PyESG Documentation

Release 0.0.13

Feng Zhu

July 10, 2015

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION

The Python Library of Earth Spherical Geometry (PyESG) is aiming to perform spherical geometry on the earth and interpolate data from an unstructured mesh grid to another one, which is the so-called "regridding" or "remapping". Thus two main usages of PyESG are:

- Spherical geometry
- Regridding

and the second usage (regridding) is the main goal of this library.

There are some existing implementations of regridding, such as NCL and EMSF. NCL mainly performs interpolation between regular mesh grids, and recently (after the version of 6.1.0) it concludes the implementation of EMSF, which performs interpolation between any mesh grids. However, the APIs of EMSF are ported to specific global models that it is not so convinient to be used in a lightweighted way. By a lightweighted way I mean calling a function like:

```
func(data, mesh_old, mesh_new)
```

Therefore, PyESG is developed to make regridding between any mesh grids easier.

CHAPTER

TWO

SPHERICAL GEOMETRY

Spherical geometry is a subject about the geometry on a sphere. In this library, basic spherical geometries are introduced to serve the goal of regridding, so it may be not comprehensive for now. The main fuctionalities of spherical geometry in this library are introduced below.

2.1 Distance between two points

This is the so-called ‘‘

2.2 Waypoints during an arc

2.3 Intersection betweeen two arcs

2.4 Angle between two arcs

2.5 Triangle area

2.6 Triangle inside test

2.7 Quadrangle area

2.8 Quadrangle inside test

CHAPTER

THREE

REGRIDDING

This section will introduce the algorithm for regridding in the PyESG.

3.1 Basic Idea

The basic idea of regridding between two mesh grids, including the case of unstructured mesh grids, is interpolation. Spherical interpolation is very complicated, and it is assumed in PyESG that for a small regional area, planar interpolation is appropriage with limit error. Since the mesh grid is unstructured, the interpolation method used is **barycentric interpolation**.

To make the API easy to use, the Python function of regridding is like below:

```
pyesg.Interp.regrid(mesh_old, mesh_new)
```

This function dose not regrid a specific data, but calculate the regridding information when one performs regridding from `mesh_old` to `mesh_new`. If the mesh grids are two-dimensional, say `(nlat, nlon)`, then the shape of the return is `(3, nlat, nlon)`, which stores the location information and weights of the three points that used to calculate the interpolated result.

So how to realize this function?

- Step 1: *Search*

To calculate the interpolated value of a specific point `(lat, lon)` in the `mesh_new`, we need to find out the three points in the `mesh_old` used for interpolation.

- Step 2: *Interpolation*

After those three points are found, the barycentric interpolation can be performed.

3.2 Search

3.3 Interpolation

CHAPTER

FOUR

SOUCE CODE

class pyesg.**Arc**(*p1*, *p2*)
An arc on the earth defined by two points p1 and p2. It also can be seen as the relationship between two points.

distance()
Calculate the great-circle distance of the arc.

[reference: http://en.wikipedia.org/wiki/Great-circle_distance]

rad()
Convert great-circle distance on the earth to radians.

waypoint(*k*)
Calculate the location of a selected point (lat, lon) according to: + the location of point 1 (lat1, lon1); + the location of point 2 (lat2, lon2); + the coefficient k decides the position between point 1 and point 2,

e.g.: + when k = 0.0, (lat, lon) is point 1; + when k = 0.5, (lat, lon) is the mid-point; + when k = 1.0, (lat, lon) is point 2. [reference: http://en.wikipedia.org/wiki/Great-circle_navigation]

class pyesg.**Interp**
Interpolation algorithms.

barycentric(*point*, *triangle*)
(point, triangle) -> weight1, weight2, weight3

Barycentric Interpolation: interpolation in a triangle.

[reference: https://classes.soe.ucsc.edu/cmps160/Fall10/ resources/barycentricInterpolation.pdf]

regrid(*mesh_old*, *mesh_new*, *method='standard'*)
(mesh_old, mesh_new) -> matrix of weights and points indices.

Calculate the remapping coefficients (weights) from an old mesh to a new mesh. + When method == standard, the search algorithm can resolve any situation but slow; + When method == quick, the situation is that mesh_old and mesh_new are very similar to each other with some points nudged.

class pyesg.**Mesh**(*lat2d*, *lon2d*)
Unstructed mesh grids, which is defined by 2 dimenional arrays lat2d and lon2d.

class pyesg.**Point**(*lat*, *lon*)
A point on the earth defined by the latitude and longitude (unit: degree).

lat_deg()
radians -> degree

lon_deg()
radians -> degree

spherical_coord()
p (lat_rad, lon_rad) -> x, y, z

Return the (x, y, z) in a UNIT spherical coordinate system.

[reference: http://en.wikipedia.org/wiki/Spherical_coordinate_system]

vector()
Return the vector from the center of the Earth to the point.

class pyesg.**Quadrangle**(*p1*, *p2*, *p3*, *p4*)
An quadrangle on the earth defined by three points p1, p2, p3, and p4. It also can be seen as the relationship between four points.

Note: p1 -> p2 -> p3 -> p4 should be rotative.

angles()
Treated as two triangles.

area()
[reference: http://mathworld.wolfram.com/SphericalPolygon.html]

class pyesg.**Triangle**(*p1*, *p2*, *p3*)
An triangle on the earth defined by three points p1, p2, and p3. It also can be seen as the relationship between three points.

angles()
Calculate the included angle between two sides on the earth. If we set a is the side p2-p3, b the side p3-p1, and c the side p1-p2. Then the return value A is the included angle betwen sides b and c.

area()
Calculate the area of the triangle bounded by the sides made by the three points p1 (lat1, lon1), p2 (lat2, lon2), and p3 (lat3, lon3) according to the Girard's Theorem: $area = R^2 * E$, where R is the radius of the sphere, and E the angle excess: $E = A + B + C - pi$. Cosine rules are used to calculate the angles A, B, and C.

[references: http://www.princeton.edu/~rvdb/WebGL/GirardThmProof.html http://en.wikipedia.org/wiki/Spherical_trigonometry http://mathforum.org/library/drmath/view/65316.html]

CHAPTER

FIVE

ABOUT

5.1 Author

5.2 License

Unless stated otherwise, all files in the PyESG project, PyESG's webpage (and wiki), all images and all documentation including this User's Guide are licensed using the new BSD license:

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

p