Interpolating climate data using CDAT

Alex Pletzer, Dave Kindig, and Srinath Vadlamani (Tech-X) – LibCF/GRIDSPEC
Paul Durack, Charles Doutriaux, Jeff Painter, and Dean Williams (LLNL) – CDAT, CMIP5
Ryan O'Kuinghttons, Bob Oehmke (NOAA) – ESMF

pletzer@txcorp.com

Jan 23 2012, AMS, New Orleans
Work funded by MoDAVE: DOE/SBIR DE-FG02-08ER85153
CDAT = Climate Data Analysis Tools

> Written in Python
> Understands data conforming to Climate Forecast (CF) conventions
> Ultra Visualization (UV)-CDAT brings together CDAT, VisIt, Paraview, VisTrails, R, ...
> Run CDAT engine on Earth System Grid to serve climate data

```python
>>> import cdms2
>>> f = cdms2.open('./contrib/ZonalMeans/Test/test_data.nc')
>>> f.listvariables()
['bounds_depth', 'bounds_time', 'bounds_y', 'bounds_x', 'O2']
>>> o2 = f('O2')
>>> o2.listattributes()
['units', 'long_name', 'name']
>>> o2.units
'mol/m^3'
>>> o2.shape
(12, 5, 150, 180)
```

http://www2-pcmdi.llnl.gov/cdat
CDAT can easily be extended, leveraging a vast body of third party Python modules

- CDAT builds many packages including scipy, ipython, Pmw, PyQt, .... Extending CDAT can be as simple as typing “python setup.py install”

- Examples:
  - mpi4py (Message Passing Interface for Python)
  - petsc4py (sparse matrix solvers, non-linear equations, time steppers, ...) [Lisandro Dalcin]
  - PyGNL
  - PyLog (PROLOG engine)
  - nltk (Natural Language Toolkit)
Our focus here is interpolation

- Explore different interpolation options provided within CDAT
  - Module regrid2
- Module gsRegrid: new in CDAT 6.0
- ESMP: Earth System Modeling Framework (ESMF) interpolation (not part of CDAT)

- Evaluate each approach using native Coupled Model Intercomparison Program Phase 5 (CMIP-5) data from various models
  - Source grids are typically not latitude-longitude
Some of the challenges of CMIP-5 and other grids

- Models use non-uniform grids to avoid numerical problems and over-resolution near poles
  - Rotated pole
  - Tripolar, Murray '96
  - Small degree of unstructuredness
cubed-sphere
Interpolation must be able to handle...

- Extremely flat cells near poles
- Recognize longitudes as periodic coordinate
- Nodal versus cell centered data
- Gap of tripolar grid near pole
Currently available options for regridding in Python/CDAT

- **regrid2** *(in CDAT 5.2)*
  - 2D, horizontal grid is a cross product of axes

- **SCRIP** *(in CDAT 5.2)*
  - 2D, curvilinear grids, conservative/linear/spline. Lacks documentation (was not able to use)

- **LibCF**
  - Multi-dimensional but only linear *(in CDAT 6.0)*. Interface to C library using *ctypes*.

- **ESMF/ESMP**
  - 2D/3D, option between linear, conservative, Python interface recently made available by Ryan O'Kuingletttons. Interface to C ESMF *(ESMC)* via *ctypes*.
LibCF regridding/interpolation

- Linear interpolation using nearest neighbors only
  - No over-shooting
  - Straightforward to parallelize
- Pseudo-Newton search of position in index space
  - Only one iteration required for uniform, rectilinear grids
- Line search to improve convergence
- Use previous index location as initial guess when regridding from structured to structured grids
- Handles dateline, can be anywhere
- Pole remains a problem
- Has support for masking

Pathological case has zero cell volume in lon-lat space
How LibCF deals with masking

- Will do its best to interpolate in the presence of masked (or invalid) values
- 3 cases:
  - All values in a cell are valid
  - Some invalid values:
    - Switch from quadrilateral/hexahedron to triangle/tetrahedron interpolation

All nodal values are valid
One missing value interpolation is still possible
Not possible to interpolate
How to call LibCF regrid from CDAT

from cdms2 import gsRegrid
...
# .... src_y, src_x can be curvilinear coordinates
# or axes, ditto for dst_y, dst_x, ....
# takes numpy or cdat cdms2 type variables
src_grd = [..., src_y, src_x]
dst_grd = [..., dst_y, dst_x]

# constructor
rg = gsRegrid.Regrid(src_grd, dst_grid,
                    mkCyclic = False,
                    handleCut = False,
                    src_bounds = None)

# compute interpolation weights
rg.computeWeights(nitermax=20, tolpos=0.01)

# interpolate src_field, result is dst_field
rg(src_var, dst_var)
LibCF: 2D interpolation was tested on 23 ocean models

... etc.
LibCF: GFDL model was made cyclic and additional row was added to fill in gap

DB: salinity2D_GFDL-ESM2M_1pctCO2_0180-0360.vsh5

Pole is well resolved
Tripolar grid, no Gap
No dateline problem
LibCF: interpolation of CNRM model shows small gap

Pole less well resolved
Small gap
Interpolation error after interpolating back onto the source grid

- Error is mostly near the coast line
LibCF: 3D test cases

- Takes ~ 20-60 seconds (only 10 levels)
- MIROC hi-res model
Summary

• Highly distorted lat-lon grids present challenges for interpolation software
  – Cuts
  – Jump in longitude
  – Pole
• LibCF interpolation has benefited from being exposed to “real” datasets
• Timings: gsRegrid takes ~ few seconds for 2D, ~40 seconds for 3D (need to understand why 3D takes so much longer)
• Can apply domain decomposition and MPI parallelization to accelerate weight computation (embarrassingly parallel)
• Lack of conservation ~ 2%. Can be “fixed” globally by multiplying weights by a constant factor
Summary (2)

- ESMF interpolation likely to offer best solution when conservation is required
  - Actively working with ESMF developers to extend Python API
  - Work by Peggy Li [ESMF Offline Regrid Generator Performance Comparison with SCRIP] shows good scalability and accuracy for atmospheric model