.py`, `epy_spectrum.py`, `epy_section.py`, `epy_point.py`
- question a resource : `epy_what.py`
- fields handling : `epy_delfield.py`, `epy_movefield.py`, `fa_sp2gp.py`
- format conversion : `epy_conv.py -o [grb|nc|geo]`
- define a LAM domain : `domain_maker.py`

apptools

Some command line tools :

- applicative : `epy_plot.py`, `epy_stats.py`, `epy_profile.py`, `epy_spectrum.py`, `epy_section.py`, `epy_point.py`
- question a resource : `epy_what.py`
- fields handling : `epy_delfield.py`, `epy_movefield.py`, `fa_sp2gp.py`
- format conversion : `epy_conv.py -o [grb|nc|geo]`
- define a LAM domain : `domain_maker.py`
- **auto-documentation :** `epy_`*xxx*`.py -h`

Present status

- March 24th, 2016 : version 0.6.7 = pre-v1.0 | v1.0 $\Rightarrow$ May 2016 !
- Météo France - CNRM workstations and Bull supercomputers (`/home/gmap/mrpe/mary/public`)
- used in operations @ MF in Vortex scripts (date control, fields movements...)
- in β-test in successive development versions since Dec. 2014 : MF, ZAMG, DHMZ...
- HTML *Sphinx* documentation

Present status

- March 24th, 2016 : version 0.6.7 = pre-v1.0 | v1.0 $\Rightarrow$ May 2016 !
- Météo France - CNRM workstations and Bull supercomputers (`/home/gmap/mrpe/mary/public`)
- used in operations @ MF in Vortex scripts (date control, fields movements...)
- in β-test in successive development versions since Dec. 2014 : MF, ZAMG, DHMZ...
- HTML *Sphinx* documentation

Portability

- key point is to compile FA/LFI/spectral transforms Fortran interfaces
- designed for Python 2.7
- other (Python) dependances : numpy, scipy, matplotlib, pyproj, argparse, gribapi, netCDF4

Under progress...

- Web interface for visualisation of Olive experiments & oper/e-suites (G. Faure, MF)
- MF : installation on Git repository & Redmine project management interface
- *framework* for developing fields transformations, e.g. $\mathsf{T} \rightarrow \theta$ (S. Riette, MF)

Under progress...

- Web interface for visualisation of Olive experiments & oper/e-suites (G. Faure, MF)
- MF : installation on Git repository & Redmine project management interface
- *framework* for developing fields transformations, e.g. T $\rightarrow \theta$ (S. Riette, MF)

And tomorrow ?

- more advanced support for netCDF
- observations support ? ODB ? BUFR ?
- graphical aspects : possibility to use Magics++ ? 3D visualisation // temporal animation ?
- ***collaborative* library : any contribution is welcome !**

Thanks for your attention