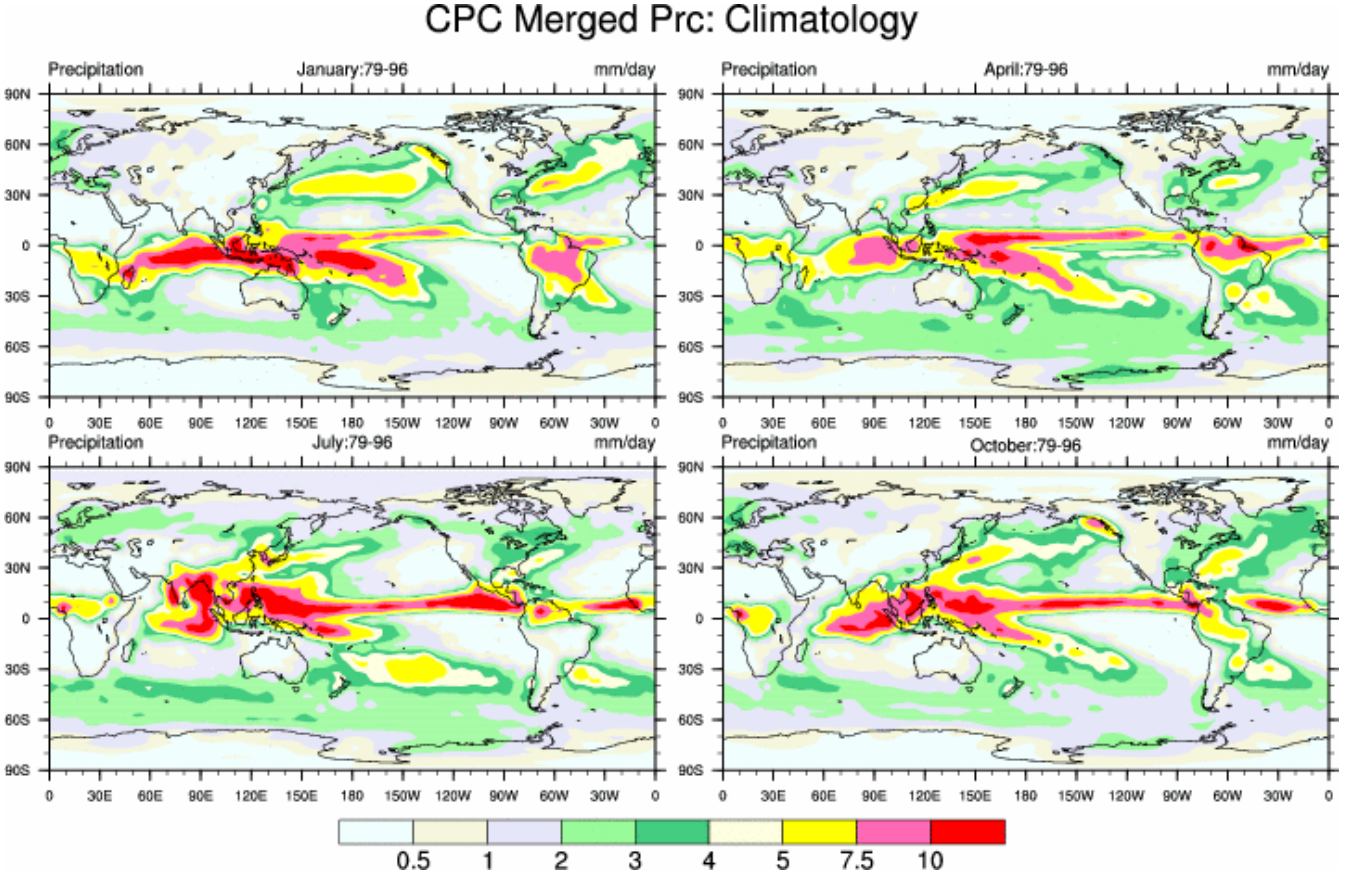
NCL Data Processing



Dennis Shea

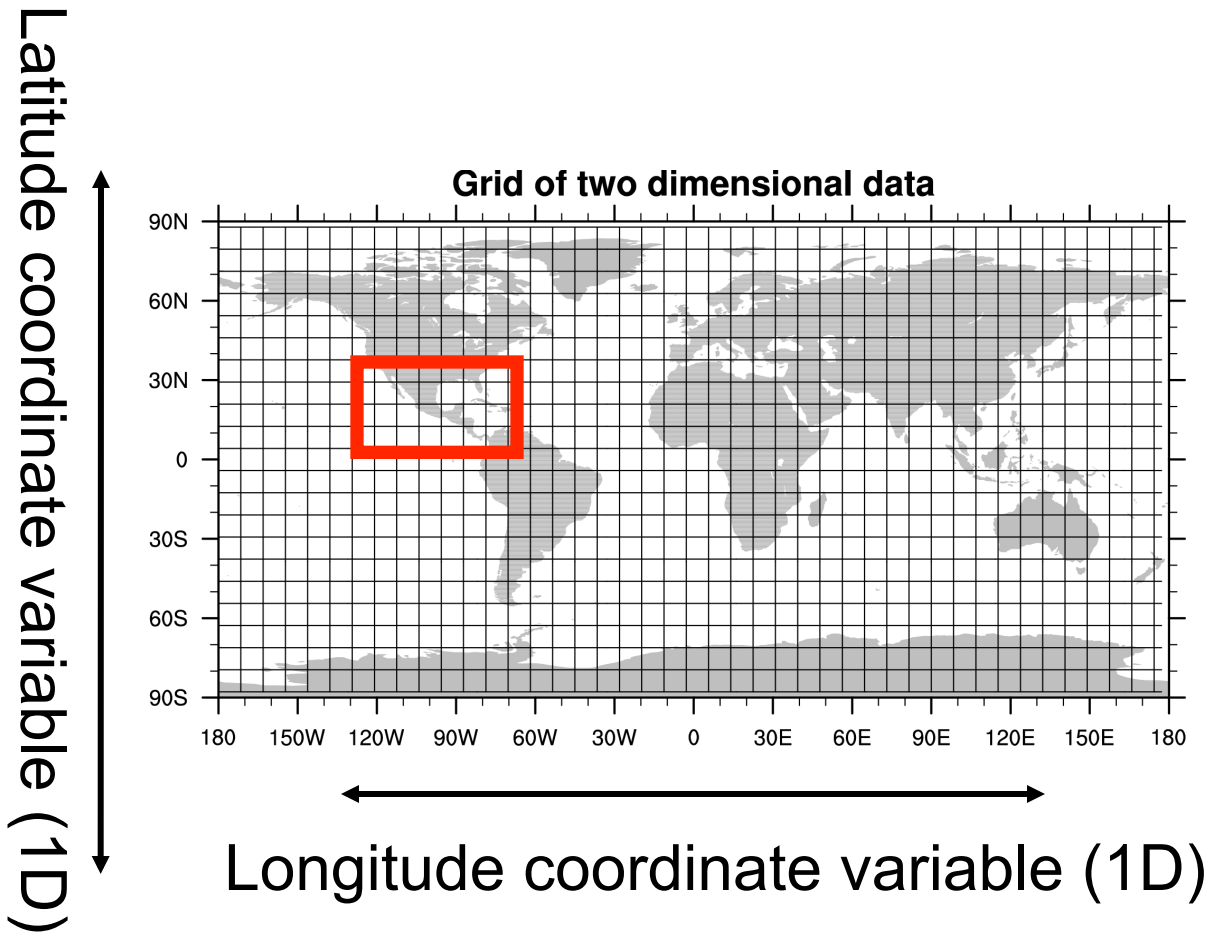
National Center for Atmospheric Research

NCAR is sponsored by the National Science Foundation

Grid(s)

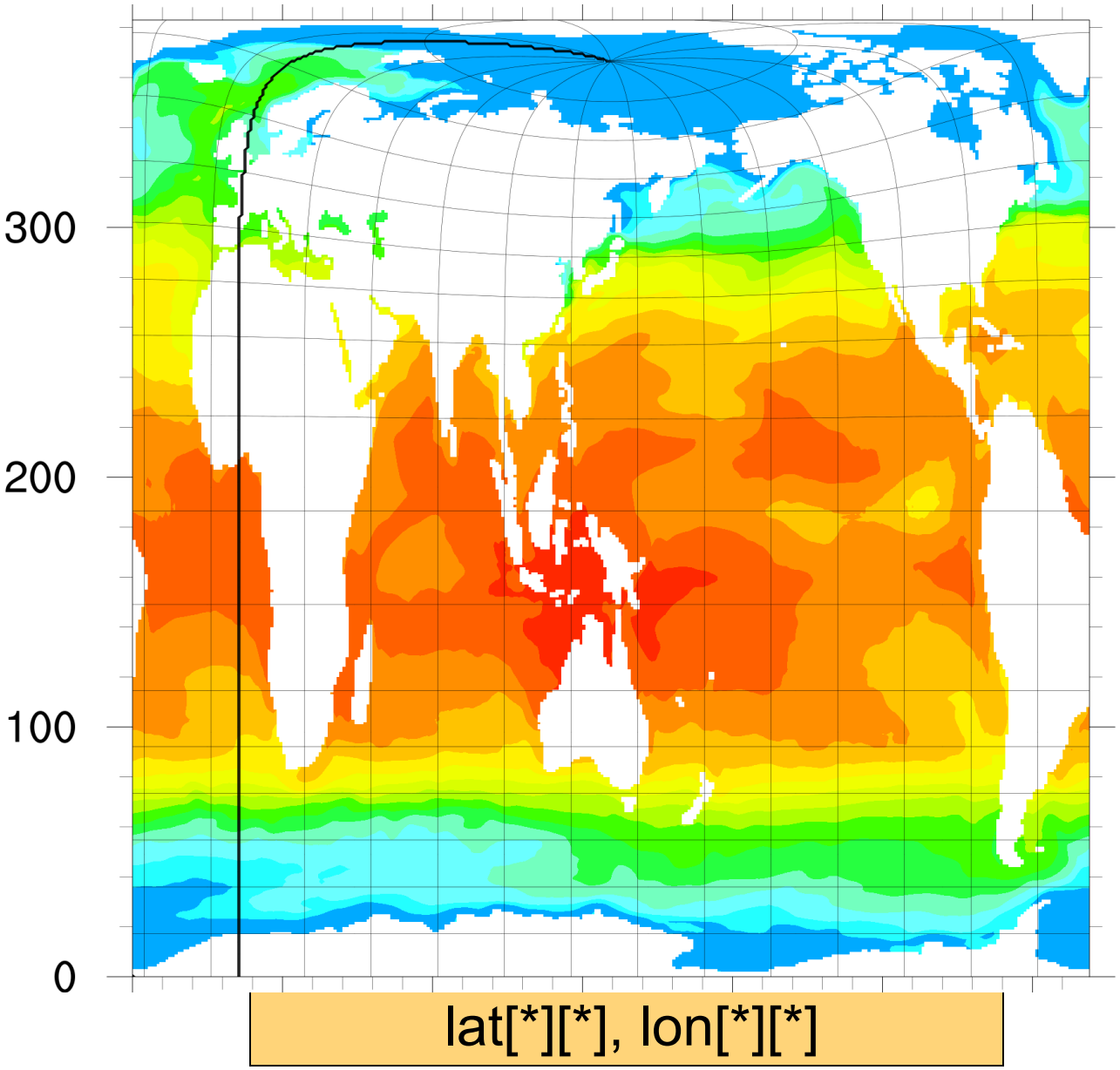
- Grid (Mesh)
  - a well-defined spatial structure
- Common Grids: Models & Reanalyses
  - **Rectilinear**
    - 1x1, 2x3, gaussian, FV, Global Reanalysis
    - x(…,lat,lon), lat(lat), lon(lon)
  - **Curvilinear**
    - WRF, POP, GODAS, RegCM, NARR
    - y(…,nlat,mlon), lat2d(nlat,mlon), lon2d(nlat,mlon)
  - **Unstructured**
    - SE (Spectral Element), FE, MPAS
    - z(…,npts), lat(npts), lon(npts)
- Why different grids?
  - advances in computer architecture
  - computational efficiency
  - addressing pole singularities
  - better representation physics and/or dynamical core

Generic Rectilinear Grid: lat[*], lon[*]

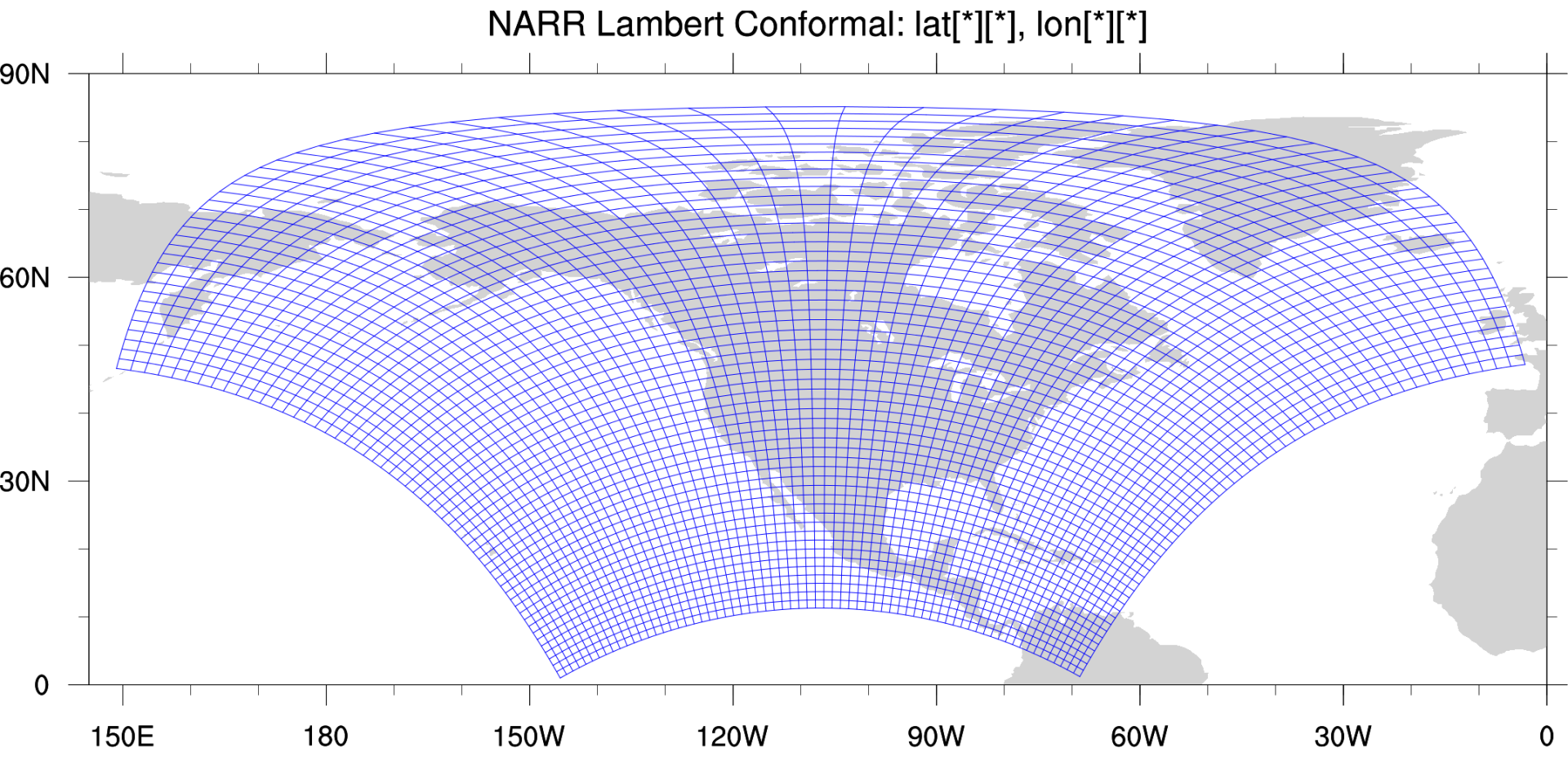


lat, lon need NOT be equally spaced: gaussian, MOM, FV

Sample Curvilinear Grid: Early POP

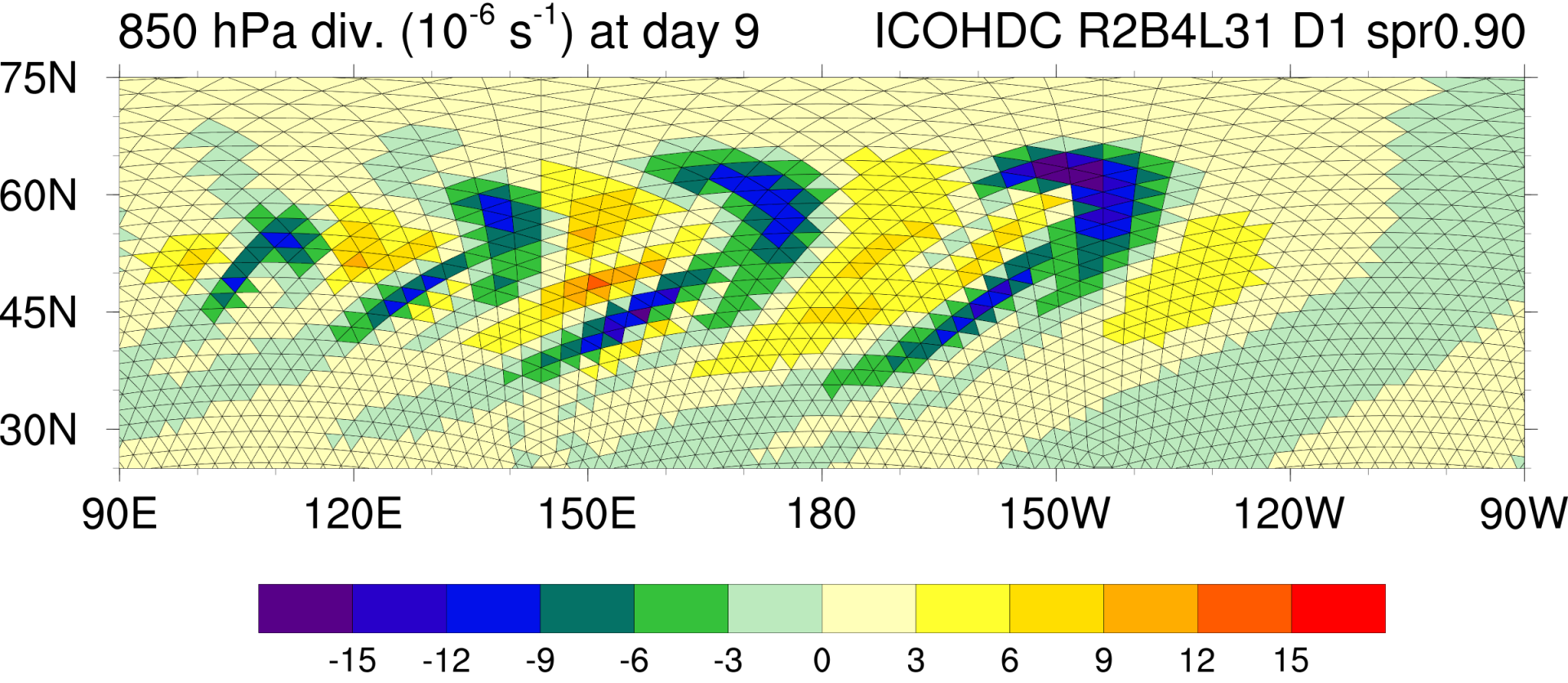


Sample Curvilinear Grid: NARR

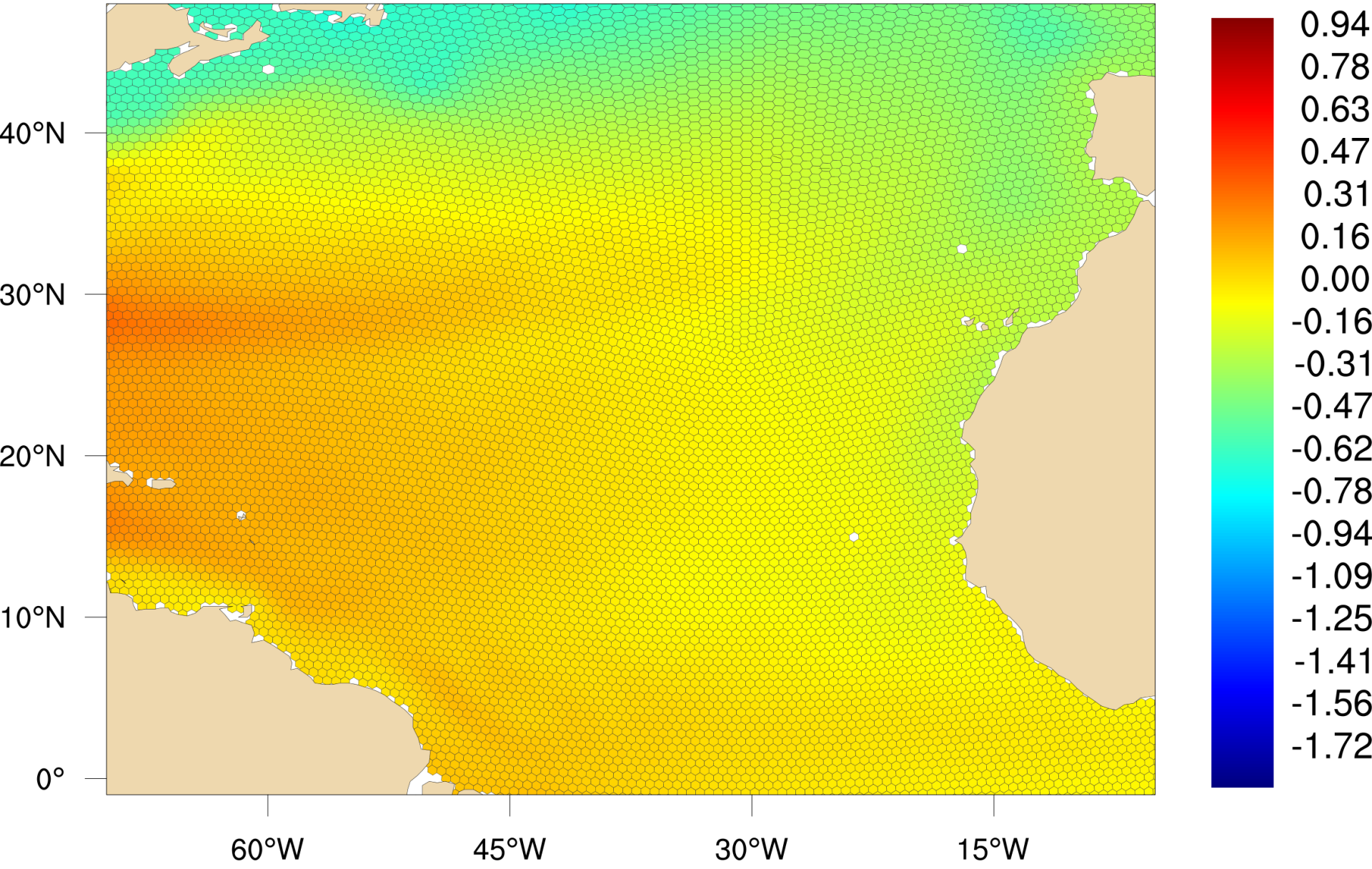


lat[*][*], lon[*][*]

Unstructured Grid: ICON



Unstructured Grid: MPAS



Regrid & Comments

- Regrid
  - **interpolation** of one well defined spatial structure to another; **horizontal** or **vertical**

- General Comments
  - quantitative evaluation of data on different grids generally requires regridding to a common grid
  - regrid low res (5x5) to high res (1x1) does **NOT** provide more information than the low res (5x5)
  - generally: interpolate high res grid to low res
  - derive quantities on original grid then regrid
  - vector quantities (*eg*, u,v) should be regridded together. Alternatively, derive a scalar quantity on the original grid (eg: divergence, vorticity), interpolate the scalar quantity; then rederive the vector components from the interpolated scalar
  - **extrapolation** should be done with **caution**

Common Regrid Methods

- **Functions**: http://www.ncl.ucar.edu/Document/Functions/regrid.shtml
- **Examples**: https://www.ncl.ucar.edu/Applications/regrid.shtml
- http://www.ncl.ucar.edu/Applications/ESMF.shtml

- Method: appropriate for spatial structure and intended usage
  - smooth variables (*eg*: T, SLP): 'any' method can be used
  - fractal (*eg:* 3-hr PRC): some form of local areal avg
  - flux quantities: conservative
  - categorical: nearest neighbor (ideally use mode)

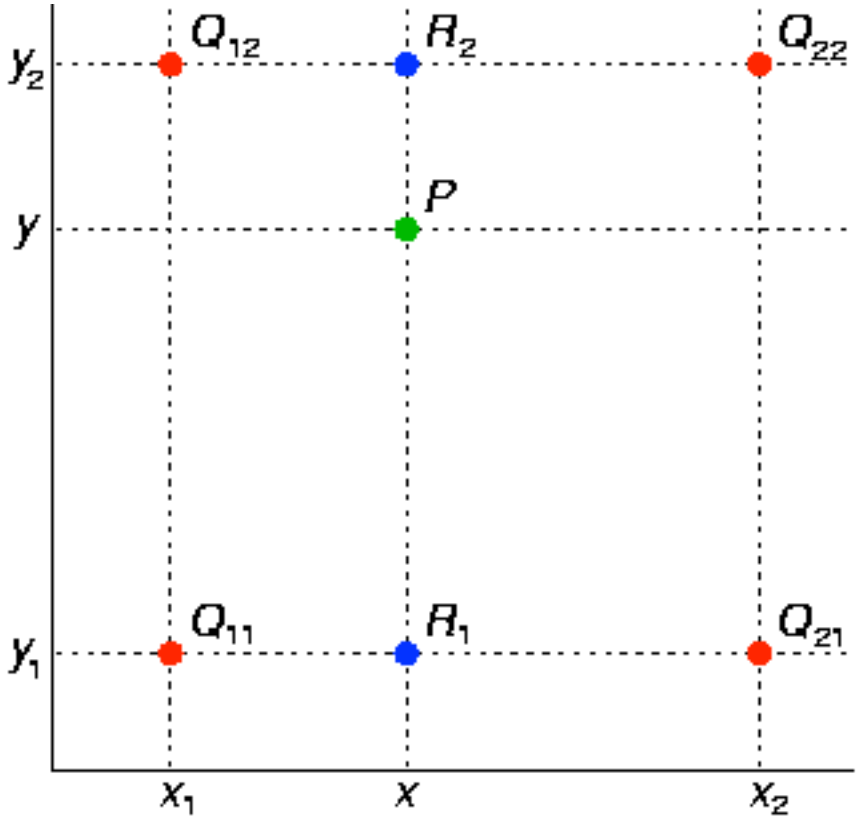
Regrid: bilinear interpolation
linint2_Wrap (linint2)

- **rectilinear** grids only: Cartesian, global or limited area
- most commonly used
- use when variable is reasonably smooth
- uses the four closest grid points of source grid
- missing data allowed but not filled in
- extrapolation is not performed
- **_Wrap preserves** attributes; **creates** coordinate variables

```
LON  = …    ; from a file, function or manually create
LAT  = …

f       = addfile (“T2m.nc", "r")
T       = f->T2m
TBLI  = linint2_Wrap(T&lon, T&lat, T, True,  LON, LAT, 0 )
printVarSummary(TBLI)
```

Bilinear Interpolation

The four red dots show the data points and
the green dot is the point at which we want to interpolate



source: en.wikipedia.org/wiki/wiki/Bilinear_interpolation

Regrid: areal conservative interpolation
area_conserve_remap_Wrap

- **global rectilinear grids only**
- **_Wrap** preserves attributes; creates coordinate variables
- missing data (_FillValue) *NOT* allowed

In particular, use for (say) flux or precipitation interpolation

```
f   = addfile ("GPCP.nc", "r")
p   = f->PRC
P   =  area_conserve_remap_Wrap (p&lon, p&lat, p          \
                                 ,newlon, newlat, False)
```

regrid: areal average interpolation
area_hi2lores_Wrap

- **rectilinear** grids; can be limited area
- **_Wrap** preserves attributes; creates coordinate variables
- missing data allowed
- designed for TRMM data

NOT strictly ‘conservative’ but close for (say) 50S to 50N

Use **area_hi2lores_Wrap** for fractal fields => lower res

```
f  = addfile (trmm.nc", "r")
p = f->PRC
P =  area_hi2lores_Wrap (p&lon, p&lat, p, True, wlat,  LON, LAT, 0 )
```

Regrid: Spherical Harmonics (Scalars) g2gsh/g2fsh/f2gsh/f2fsh_Wrap

- **global** rectilinear
- no missing values allowed
- use caution with bounded data; RH (0-100) , q (0..)
  - may 'over-shoot' bound; reset to low or upper bound
- triangular truncation
- **_Wrap** preserve attributes; create coordinate var

```
f      = addfile ("T2m.nc", "r")
T256   = f->T                  ; (time,256,512)
Tg     = g2gsh_Wrap (T256, (/64,128/), trunc)   ; trunc=42
Tf25   = g2fsh_Wrap  (T256, (/73,144/) )

Ta     = f2fsh_Wrap(Tf25, (/50,100/))
Tb     = f2gsh_Wrap(Tf25, (/64,128/), trunc)
```

Regrid: Spherical Harmonics (Vectors) g2gshv/g2fshv/f2gshv/f2fshv_Wrap

- **global** rectilinear
- no missing values allowed
- triangular truncation
- **procedures** (not functions; historical reasons)
- **_Wrap** preserve attributes; create coordinate var

```
f       = addfile ("CESM_gau.nc", "r")
u       = f->U
v       = f->V
uNew = new ( (/nt,jlat,ilon/), typeof (u) )
vNew = new ( (/nt,jlat,ilon/), typeof (v) )
g2gshv_Wrap (u,v, uNew,vNew, trunc)
```

Regrid: Rectilinear -> Simple Curvilinear

- **rgrid2rcm**: rectilinear -> simple curvilinear
- brute force search algorithm; not particularly fast
- bilinear interpolation
- missing values allowed but not filled in
- **_Wrap** preserve attributes; create coordinate var

```
f      = addfile ("curvilinear_file.nc", "r")    ; destination grid
lat2d  = f->xlat     ; lat2d[*][*] , (nlat,mlon)
lon2d = f->xlon     ; lon2d[*][*] , (nlat,mlon)

frl      = addfile ("rectilinear_file.nc", "r")    ; source grid
x        = frl->X     ; x(…,lat,lon), x&lat, x&lon

xgrd  = rgrid2rcm_Wrap (x&lat, x&lon, x, lat2d, lon2d, 0)
```

Regrid: Simple Curvilinear -> Rectilinear

- **rcm2rgrid**: simple curvilinear -> rectilinear
- brute force search algorithm; not particularly fast
- bilinear interpolation
- missing values allowed but not filled in
- **_Wrap** preserve attributes; create coordinate var

```
f       = addfile ("curvilinear_file.nc", "r")    ; source grid
lat2d   = f->xlat     ; lat2d[*][*] , (nlat,mlon)
lon2d   = f->xlon     ; lon2d[*][*] , (nlat,mlon)
z       = f->Z        ; z(…,nlat,mlon)

frl     = addfile ("rectilinear_file.nc", "r")    ; destination grid
lat     = frl->lat
lon     = frl->lon

zgrd    = rcm2rgrid_Wrap (lat2d, lon2d, z, lat, lon, 0)
```

Regrid: NCL-ESMF

- Integrated in conjunction with NOAA Cooperative Institute for Research in Environmental Sciences
- Available since NCL V6.1.0 (May 2012)
- Works with **rectilinear, curvilinear, unstructured** grids
- Multiple interpolation methods available
  - Bilinear
  - Conservative
  - Patch
  - Nearest neighbor
- Can handle masked points
- Better treatment for values at poles
- Works on global/regional grids
- Satellite swath, random
- Can run in parallel or single-threaded mode

Regrid: NCL-ESMF

- Most general & highest quality regridding

- **Functions**: http://www.ncl.ucar.edu/Document/Functions/ESMF.shtml
- **Examples**: https://www.ncl.ucar.edu/Applications/regrid.shtml

- **Basic Steps:**
  - Reading or generating the "source" grid.
  - Reading or generating the "destination" grid.
  - Creating NetCDF files that describe these two grids (auto)
  - ***Generating a NetCDF file that contains the weights***
    - **Weight file can be reused/shared**
  - Applying weights to data on the source grid, to interpolate the data to the destination grid (simple function; very fast).
  - Copying over any metadata to the newly regridded data.

Regrid: NCL-ESMF: Methods

- "bilinear" - the algorithm used by this application to generate the bilinear weights is the standard one found in many textbooks. Each destination point is mapped to a location in the source mesh, the position of the destination point relative to the source points surrounding it is used to calculate the interpolation weights.
- "patch" - this method is the ESMF version of a technique called "patch recovery" commonly used in finite element modeling***. It typically results in better approximations to values and derivatives when compared to bilinear interpolation.***
- "conserve" - this method will typically have a larger interpolation error than the previous two methods, but will do a much better job of preserving the value of the integral of data between the source and destination grid.
- "neareststod" - *Available in* ***version 6.2.0*** *and later.* The nearest neighbor methods work by associating a point in one set with the closest point in another set.

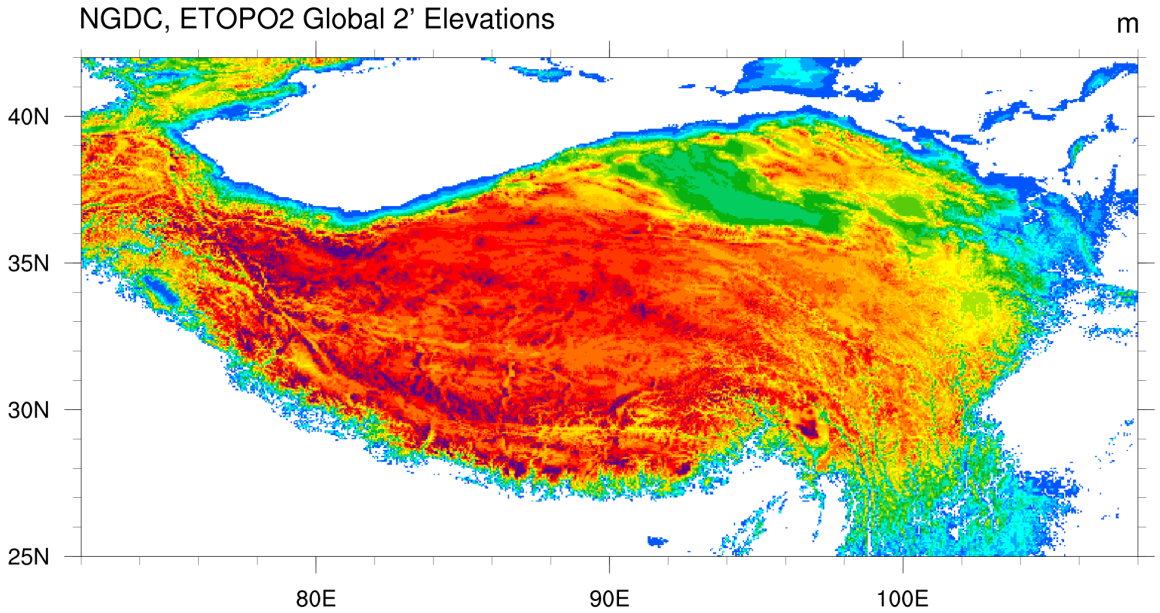
Sample ESMF Code: Curv ->Rect (1)

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/esmf/ESMF_regridding.ncl"

  InterpMethod = "bilinear"          ; "bilinear", "conserve", "patch"
  srcFileName  = "merged_AWIP32.1979010100.3D.NARR.grb"

  sfile        =  addfile(srcFileName,"r")          ; SOURCE
  x            = sfile->FOO                 ; (nlat,mlon)
  lat2d        = sfile->gridlat             ; (nlat,mlon)
  lon2d        = sfile->gridlon
  nmsg         = num(ismissing(x))      ; # of msg values

  x@lat2d      = lat2d                      ; These attributes will be used by
  x@lon2d      = lon2d                      ; ESMF_regrid for the source grid

;---Create the DESTINATION rectilinear lat[*]/lon[*] arrays.
  lat   = fspan(   1.0, 85.0 ,337)          ; nlat=337
  lon   = fspan( 150.0,358.5 ,831)          ; nlon-831
```

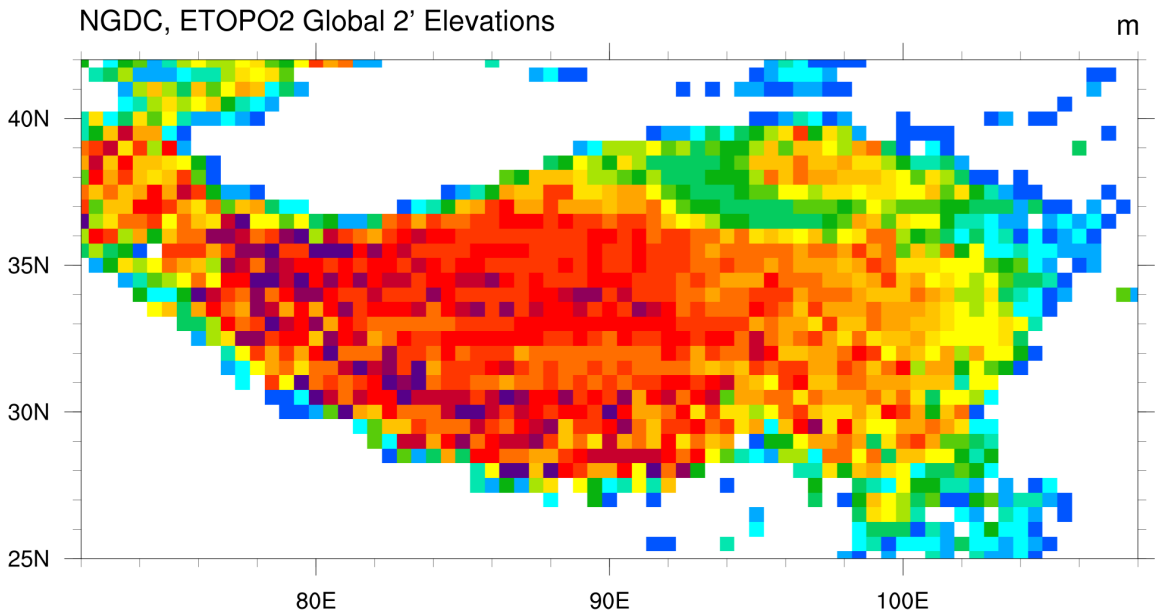
Sample ESMF Code: Curv ->Rect (2)

```
;---Create regrid options
    Opt                     = True
    Opt@InterpMethod    = InterpMethod
    Opt@WgtFileName    = "NARR_to_Rect.WgtFile_"+InterpMethod+".nc”
    if (nmsg.gt.0) then
       Opt@SrcMask2D      = where(ismissing(x),0,1)
    end if
    Opt@SrcRegional     = True
    Opt@DstGridType     = "rectilinear"
    Opt@DstGridLat      = lat
    Opt@DstGridLon      = lon
    Opt@DstRegional     = True

    Opt@ForceOverwrite = True           ; my personal favorites
    Opt@RemoveSrcFile  = True           ; remove grid description files
    Opt@RemoveDstFile  = True
    Opt@NoPETLog       = True           ; 6.2.1 onward
    Opt@Debug          = True

;---Perform the regrid: NARR ==> rectilinear (_reclin)
   x_reclin = ESMF_regrid(x, Opt)
```

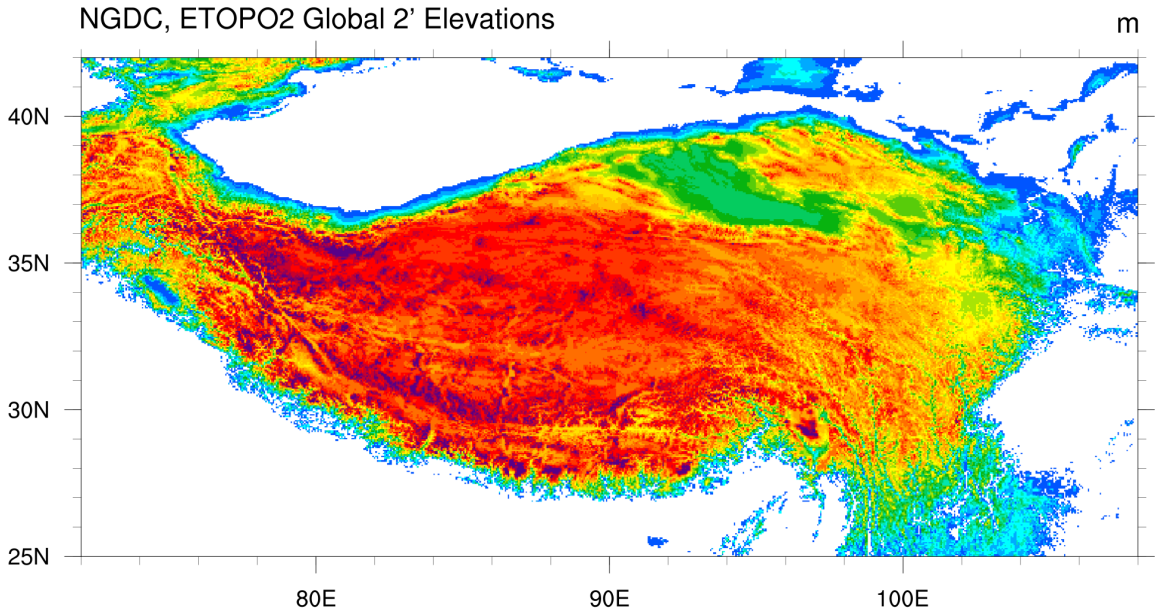
TOPO: Original data 511 x 1081



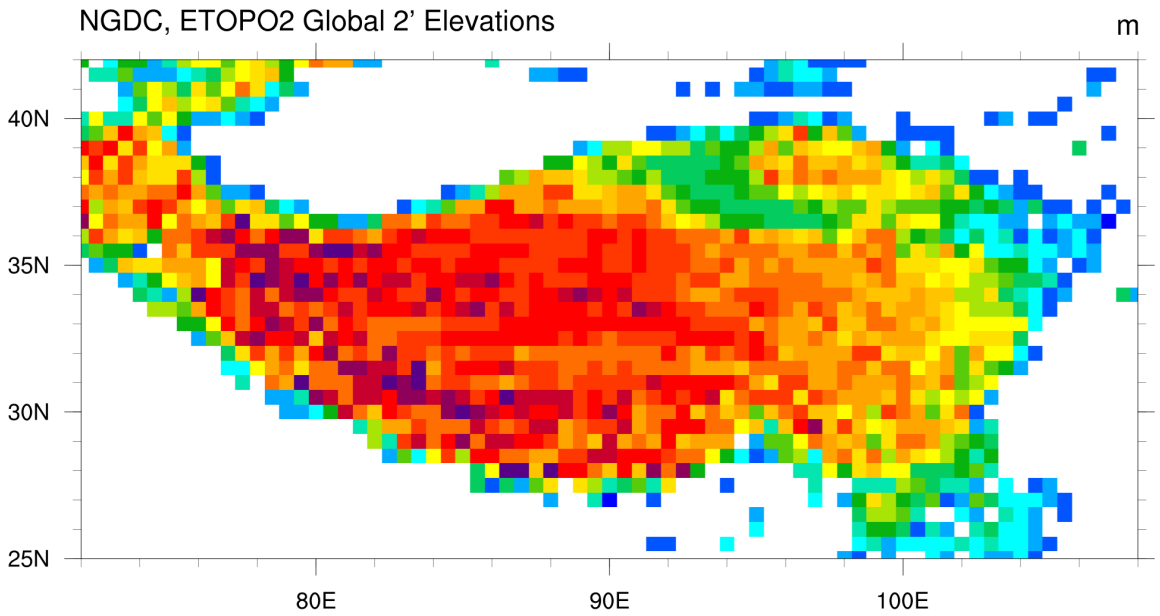
TOPO: Regridded to 0.5 degree 35 x 73 (bilinear)

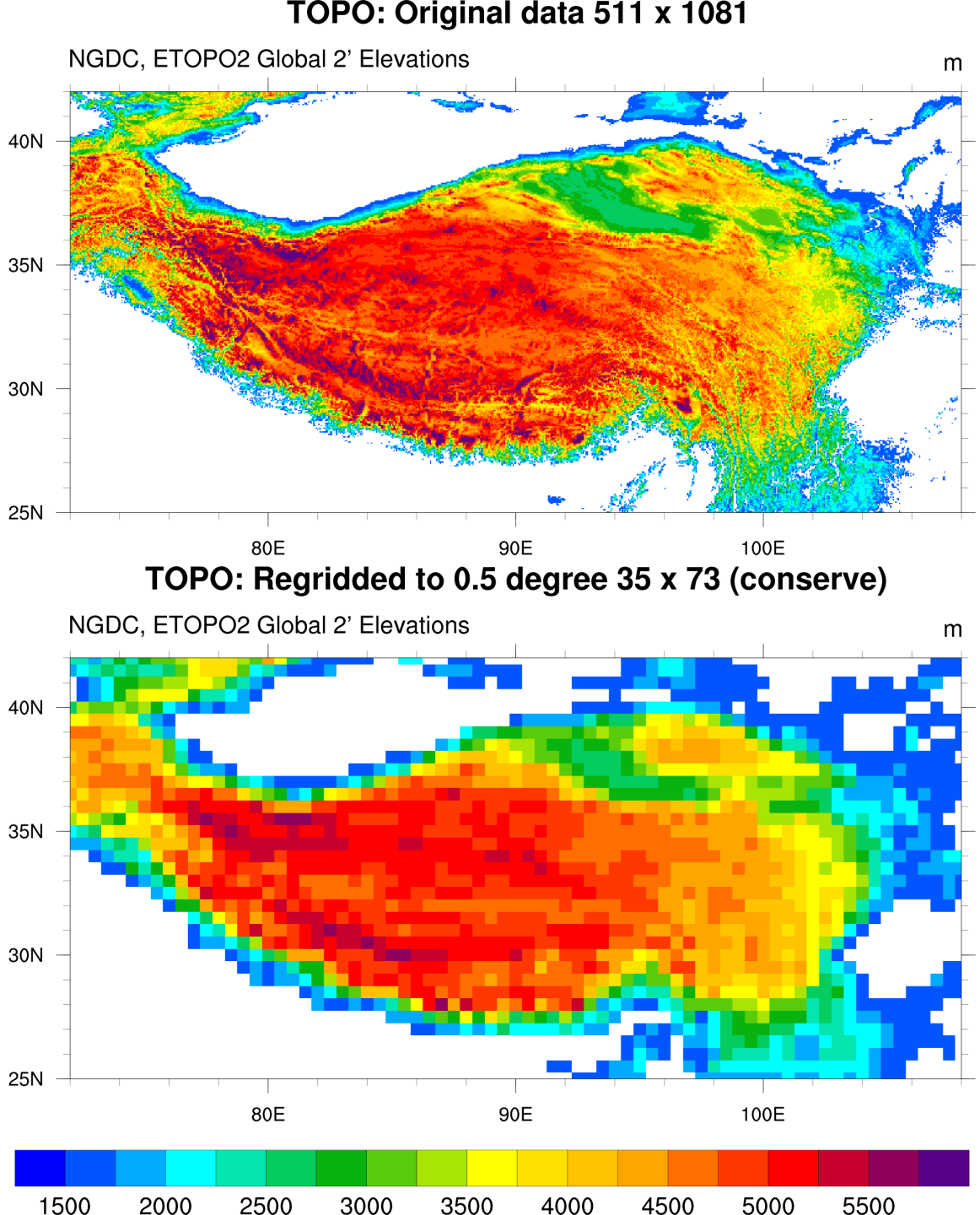


TOPO: Original data 511 x 1081

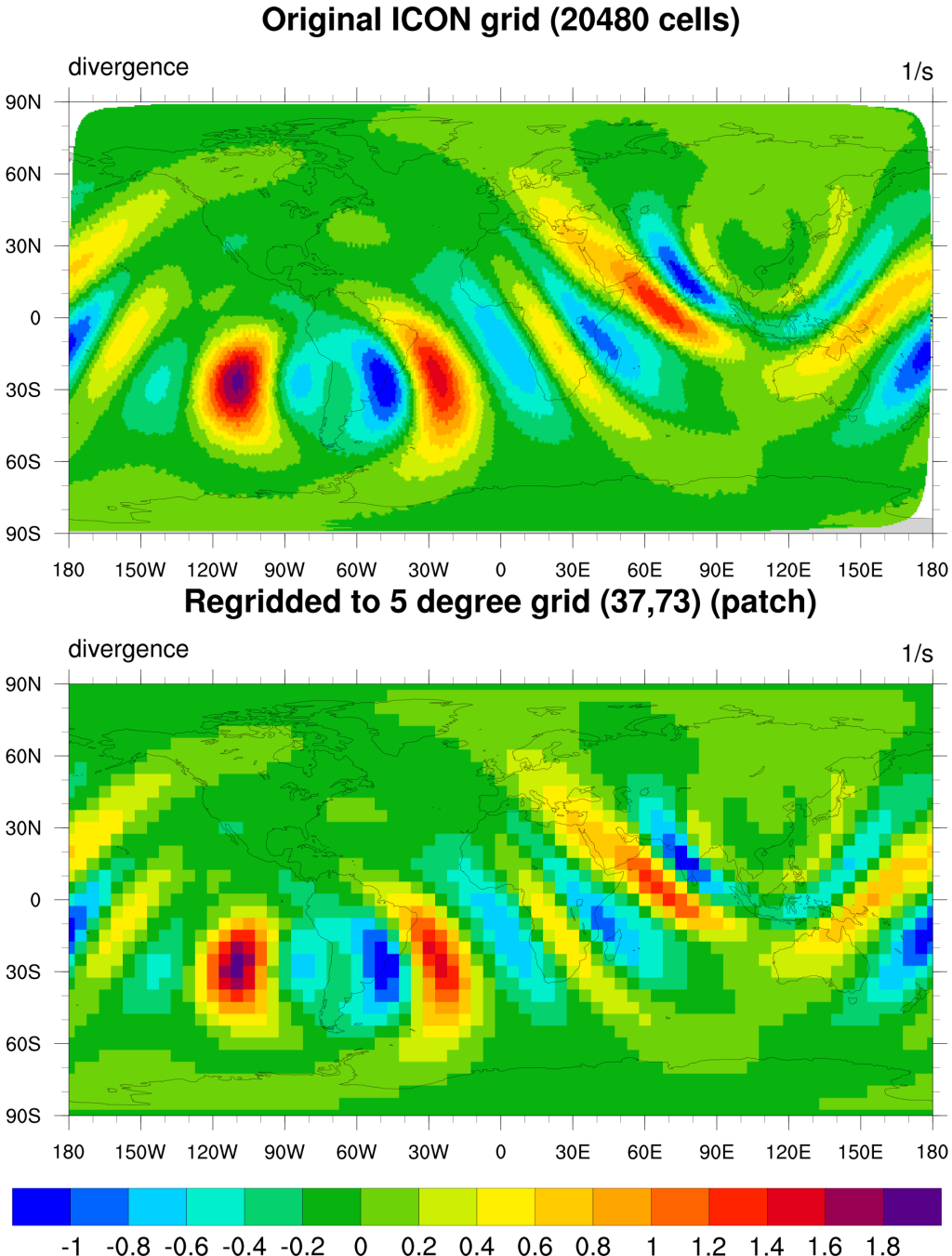


TOPO: Regridded to 0.5 degree 35 x 73 (patch)

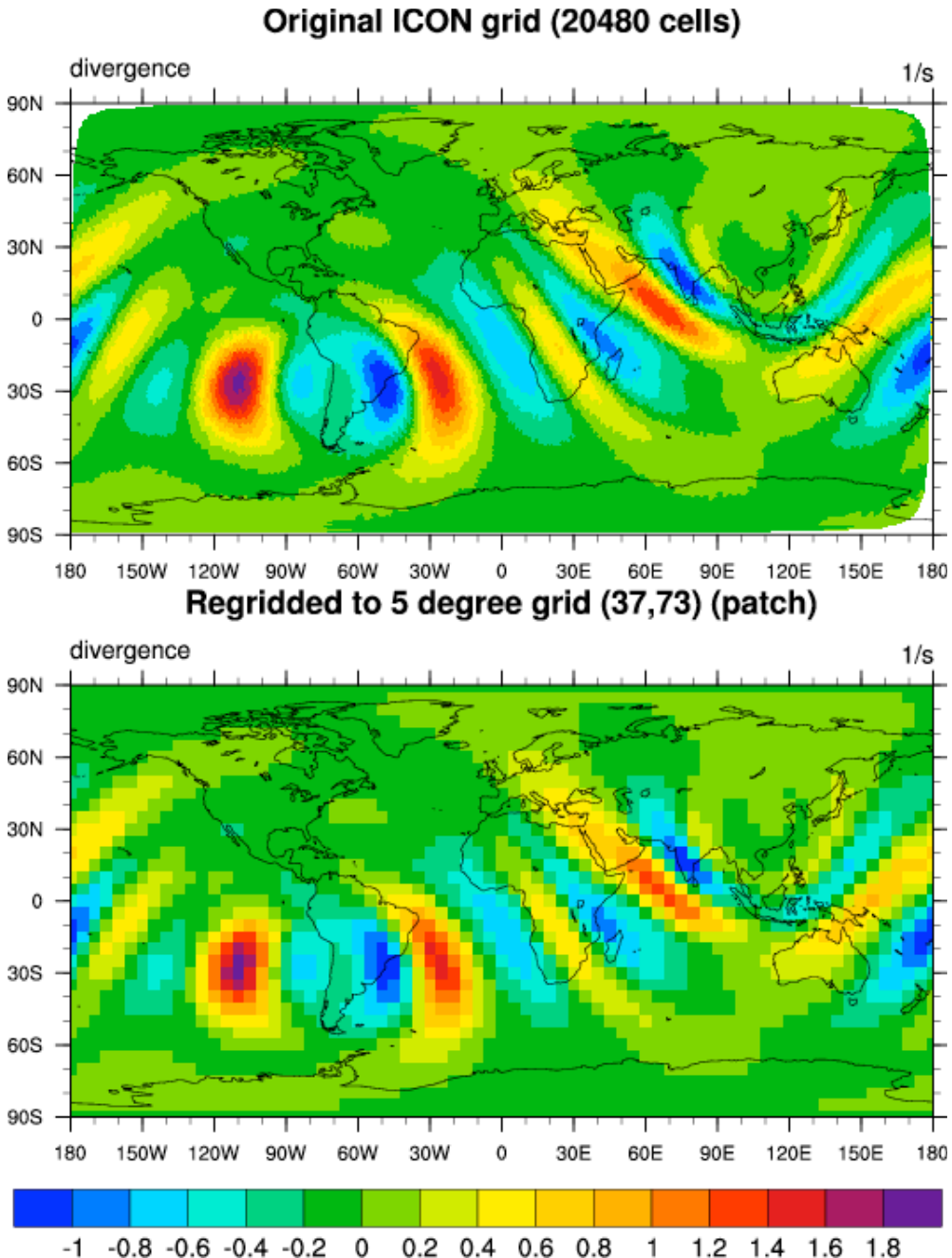



TOPO: Original data 511 x 1081
NGDC, ETOPO2 Global 2' Elevations
m
40N
35N
30N
25N
80E
90E
100E
TOPO: Regridded to 0.5 degree 35 x 73 (conserve)
NGDC, ETOPO2 Global 2' Elevations
m
40N
35N
30N
25N
80E
90E
100E
1500
2000
2500
3000
3500
4000
4500
5000
5500

Regrid ESMF: ICON

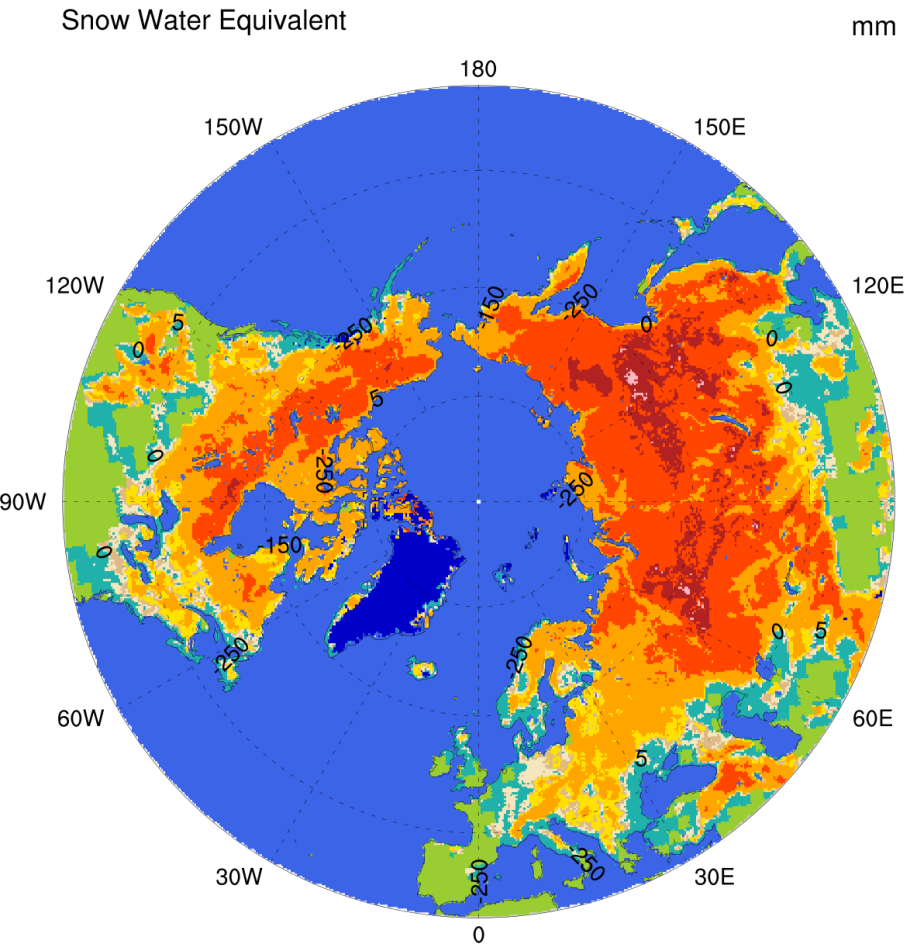


Regrid ESMF: ICON

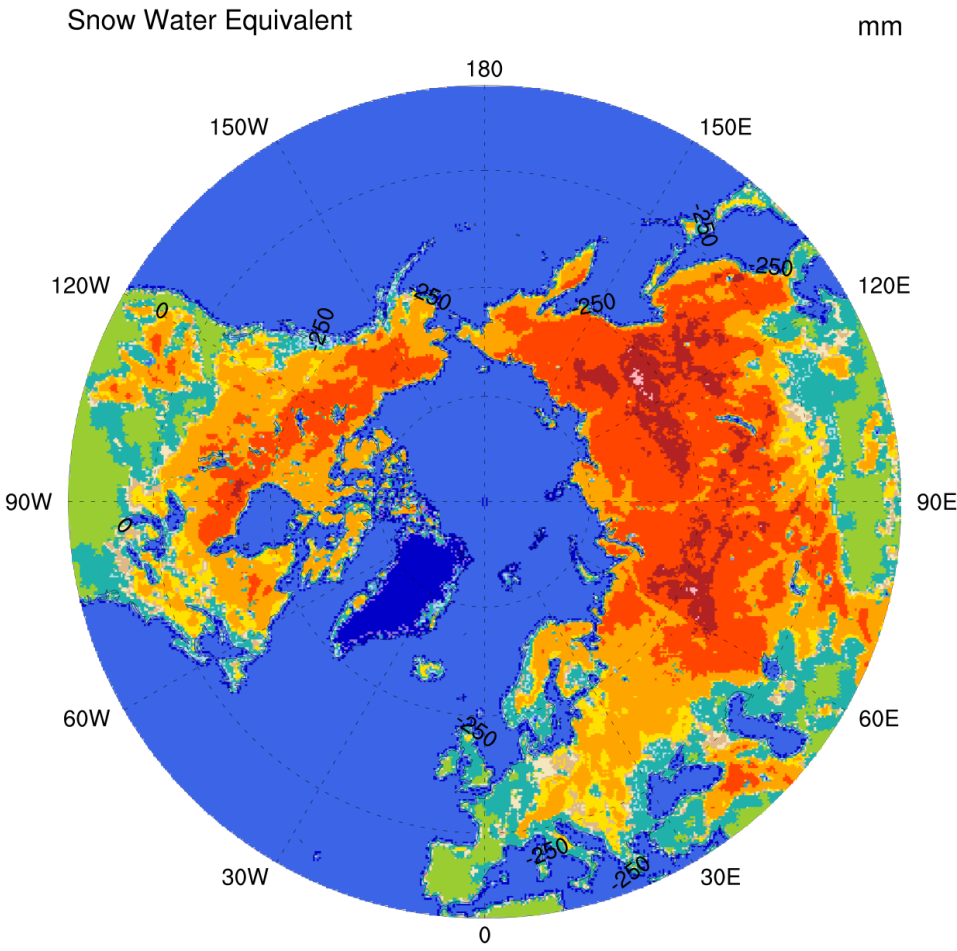


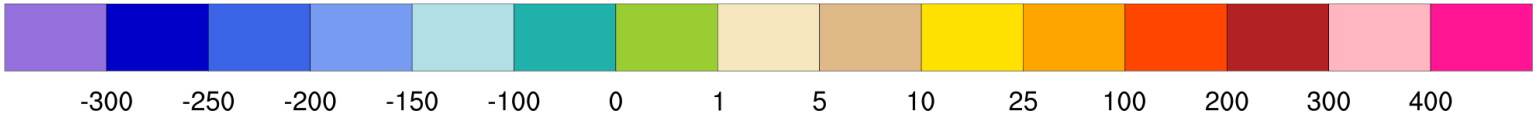
Regrid: ESMF: EASE

Original EASE grid (721,721)

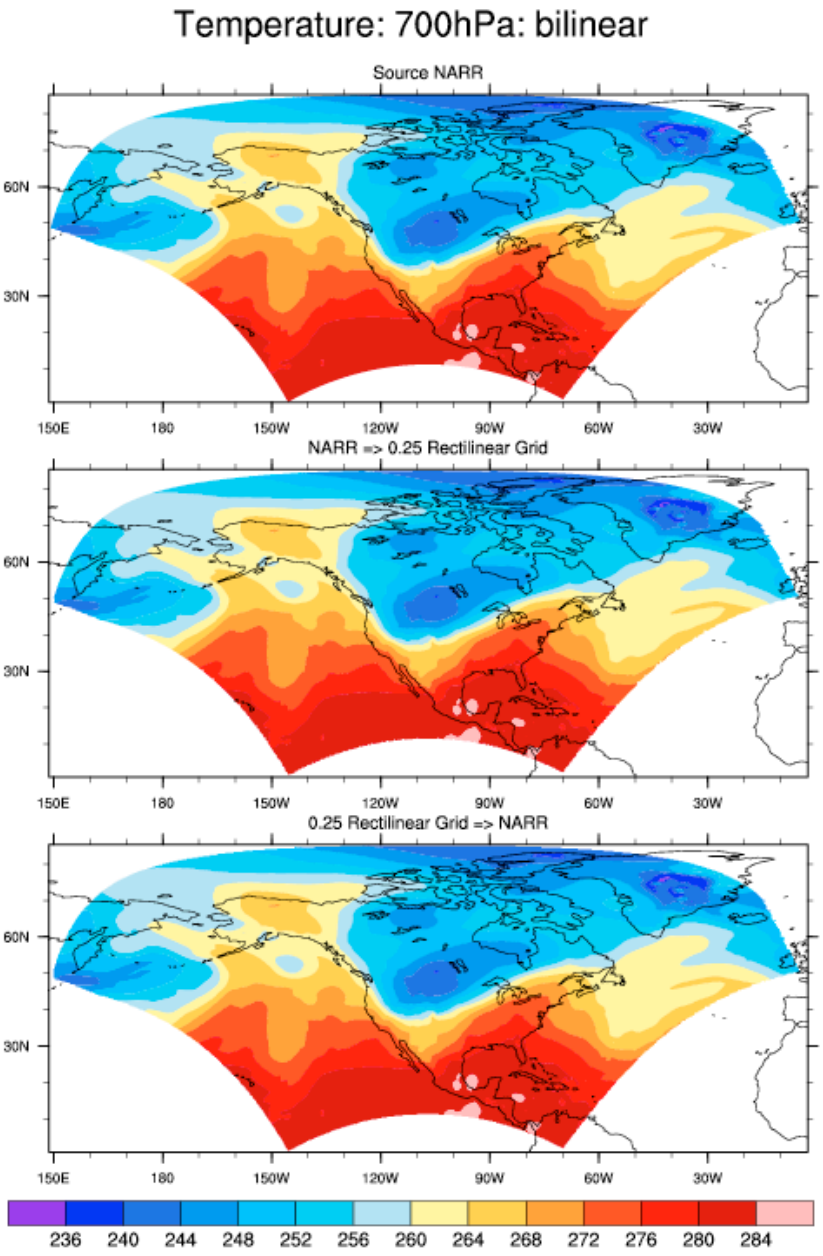


Regridded to 0.25 degree grid (359 x 1439)

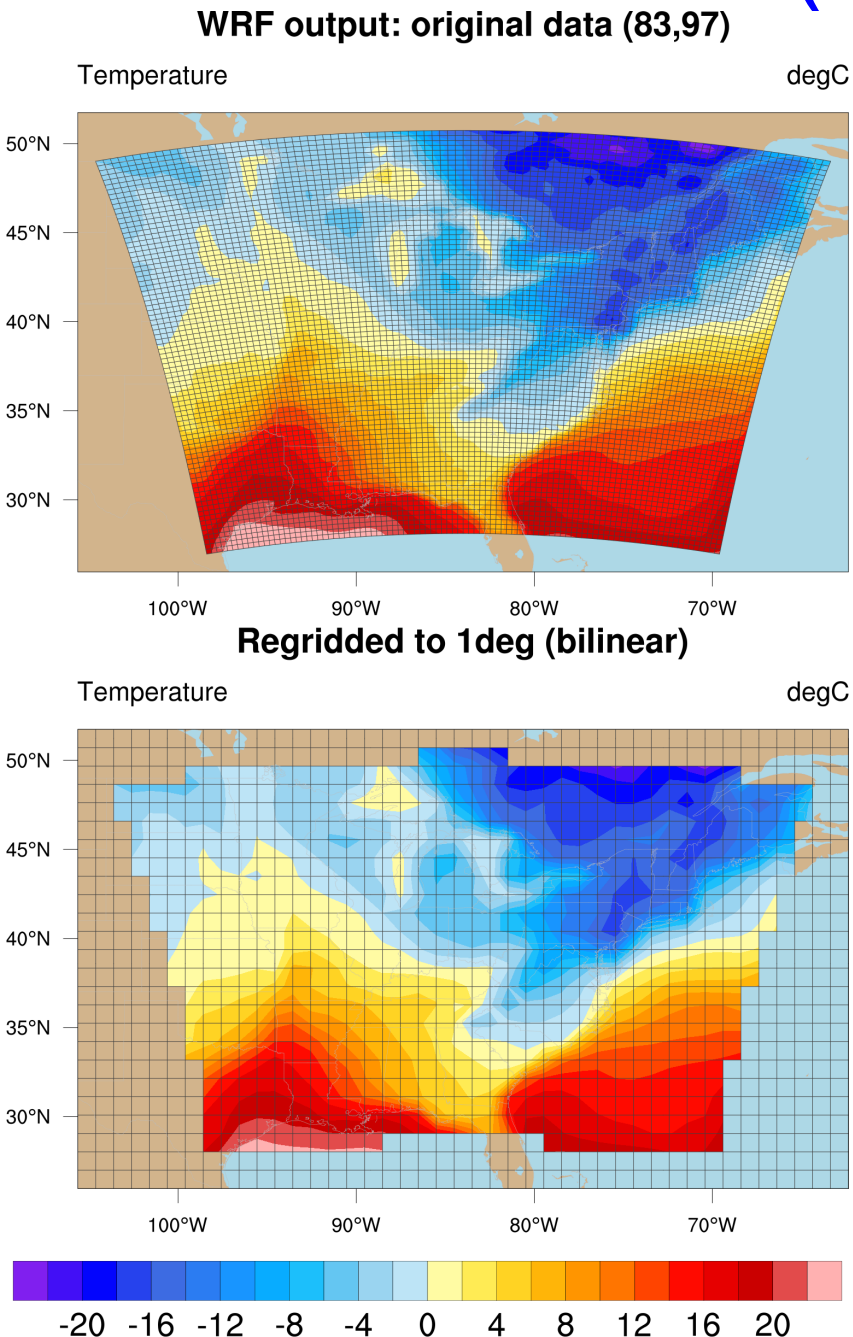




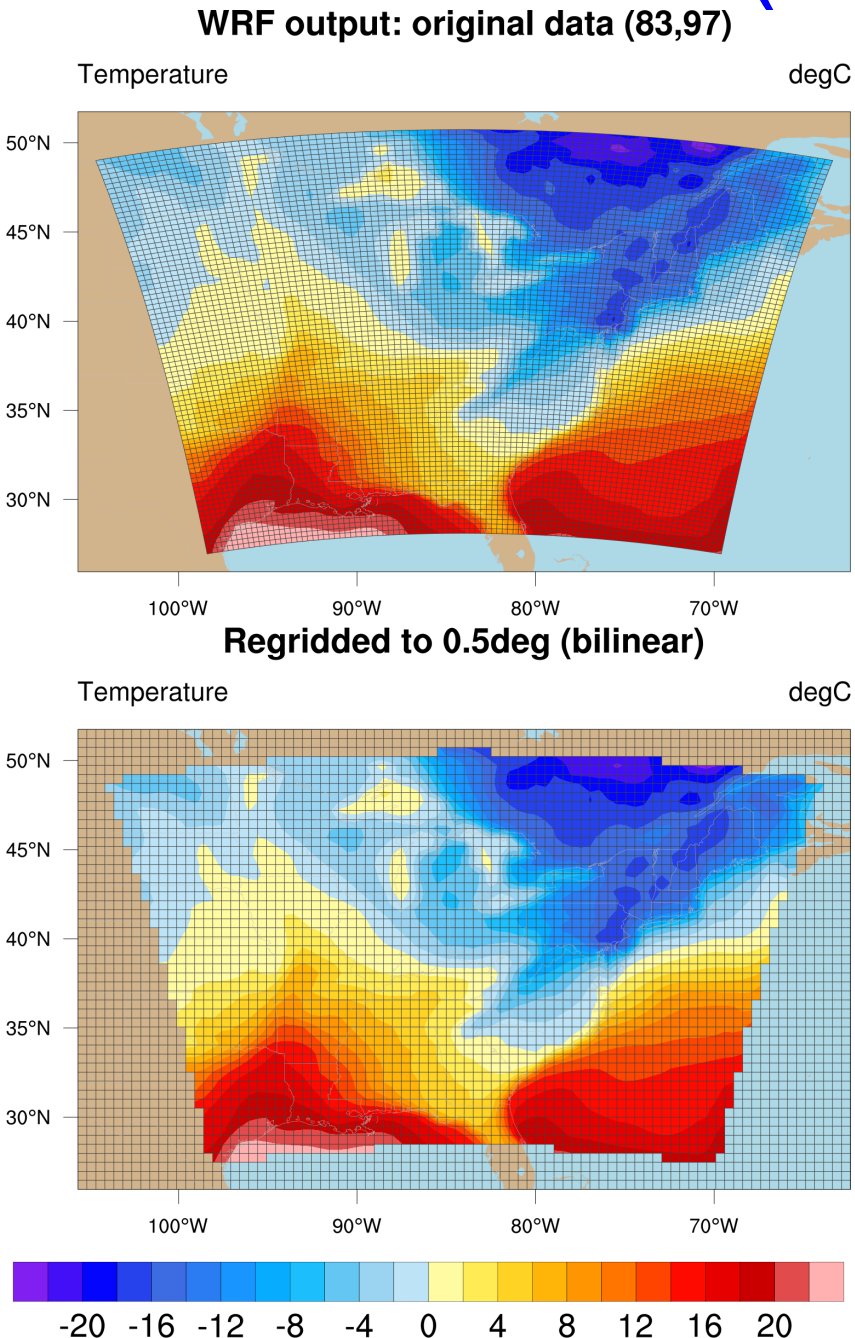
Regrid: ESMF: NARR



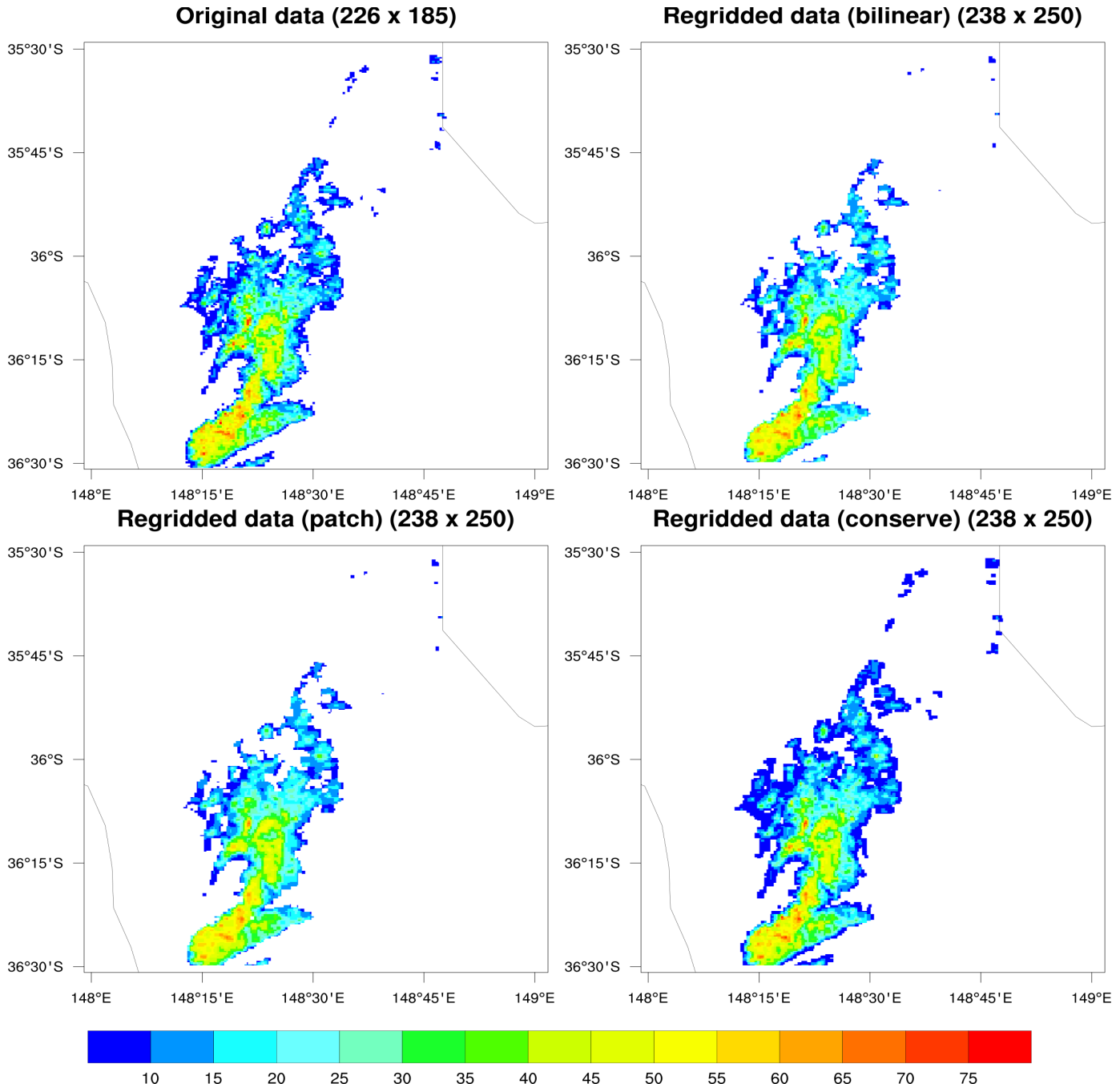
Regrid: ESMF: WRF (1 deg)



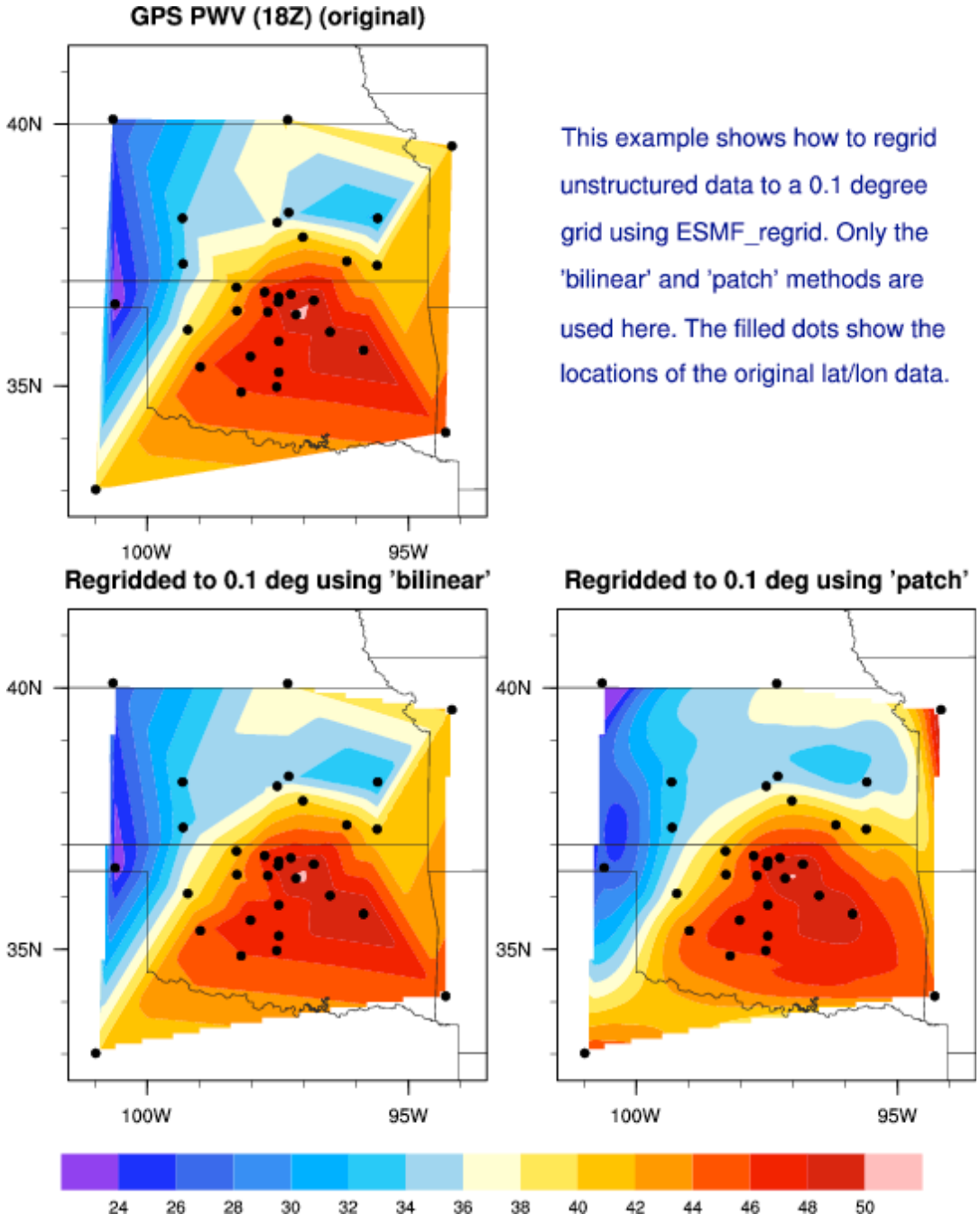
Regrid: ESMF: WRF (0.5 deg)



Regrid ESMF: Swath to WRF Grid: Australia Snow

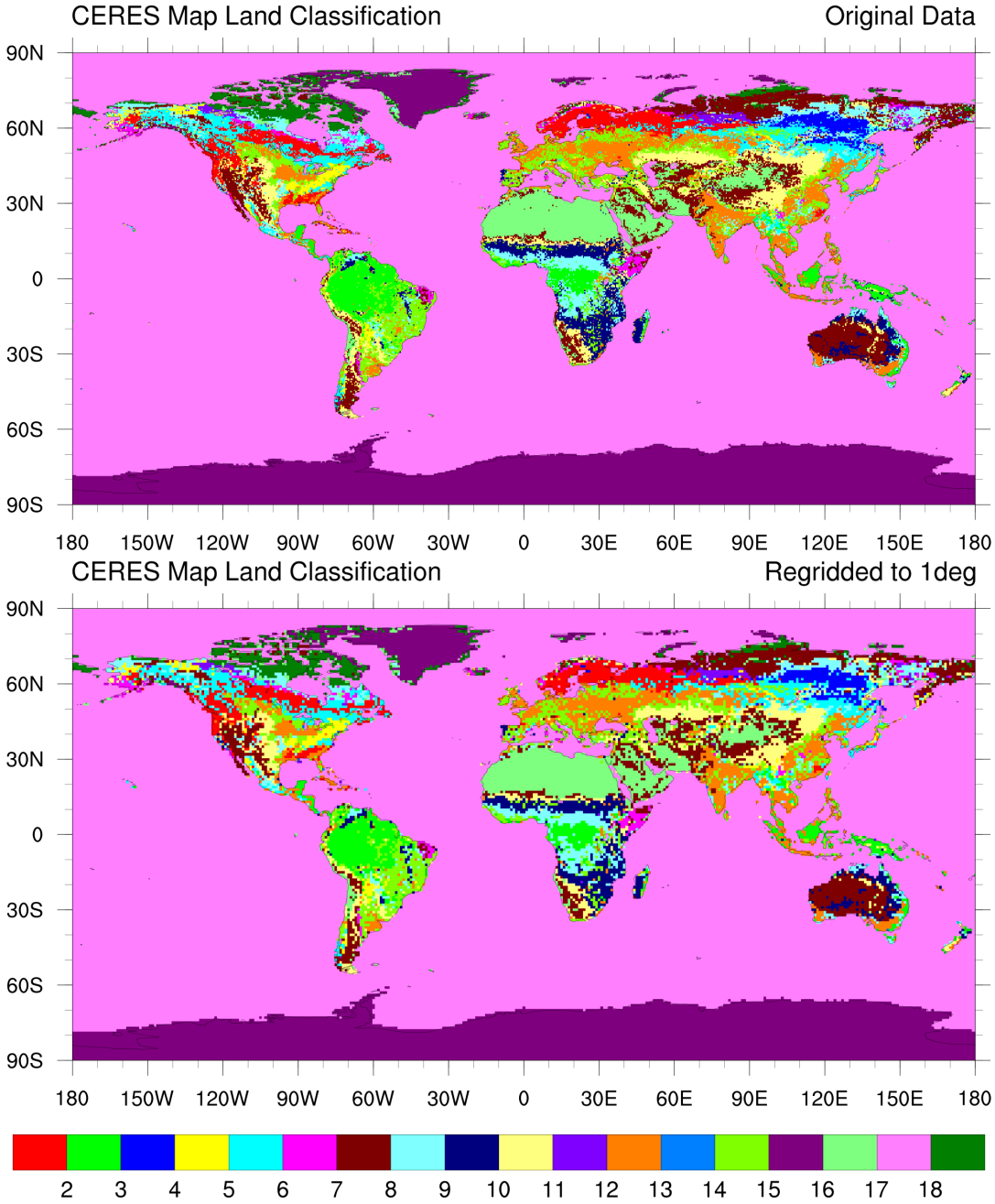


Regrid ESMF: Random to Grid



This example shows how to regrid unstructured data to a 0.1 degree grid using ESMF_regrid. Only the 'bilinear' and 'patch' methods are used here. The filled dots show the locations of the original lat/lon data.

Regrid ESMF: Categorical

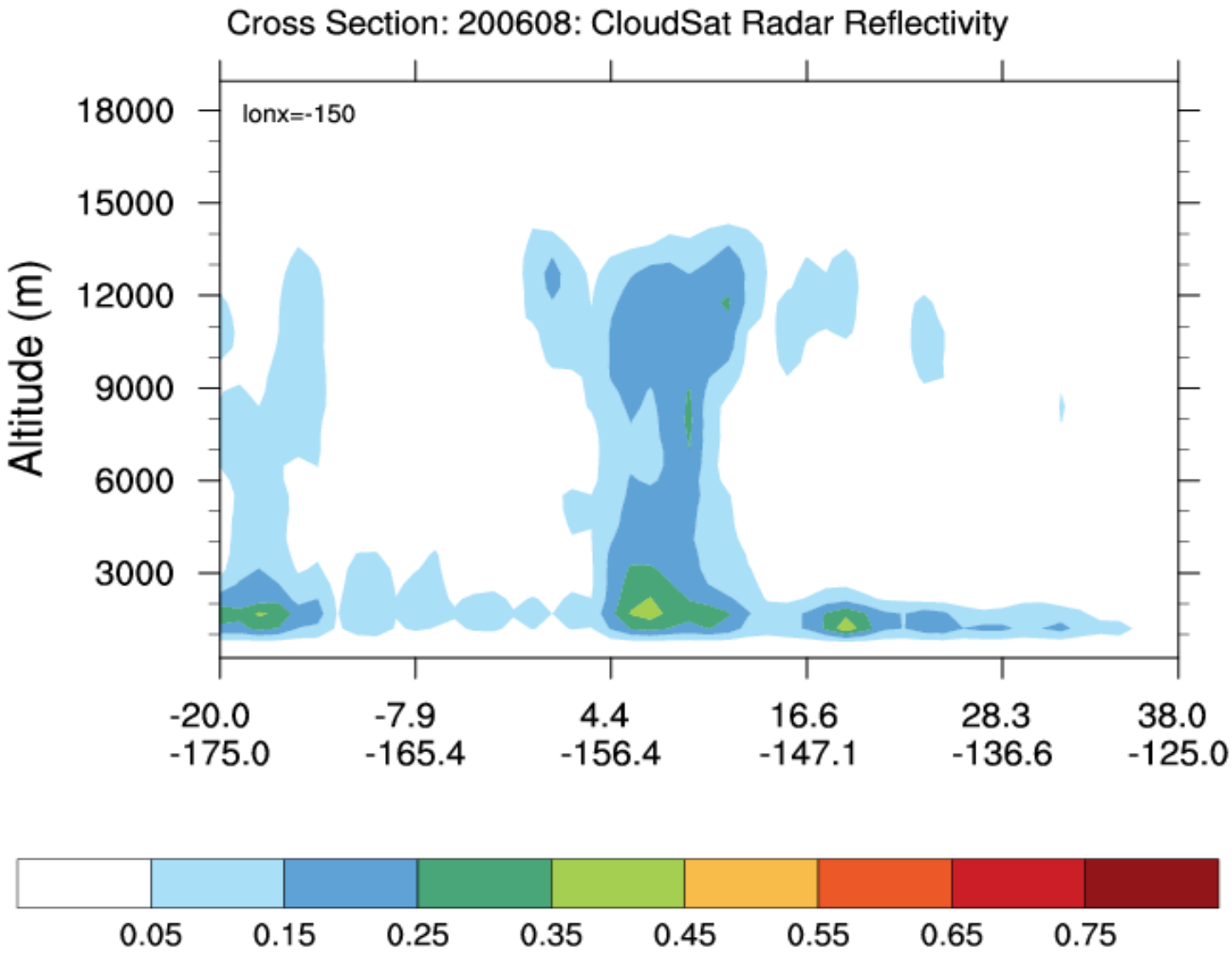


linint2_points_Wrap: Arbitrary Cross-Sec Interpolation

interpolate **rectilinear** grid to arbitrary points

```
diri       = "/Data/Cloud_Sat/"
fili = "cfadDbze94_200606-200612.nc"
f    = addfile(diri+fili ,  "r")
x    = f->cfadDbze94              ; x(time,alt40,lat,lon)
                                    specify lat and lon points
lonx = (/ -175, -165.4, -156.4, -147.1, -136.6, -125.0/)
laty  = (/   -20,     -7.9,       4.4,    16.6,    28.3,    38.0/)
                                   ; interpolate data to given laty/lonx
xsec = linint2_points_Wrap (x&lon, x&lat, x, False, lonx,laty, 0)
        ; [alt40 | 40] x [pts | 6]
```

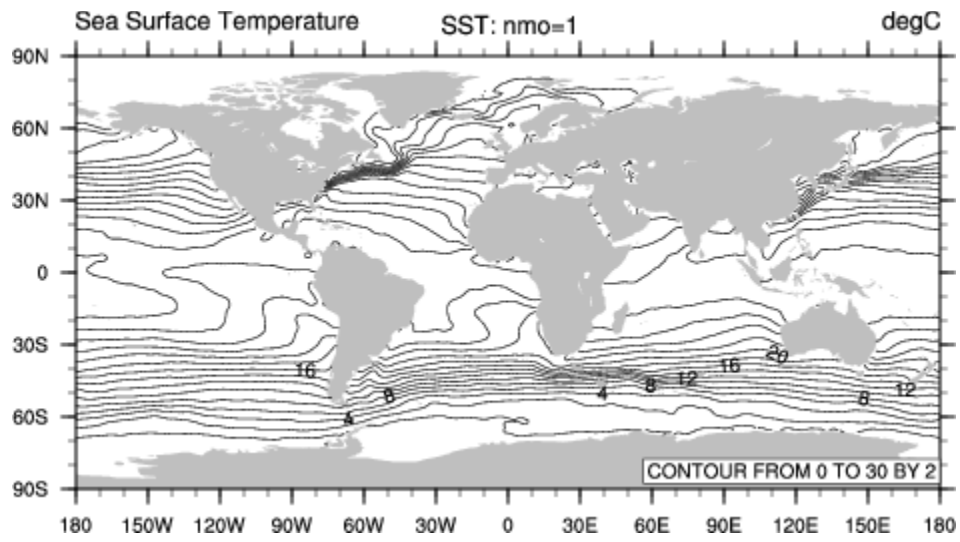
linint2_points: Cross-section

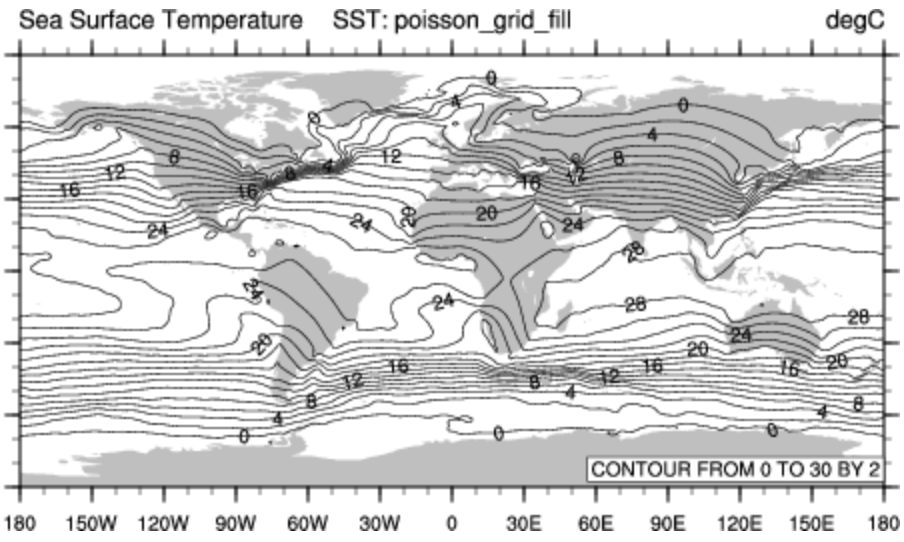


poisson_grid_fill

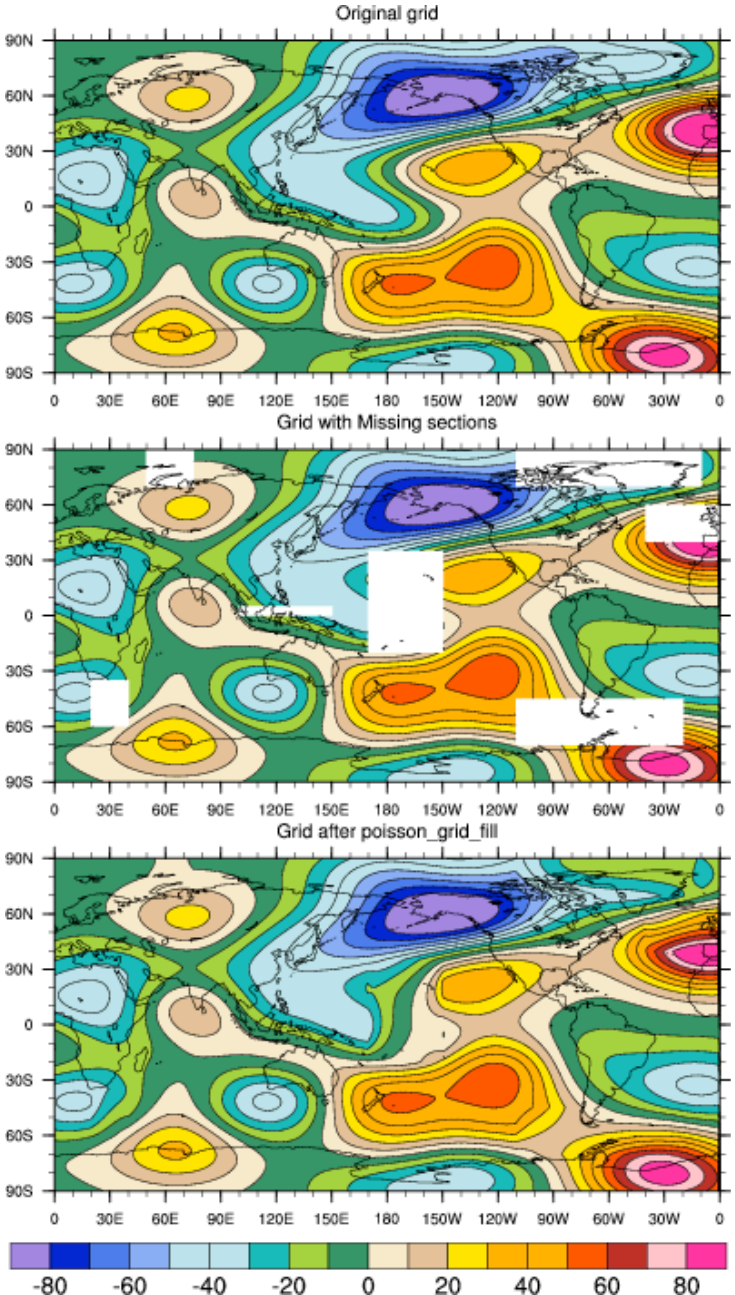
- replaces **all** _FillValue grid ponts
  - Poisson's equation solved via relaxation
  - **original values unchanged**; boundary conditions
  - works on **any** grid with spatial dimensions [*][*]

```
in   = addfile  (Ocean.nc","r")
sst  = in->SST
poisson_grid_fill (sst, True, 1, 1500, 0.02, 0.6, 0)
```



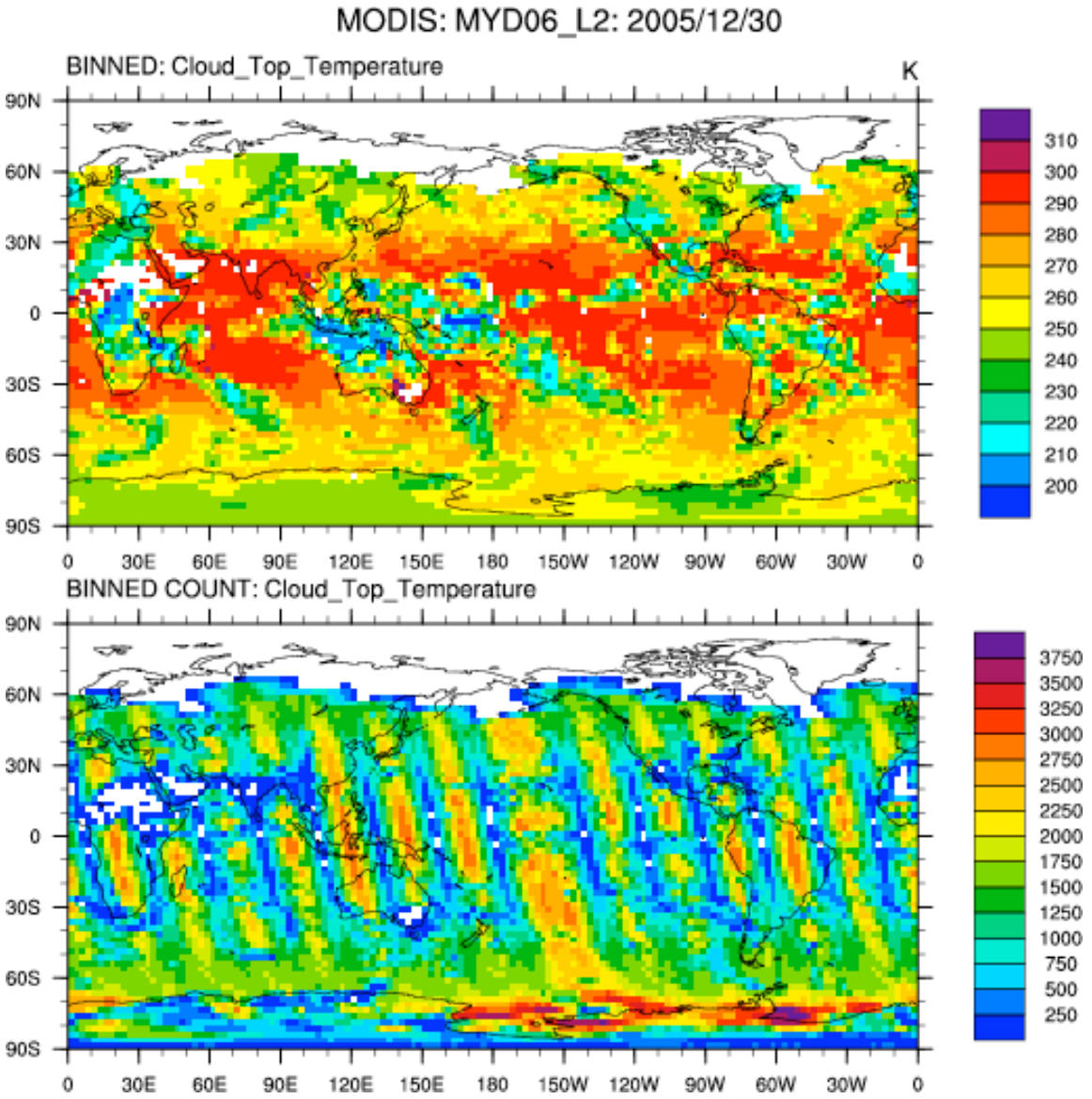


poisson_grid_fill



Regrid: Binning

bin_sum: frequently used with satellite swaths



131

HDF-EOS

files

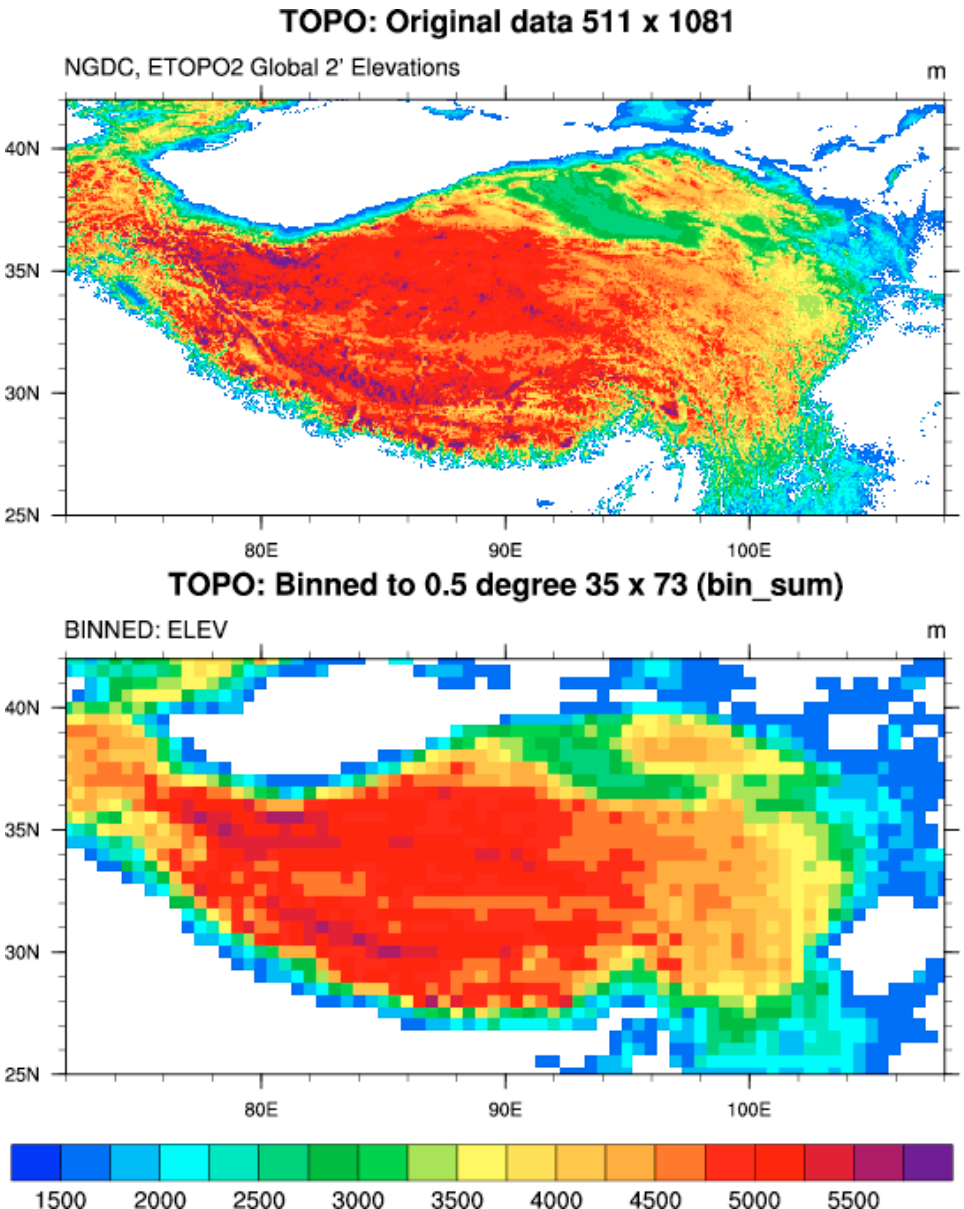
---

swath

data

Regrid: Binning

bin_sum: could be used to regrid (local area avg)



Vertical Interpolation

Functions:

vinth2p, vinth2p_ecmwf: hybrid (sigma) to isobaric levels

int2p_n_Wrap: **any** vertical coordinate to another

Examples:

http://www.ncl.ucar.edu/Applications/vert_interp.shtml

http://www.ncl.ucar.edu/Applications/isent.shtml

Vertical interpolation: POP: int2p_n