An open source software suite for multi-dimensional geoscience data computation and visualization combined with GIS technology

Yaqiang Wang

Chinese Academy of Meteorological Sciences
Geoscience data analysis and visualization

- Multi-dimensional array computation
- Geoscience data analysis functions
- Normally used data formats (netCDF, GRIB…)
- Spatial data support, GIS functions
- Data visualization
- Script ability
- Data analysis and visualization library
Commercial software

ArcGIS, MatLab, IDL…

- Very expensive
- Not support popular meteorological data formats conveniently

NetCDF in ArcGIS

- Atmospheric community – ESRI collaboration
- Since 2006 release of ArcGIS 9.2 NetCDF CF format can be read in GIS
Freely available software – Atmospheric community

GrADS
NCL
Python: Numpy + Matplotlib + Basemap…
…

We may need open source “ArcGIS” + “MatLab”
MeteoInfo software package

- Motivation: Scientific data analysis and visualization with GIS ability – especially for meteorological community
- MeteoInfo library: Java and C# version
- MeteoInfo GIS desktop: Java and C# version
- MeteoInfoLab: Java and Jython
MeteoInfo: GIS Desktop application

- GIS functions were developed from ground level.
- Two editions: Java and C#.
- Freely and open source from the website http://www.meteothinker.com
- Cross-platform: Windows, Unix, Linux, Mac OS.
- Support netCDF, GRIB 1&2, HDF, GrADS, MICAPS, AWX ...
- Script ability.

(Wang 2014 Meteorological Applications)
MeteoInfo – Spatial and meteorological data operation in a GIS environment
MeteoInfo – Write plugins to do specific task

TrajStat plugin (open source)
(Wang et al., 2009 EMS)

Total spatial variation – number of clusters
MeteoInfo – Software development using the library
MeteoInfoLab: Scientific data analysis and visualization

MeteoInfoLab is open source software product developed using Java and Jython based on MeteoInfo Java library (Unidata netCDF Java library is used). The purpose is to provide an optional scientific computation and visualization tool similar with MatLab, NCL or Python (Numpy + Matplotlib + Basemap).
Multi-dimensional array support in Java
mipylib – Jython library of MeteoInfoLab

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset</td>
<td>Add svd and more functions</td>
</tr>
<tr>
<td>geolib</td>
<td>Add svd and more functions</td>
</tr>
<tr>
<td>meteolib</td>
<td>Rearrange the mipylib package</td>
</tr>
<tr>
<td>numeric</td>
<td>Update to version 1.4</td>
</tr>
<tr>
<td>plotlib</td>
<td>Add svd and more functions</td>
</tr>
<tr>
<td><strong>init</strong>.py</td>
<td>Add svd and more functions</td>
</tr>
<tr>
<td><strong>init</strong>.py</td>
<td>Add svd and more functions</td>
</tr>
<tr>
<td>miutil.py</td>
<td>Add random module</td>
</tr>
<tr>
<td>miutil.py</td>
<td>Add random module</td>
</tr>
</tbody>
</table>

Files: minum.py, miplot.py, midata.py
Mathematics and statistics components

Apache Commons™
http://commons.apache.org/

Examples:

```python
a1 = array([[25, 15, -5], [15, 18, 0], [-5, 0, 11]])
r1 = np.linalg.cholesky(a1)
print r1

a2 = array([[18, 22, 54, 42], [22, 70, 86, 62], [54, 86, 174, 134], [42, 62, 134, 106]])
r2 = np.linalg.cholesky(a2)
print r2
```

Result:

```python
>>> run script...
array([[ 5.0,  0.0,  0.0],
       [ 3.0,  3.0,  0.0],
       [-1.0,  1.0,  3.0]])
array([[ 4.242640687119285,  0.0,  0.0,  0.0],
       [ 5.185449728701349, 6.565905201197403,  0.0,  0.0],
       [12.727922061357857, 3.0460384954008553, 1.6497422479090682,  0.0],
       [ 9.899494936611667, 1.624553864213788, 1.8497110052313714, 1.3926212476455935]])
```
As you work in MeteoInfoLab, you issue commands that create variables and call functions. For example, create two variables named \( a \) and \( b \) by typing this statement at the command line:

```python
>>> a = 1
>>> b = 2
```

Then create \( c \) variable by adding \( a \) and \( b \). The variable value can be printed by typing variable name:

```python
>>> c = a + b
>>> c
3
```

And try other functions:

```python
>>> d = \cos(a)
>>> d
0.5403023058681398
>>> e = a \times b
>>> e
2
```

You can recall previous commands by pressing the up- and down-arrow key.
```python
>>> a = [1, 2, 3, 4]
>>> plot(a)
```
Also you can write a script program in editor panel and run it by clicking Run Script button in toolbar.
Array Indexing, Slicing and Iterating

One-dimensional arrays can be indexed, sliced and iterated over, much like lists and other Python sequences.

```python
>>> a = arange(10)**3
>>> a
array([  0,   1,   8,  27,  64, 125, 162, 343, 512, 729])
>>> a[2]
8
>>> a[2:5]
array([  8,  27,  64])
>>> a[4:6] = -1000  # equivalent to a[0:6:2] = -1000; from start to position 6, exclusive
>>> a
array([-1000,   1, -1000,  27, -1000, 125, 162, 343, 512, 729])
>>> a[::-1]  # reversed a
array([ 729,  512,  343,  162, -1000,  27, -1000,   1, -1000])
>>> for i in a:
...     print(i**(1/3.))
...
nan
1.0
nan
3.0
nan
5.0
6.0
7.0
8.0
9.0
```

Multidimensional arrays can have one index per axis. These indices are given in a tuple separated by commas:

```python
>>> b = array([[[ 0, 1, 2, 3],[10, 11, 12, 13],[20, 21, 22, 23],[30, 31, 32, 33],[40, 41, 42, 43]]])
>>> b[2,3]
23
>>> b[0:5, 1]
array([ 1, 11, 21, 31, 41])
>>> b[:,1,1]
array([ 1, 11, 21, 31, 41])
>>> b[1:3, :]
array([[[10, 11, 12, 13],
         [20, 21, 22, 23]]])
```

# each row in the second column of b
# equivalent to the previous example
# each column in the second and third row of b
MeteoInfoLab – Read meteorological data

```python
>>> f = addfile('model.ctl')
```

Query data file content including dimensions, attributes and variables by typing the data file variable:

```python
>>> f
File Name: D:/Temp/grads/model.ctl
Dimensions: 5
  X = 72;
  Y = 46;
  Z = 7;
  T = 5;
  T_z = 5;
X Dimension: Xmin = 0.0; Xmax = 355.0; Xsize = 72; Xdelta = 5.0
Y Dimension: Ymin = -90.0; Ymax = 90.0; Ysize = 46; Ydelta = 4.0
Global Attributes:
  : data_format = "GrADS binary"
  : fill_value = -2.56E33
  : title = "5 Days of Sample Model Output"
Variations: 8
float PS(T,Y,X);
  PS: description = "Surface"
float U(T,Y,X);
  U: description = "U"
float V(T,Y,X);
  V: description = "V"
float Z(T,Y,X);
  Z: description = "Geopotential"
float T(T,Y,X);
  T: description = "Temperature"
float Q(T,Z,S,Y,X);
  Q: description = "Specific"
float TS(T,Y,X);
  TS: description = "Surface"
float P(T,Y,X);
  P: description = "Precipitation"
```

Get data variable from data file object:

```python
>>> v = f['PS']
```

`PS` variable has 3 dimensions of time, latitude and longitude. Get 2 dimension data array from the data variable with slice by fixing time dimension:

```python
>>> ps = v[0,:,:]
>>> ps
```
**MeteoInfoLab – Plot data**

Plot map: create a map axes with axesm function, read shape file, view geodata layer:

```python
>>> axesm()     #Create a map axes
>>> mlayer = shaperead('D:/Temp/map/country1.shp')
>>> geoshow(mlayer, edgecolor=(0,0,255))
```

Save figure:

```python
>>> savefig('D:/Temp/test/press.png', 400, 300)
```

Create and plot filled contour layer from the dimension data array:

```python
>>> layer = contourfm(ps, 20)
```

Add title, x and y labels and colorbar:

```python
>>> title('Pressure')
>>> xlabel('Longitude')
>>> ylabel('Latitude')
>>> colorbar(layer)
```
Now try to get 0-D $z$ array (single value) along time dimension by fixing time, level, latitude and longitude dimensions:

```python
>>> hgt = f['Z'][0,[500],[40],[-90]]
>>> hgt
5759.111328125
```

Get 1-D $z$ array along longitude dimension and plot it:

```python
>>> hgt = f['Z'][0,[500],[40],[180,360]]
>>> clf()  # Clear figure
>>> plot(hgt, 'b-*')
```

![Plot of hgt data](image)
Get and plot 2-D \( z \) array with dimensions of latitude and longitude:

```python
>>> hgt = f['Z'][0,[500],[0,90],[180,360]]
>>> clf()
>>> axesm()
(org.meteoinfo.chart.plot.MapPlot@c3d5957, +proj=longlat +lat_0=0 +lon_0=0 +lat_1=30 +lat_2=)
>>> geoshow(mlayer, edgecolor=(0,0,255))
>>> layer = contourm(hgt)
>>> clabel(layer)
```
Another example, in this case with $X$ and $T$ varying (Hovmoller plot):

```python
>>> clf()
>>> hgt = f[Z][0:4,[500],[40],[180,360]]
>>> layer = contour(hgt, 10)
>>> clabel(layer)
>>> yaxis(axistype='time', timetickformat='yy-MM-dd')
>>> yticks(hgt.dimvalue(θ))
>>> xlabel('Longitude')
>>> ylabel('Date')
```
To alter the projection:

```python
>>> clf()
>>> axesm(proj='stere', lat_0=90, lon_0=-92, gridline=True)
(org.meteoinfo.chart.plot.MapPlot@bf0b58a, +proj=stere +lat_0=90 +lon_0=-92 +lat_1=30 +lat_2
>>> geoshow(mlayer, edgecolor='gray')
>>> hgt = f['Z'][0,[500],[15,80],[210,320]]
>>> layer = contourfm(hgt, 20)
>>> colorbar(layer)
```
Vertical Integrated Water Vapor Flux Divergency (2016-07-02)
MeteoInfoLab – More examples

Scores by group and gender

<table>
<thead>
<tr>
<th>Group</th>
<th>Men</th>
<th>Women</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>G2</td>
<td>35</td>
<td>54</td>
</tr>
<tr>
<td>G3</td>
<td>30</td>
<td>69</td>
</tr>
<tr>
<td>G4</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>G5</td>
<td>27</td>
<td>50</td>
</tr>
</tbody>
</table>

Histogram

Box plot demo

Buffer and Intersection
Introduction

Meteoinfo is a freely available software designed to view and analyze meteorological and spatial data interactively. Some GIS functions were developed from ground level. It has two editions: Java and C#. Meteoinfo may be run in Windows, Mac OS, Linux and Unix. Meteoinfo can also be run automatically using Meteoinfo scripting with the IronPython language (C# edition) or Jython language (Java edition). The main functions are packed in the Meteoinfo class library, which could be used to conveniently develop the software.

MeteoinfoLab is a free software product developed using Java and Jython based on Meteoinfo Java library (Unidata netCDF Java library is used). The purpose is to provide an optional scientific computation and visualization tool similar with Matlab and/or NCL.

Website: http://www.meteothinker.com
Source code: https://github.com/meteoinfo
Acknowledgement

- NetCDF Java: Available at [www.unidata.ucar.edu/software/netcdf-java](http://www.unidata.ucar.edu/software/netcdf-java)
- Jython: Available at [http://www.jython.org/](http://www.jython.org/)
- Proj4J: Available at [http://trac.osgeo.org/proj4j/wiki](http://trac.osgeo.org/proj4j/wiki)
- wContour: Available at [http://www.meteothinker.com/](http://www.meteothinker.com/)
- L2FProd: Available at [http://common.L2FProd.com](http://common.L2FProd.com)
- RSyntaxTextArea: Available at [http://fifesoft.com/rsyntaxtextarea/](http://fifesoft.com/rsyntaxtextarea/)
- JavaHelp: Available at [https://javahelp.java.net/](https://javahelp.java.net/)
- And more …
Thanks for your attention!

Welcome to use and develop MeteoInfo!