Converting the OMWG diagnostic scripts to NCL

Observations and lessons learned

Dave Brown

NCL Development Team

Topics

- Motivation for this project
- Background and current status
- Implementation guidelines
- Comparison of graphical output
- NCL vs. IDL
- Comparison of source code
- Lessons learned

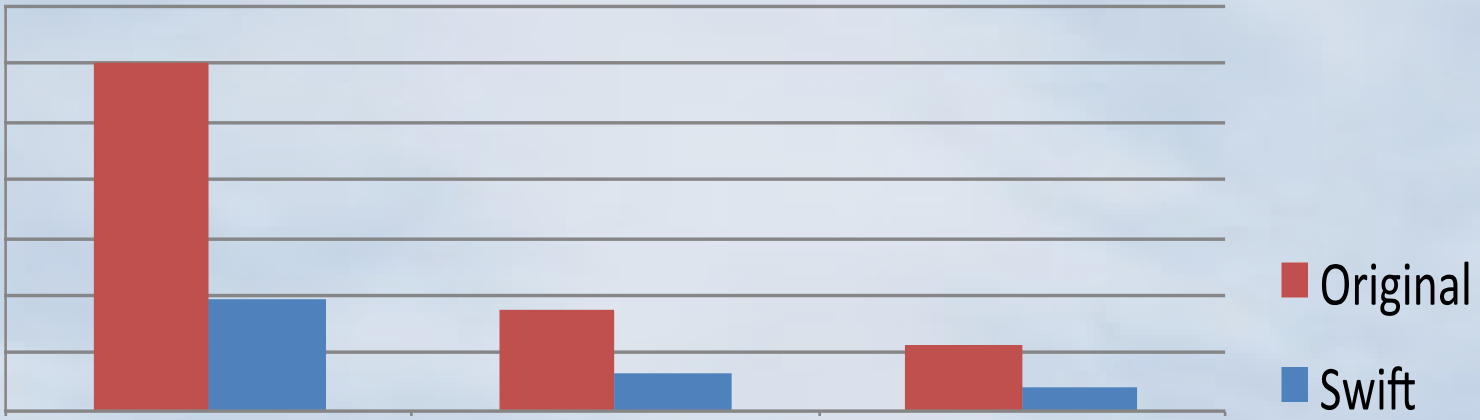
Motivation

- ParVis: provide parallel-processing solutions for the big data problem facing climate researchers
- Led by ANL; collaborators include NCAR, PNNL, SNL and UC-Davis
- Multiple goals including (among others):
  - (long term) ParNCL: a parallel version of NCL
  - (short term) Use SWIFT, a task-parallel scripting tool to improve performance for existing tasks
- The diagnostics make good ParVis case studies
- Immediate benefit: provide non-proprietary open and free code that users can deploy anywhere

AMWG diagnostics status

- C-shell scripts run NCO tools for data reduction and NCL for analysis and viz
- Converted to Swift originally by John Dennis
- Changes to Swift to accommodate the diagnostic package work flow.
- For ParVis, an all-NCL version developed for comparison between the new ParNCL and the existing version using NCO tools (or a parallel-enabled replacement, Pagoda from PNNL)

Various Datasets



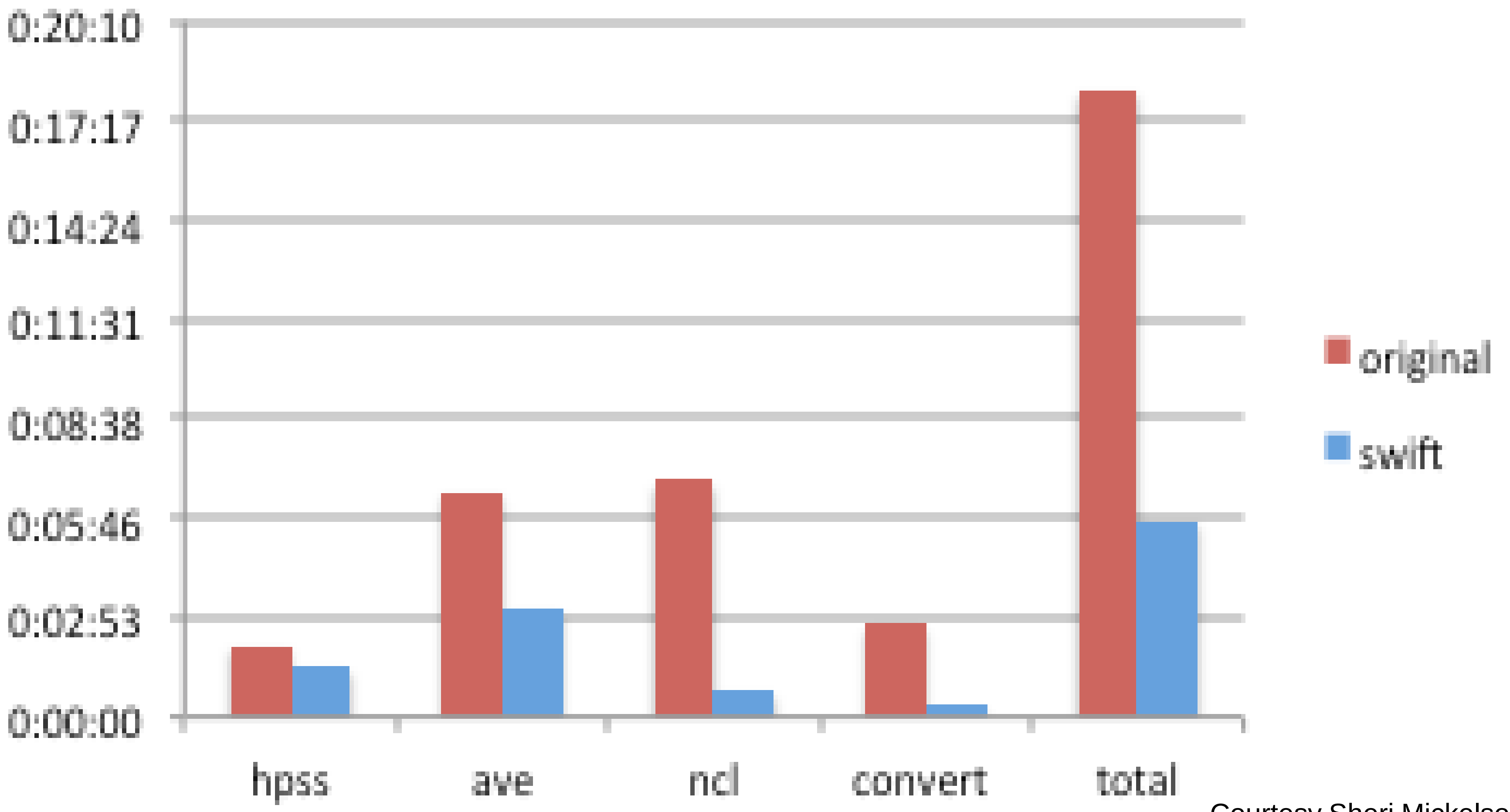
Courtesy Mike Wilde

AMWG Diagnostics

OMWG diagnostics status

- 96% complete – 84 of 87 scripts
  - popdiag and popdiagdiff finished
  - popdiagts: 3 to go
- 2 of 3 Fortran procedures (wrapped as shared objects for now – eventually will become built-in NCL routines)
- Basically transparent to user – scripts work the same as they always have

Preliminary Timings for popdiag.csh



Courtesy Sheri Mickelson

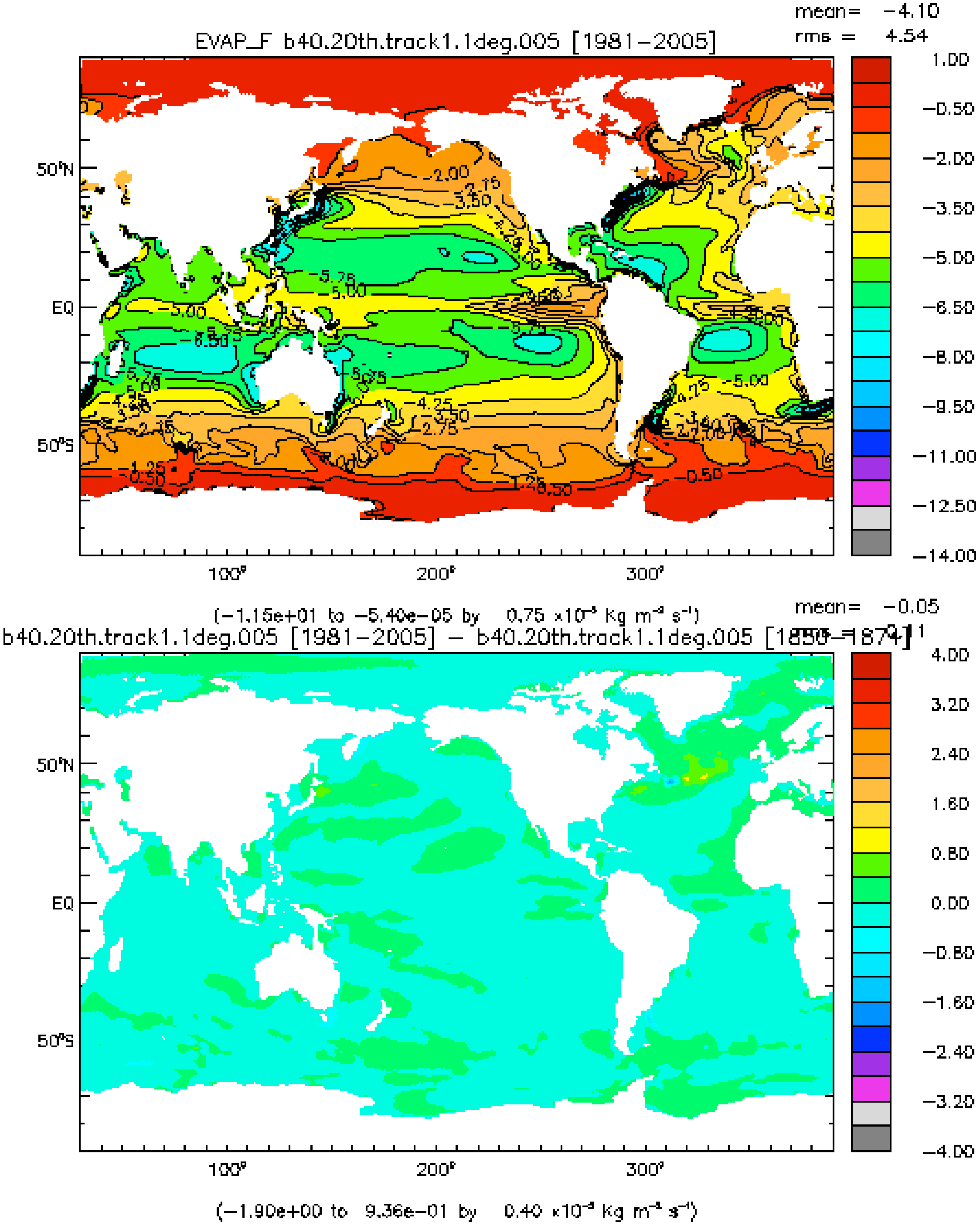
Conversion project guidelines

- Conservative approach
- Results must match mathematically and graphically
- Therefore initial version retains original colormaps, contour levels, and line colors for ease of verification
- Similar positioning of annotations, but font styles, etc. allowed some variation
- Fairly literal translation of code where performance not affected
- Array arithmetic used more aggressively since NCL looping performance is slower

OMWG diagnostic output comparisons

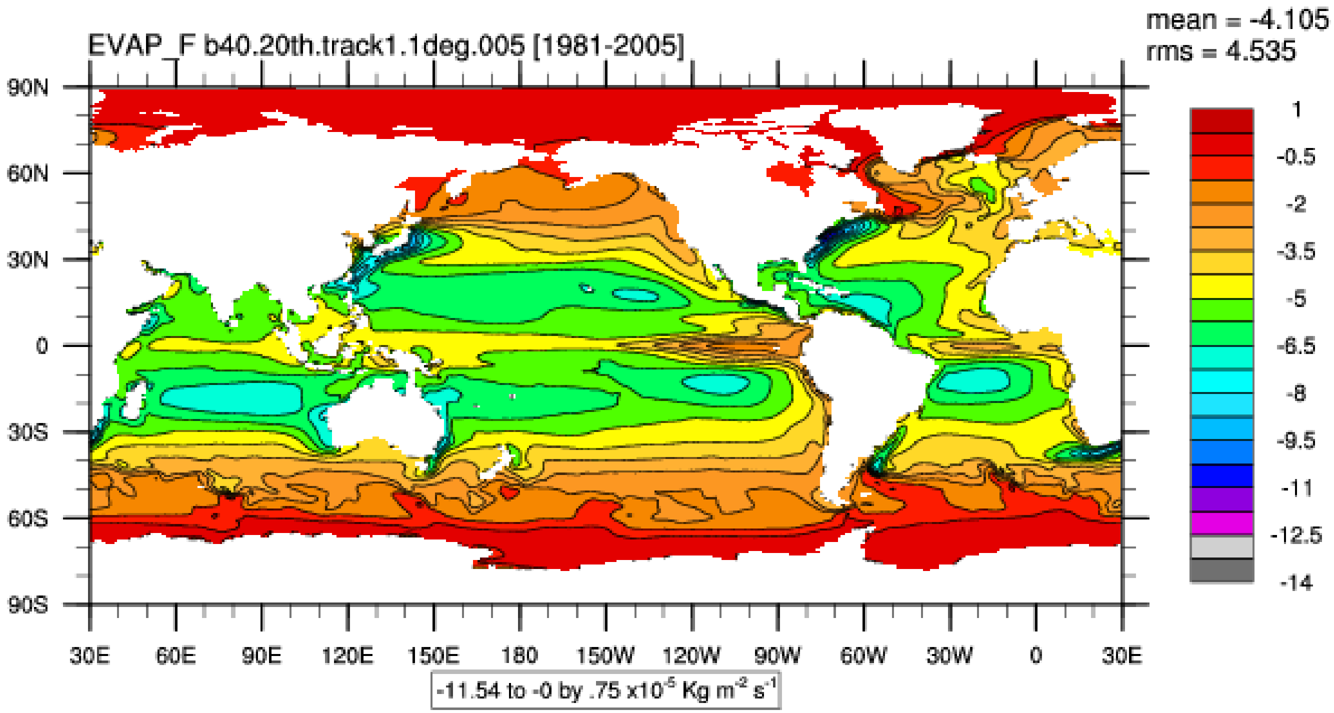
- NCL output online:
  - http://www.ncl.ucar.edu/Applications/popdiag/pd.1981_2005/popdiag.html
- Current IDL output online:
  - http://www.cesm.ucar.edu/experiments/cesm1.0/diagnostics/b40.20th.track1.1deg.005/ocn_1981-2005-obs/popdiag.html

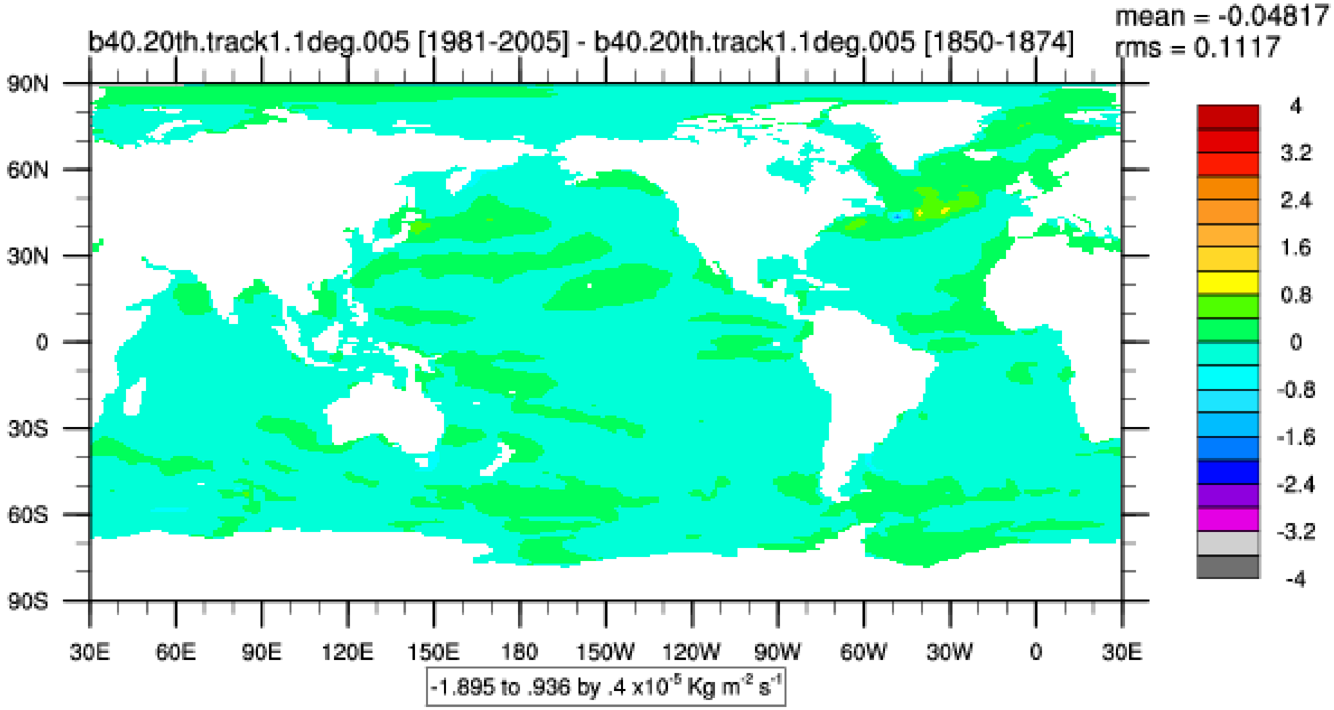
IDL:



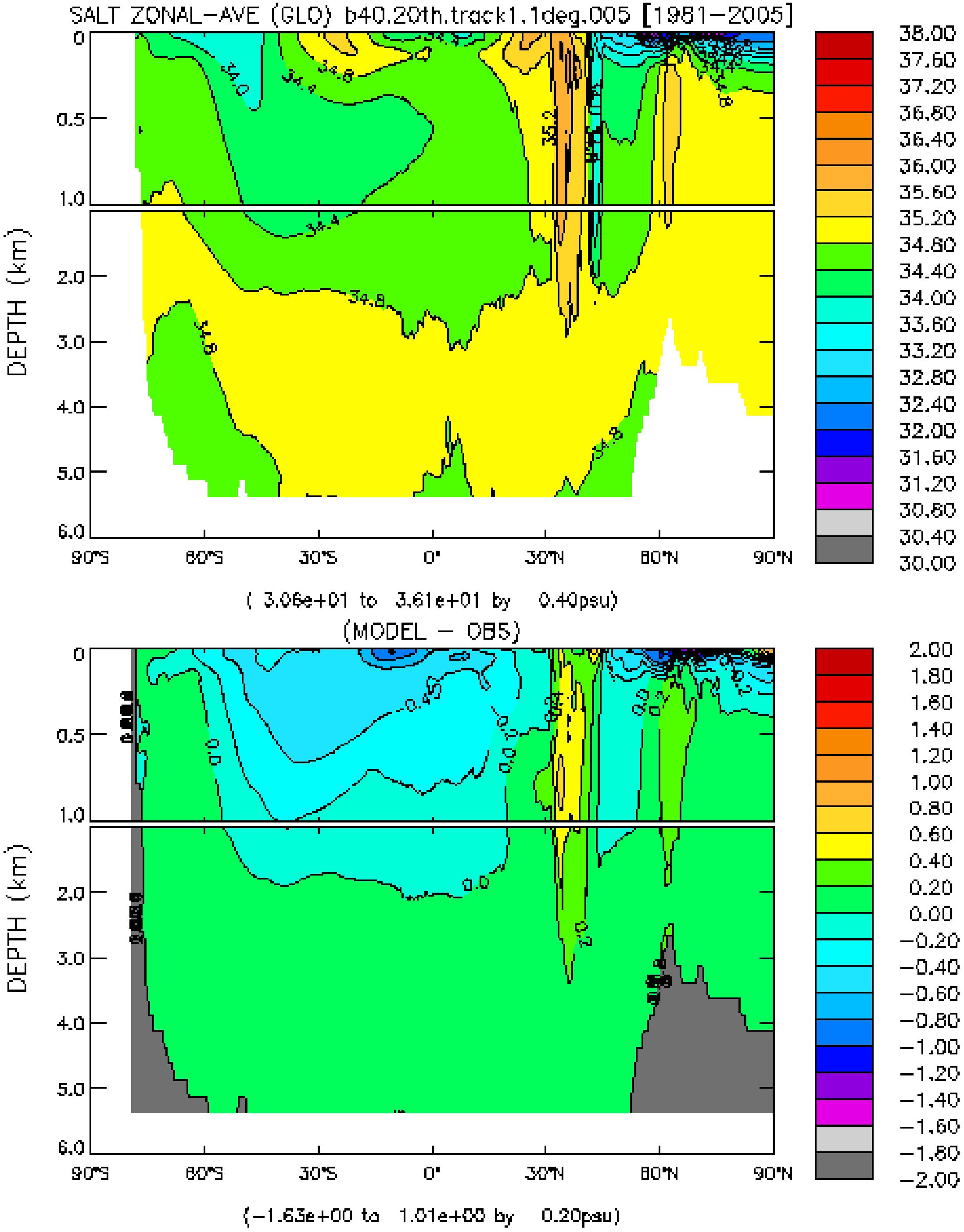
mean= -4.10
rms = 4.54
EVAP_F b40.20th.track1.1deg.005 [1981–2005]
50°N
EQ
50°S
100°
200°
300°
-2.00
2.75
3.50
-5.75
5.00
-5.00
-6.50
-4.25
-0.50
1.25
1.00
-0.50
-2.00
-3.50
-5.00
-6.50
-8.00
-9.50
-11.00
-12.50
-14.00
(-1.15e+01 to -5.40e-05 by 0.75 x10⁻⁵ kg m⁻² s⁻¹)
mean= -0.05
rms = 0.11
b40.20th.track1.1deg.005 [1981–2005] – b40.20th.track1.1deg.005 [1850–1874]
50°N
EQ
50°S
100°
200°
300°
4.00
3.20
2.40
1.60
0.80
0.00
-0.80
-1.60
-2.40
-3.20
-4.00
(-1.90e+00 to 9.36e-01 by 0.40 x10⁻⁵ kg m⁻² s⁻¹)

NCL:

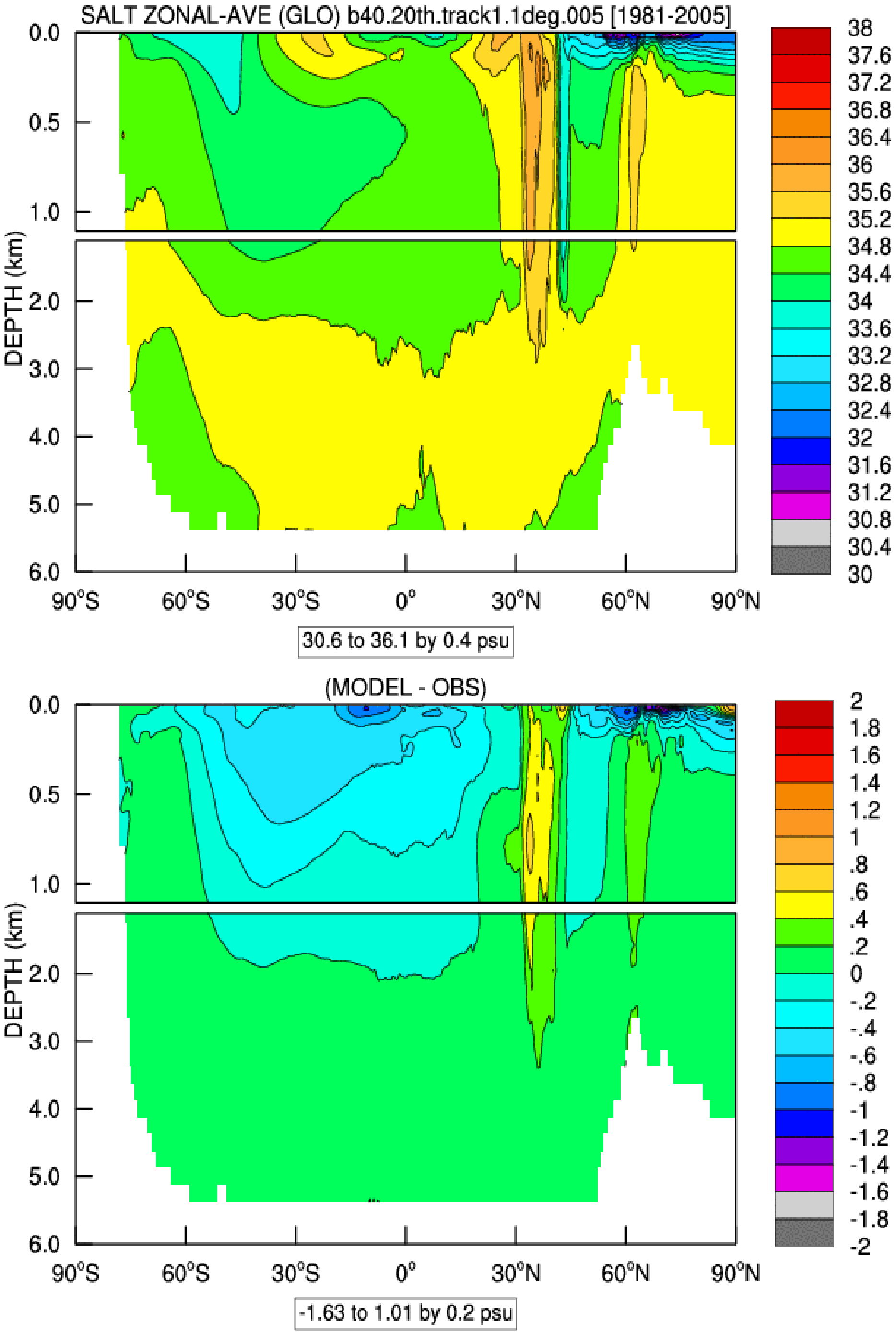

EVAP_F b40.20th.track1.1deg.005 [1981-2005]
mean = -4.105
rms = 4.535
90N
60N
30N
0
30S
60S
90S
30E
60E
90E
120E
150E
180
150W
120W
90W
60W
30W
0
30E
1
-0.5
-2
-3.5
-5
-6.5
-8
-9.5
-11
-12.5
-14
-11.54 to -0 by .75 x10^{-5} Kg m^{-2} s^{-1}


b40.20th.track1.1deg.005 [1981-2005] - b40.20th.track1.1deg.005 [1850-1874]
mean = -0.04817
rms = 0.1117
90N
60N
30N
0
30S
60S
90S
30E
60E
90E
120E
150E
180
150W
120W
90W
60W
30W
0
30E
4
3.2
2.4
1.6
0.8
0
-0.8
-1.6
-2.4
-3.2
-4
-1.895 to .936 by .4 x10^{-5} Kg m^{-2} s^{-1}

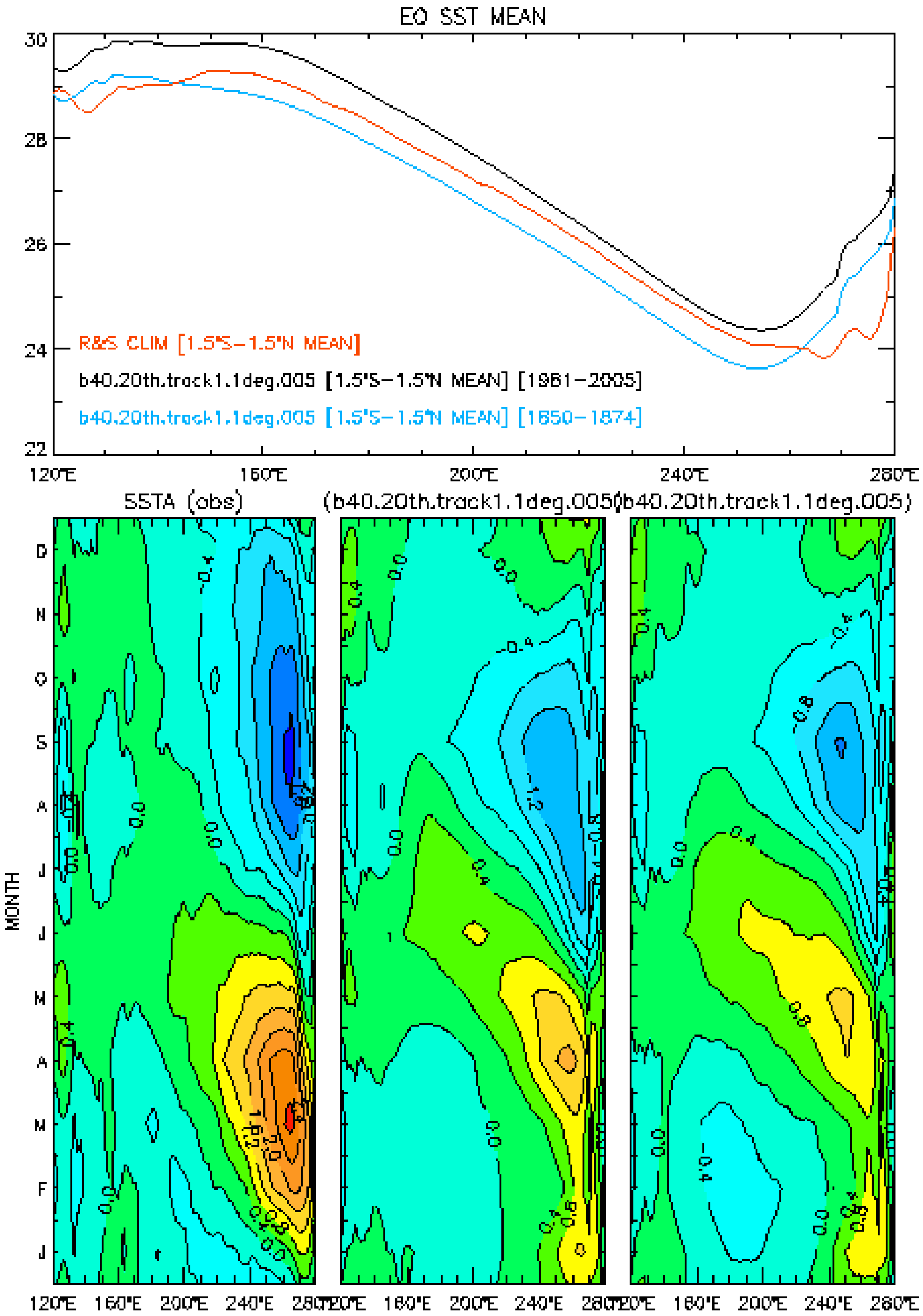
IDL:


SALT ZONAL-AVE (GLO) b40.20th.track1.1deg.005 [1981-2005]
DEPTH (km)
0
0.5
1.0
2.0
3.0
4.0
5.0
6.0
90°S
60°S
30°S
0°
30°N
60°N
90°N
34.0
34.4
34.8
35.2
38.00
37.60
37.20
36.80
36.40
36.00
35.60
35.20
34.80
34.40
34.00
33.60
33.20
32.80
32.40
32.00
31.60
31.20
30.80
30.40
30.00
(3.06e+01 to 3.61e+01 by 0.40psu)
(MODEL – OBS)
0.0
-0.4
-0.2
0.2
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
–0.20
–0.40
–0.60
–0.80
–1.00
–1.20
–1.40
–1.60
–1.80
–2.00
(–1.63e+00 to 1.01e+00 by 0.20psu)

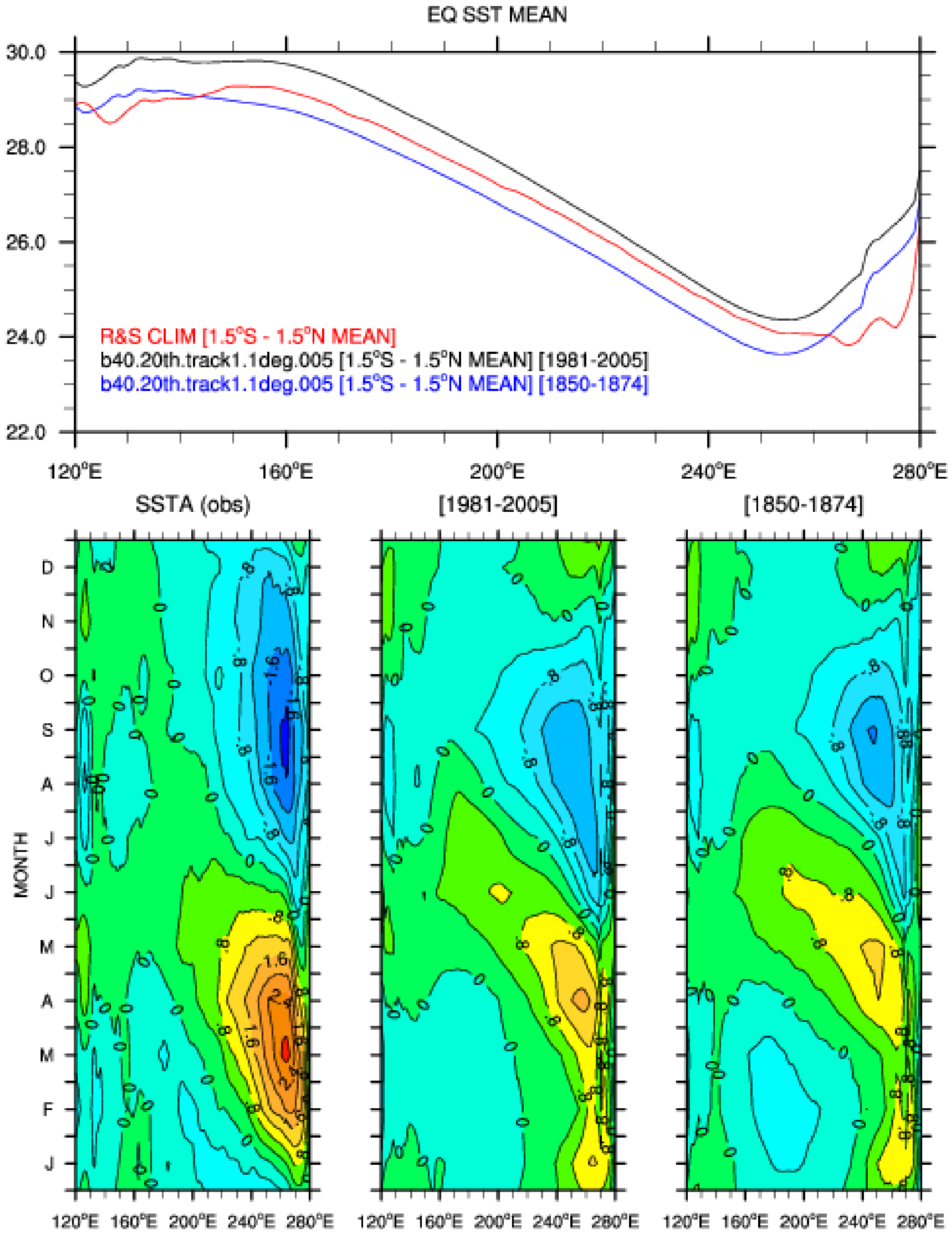
NCL:


SALT ZONAL-AVE (GLO) b40.20th.track1.1deg.005 [1981-2005]
0.0
0.5
1.0
2.0
3.0
4.0
5.0
6.0
DEPTH (km)
90°S
60°S
30°S
0°
30°N
60°N
90°N
30.6 to 36.1 by 0.4 psu
38
37.6
37.2
36.8
36.4
36
35.6
35.2
34.8
34.4
34
33.6
33.2
32.8
32.4
32
31.6
31.2
30.8
30.4
30
(MODEL - OBS)
0.0
0.5
1.0
2.0
3.0
4.0
5.0
6.0
DEPTH (km)
90°S
60°S
30°S
0°
30°N
60°N
90°N
-1.63 to 1.01 by 0.2 psu
2
1.8
1.6
1.4
1.2
1
.8
.6
.4
.2
0
-.2
-.4
-.6
-.8
-1
-1.2
-1.4
-1.6
-1.8
-2

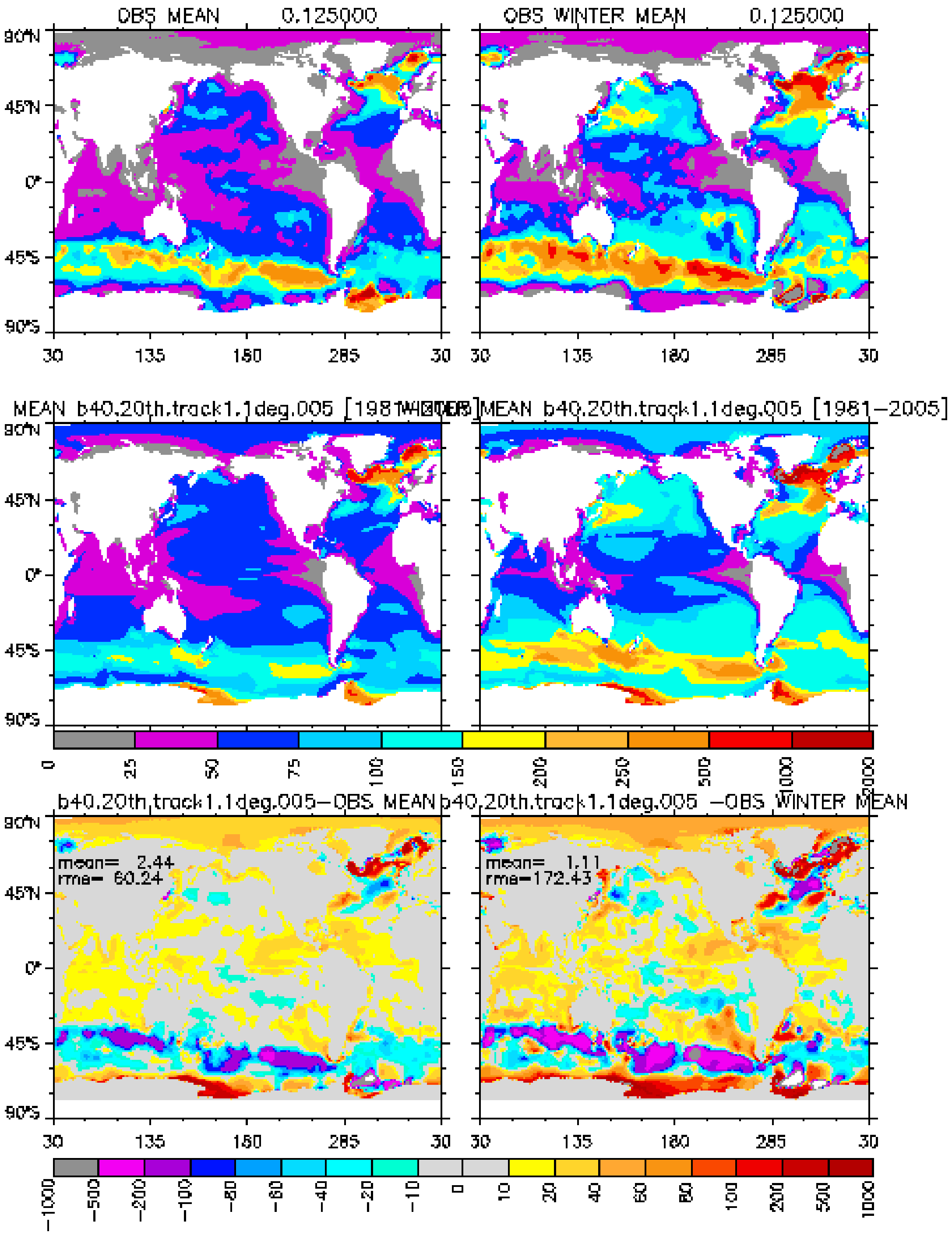
IDL:


EQ SST MEAN
30
28
26
24
22
R&S CLIM [1.5°S–1.5°N MEAN]
b40.20th.track1.1deg.005 [1.5°S–1.5°N MEAN] [1981–2005]
b40.20th.track1.1deg.005 [1.5°S–1.5°N MEAN] [1850–1874]
120°E
160°E
200°E
240°E
280°E
SSTA (obs)
(b40.20th.track1.1deg.005)
(b40.20th.track1.1deg.005)
MONTH
D
N
O
S
A
J
J
M
A
M
F
J
-0.4
0.0
0.4
0.8
1.2
1.6
2.0
-0.8
-1.2
120°E 160°E 200°E 240°E 280°E

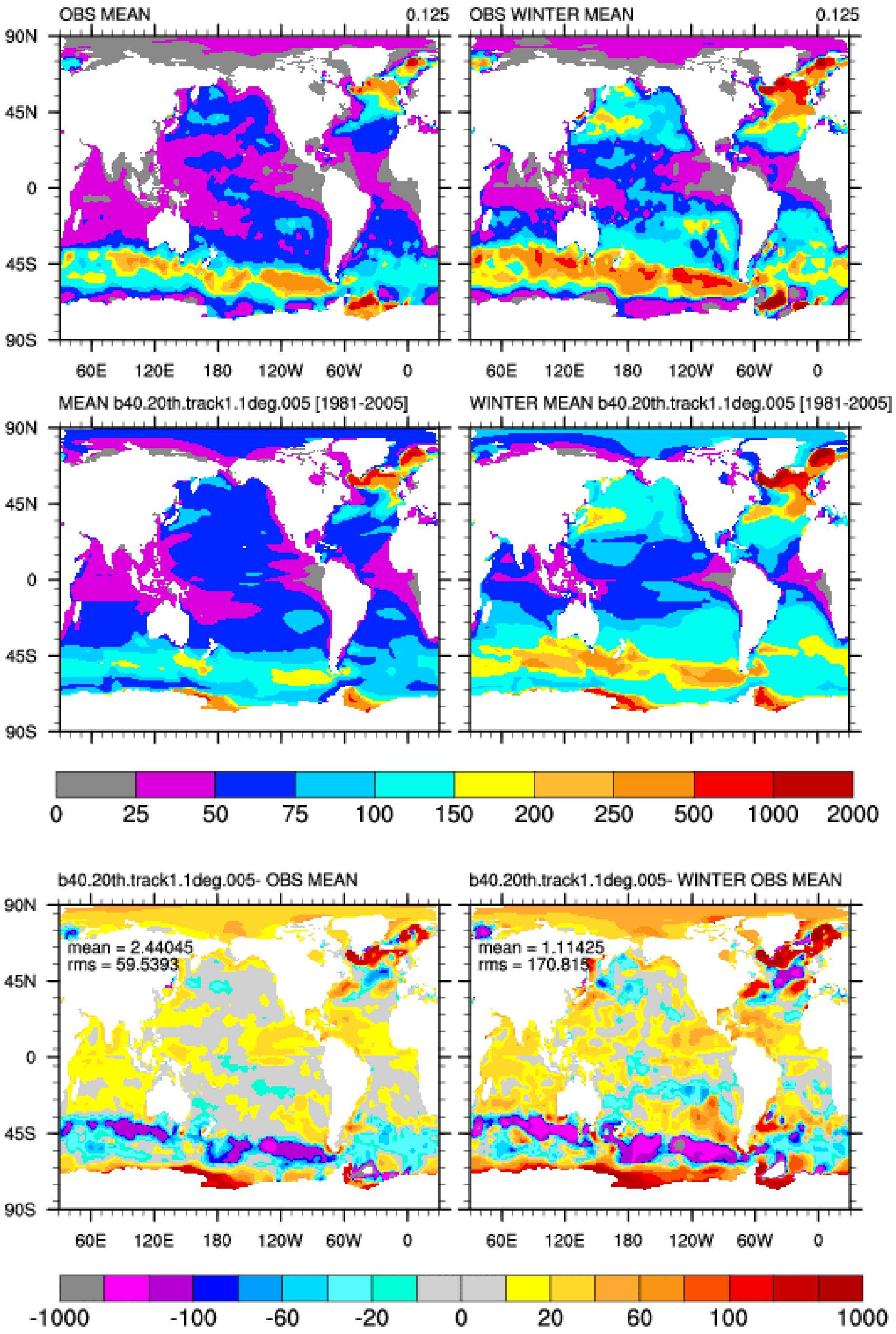
NCL:


EQ SST MEAN
30.0
28.0
26.0
24.0
22.0
120°E
160°E
200°E
240°E
280°E
R&S CLIM [1.5°S - 1.5°N MEAN]
b40.20th.track1.1deg.005 [1.5°S - 1.5°N MEAN] [1981-2005]
b40.20th.track1.1deg.005 [1.5°S - 1.5°N MEAN] [1850-1874]
SSTA (obs)
[1981-2005]
[1850-1874]
MONTH
D
N
O
S
A
J
J
M
A
M
F
J
120°E 160°E 200°E 240°E 280°E
120°E 160°E 200°E 240°E 280°E
120°E 160°E 200°E 240°E 280°E

IDL:


OBS MEAN 0.125000
OBS WINTER MEAN 0.125000
MEAN b40.20th.track1.1deg.005 [1981–2005]
WINTER MEAN b40.20th.track1.1deg.005 [1981–2005]
b40.20th.track1.1deg.005–OBS MEAN
b40.20th.track1.1deg.005 –OBS WINTER MEAN
mean= 2.44
rms= 60.24
mean= 1.11
rms=172.43
90°N
45°N
0°
45°S
90°S
30
135
180
285
30
0
25
50
75
100
150
200
250
500
1000
2000
–1000
–500
–200
–100
–80
–60
–40
–20
–10
0
10
20
40
60
80
100
200
500
1000

NCL:


OBS MEAN
0.125
OBS WINTER MEAN
0.125
90N
45N
0
45S
90S
60E
120E
180
120W
60W
0
MEAN b40.20th.track1.1deg.005 [1981-2005]
WINTER MEAN b40.20th.track1.1deg.005 [1981-2005]
0
25
50
75
100
150
200
250
500
1000
2000
b40.20th.track1.1deg.005- OBS MEAN
b40.20th.track1.1deg.005- WINTER OBS MEAN
mean = 2.44045
rms = 59.5393
mean = 1.11425
rms = 170.815
-1000
-100
-60
-20
0
20
60
100
1000

NCL vs. IDL
(the good, the bad, and the ugly)

- Many apparent similarities
  - ; (semicolon) starts a comment
  - Fortran-like syntax features: e.g. .eq. (NCL), eq (IDL)
  - Overall verbosity (lines of code): 14424 (NCL), 14388 (IDL)
  - Similar array syntax: 0-based element counting
- Significant differences
  - NCL: row-major like C; IDL: column-major like Fortran
  - Graphics code has a different model
  - NCL’s built-in support for missing values helps simplify code
  - NCL’s NetCDF-like variable model allows easier access for attributes and other metadata
  - IDL looping is definitely faster (script is compiled)
  - (Therefore) more important to use array syntax in NCL

NetCDF file handling comparison

```
; IDL open file, read variable, and handle attribute if it exists
fileid = ncdf_open(file_netcdf)
varid = ncdf_varid(fileid, 'SALT')
ncdf_varget, fileid, varid, salt
f_struct = ncdf_varinq(fileid,varid)
n_att = f_struct.natts
for n_att=0,n_att-1 do begin
  if ( ncdf_attname(fileid, varid, n_att) eq 'scale_factor' ) then begin
      ncdf_attget, fileid, varid, 'scale_factor', scale_field
      good = where(salt gt -10. AND salt lt 1.e10)
      salt[good] = scale_field * salt[good]
  endif
endfor
```

```
;NCL open file, read variable and handle attribute if it exists
;Note: attribute is part of variable, _FillValue support ensures that
;missing values are automatically ignored

fileid = addfile(file_netcdf,"r”)
salt = fileid->SALT
if (isatt(salt,”scale_factor”)) then
  salt = salt * salt@scale_factor
end if
```

(the good)

Calculating weighted average (IDL)

```
; variable field contains temperature anomalies : lon x lat x time
; the task is to average values near the equator from y_min to y_max
; tarea has the area weights on the T grid
; anom is lon x time averaged over lat
; triple-nested loop handles each array element individually

anom = dblarr(nx,nt)
anom(*,*) = double(0.)
for n=0,nt-1 do begin
  for i=0,nx-1 do begin
    area_wt = double(0.)
    max_anom = double(0.)
    for j=y_min,y_max do begin
      if ( field(i,j,n) lt missing ) then begin
        anom(i,n) = anom(i,n) + tarea(i,j) * field(i,j,n)
        if (anom(i,n) gt max_anom) then max_anom = anom(i,n)
        area_wt = area_wt + tarea(i,j)
      endif
    endfor
    if ( area_wt ne 0. ) then begin
      anom(i,n) = anom(i,n) / area_wt
    endif else begin
      anom(i,n) = missing
    endelse
  endfor
endfor
```

Calculating weighted average (NCL)

```
; variable field contains temperature anomalies : time x lat x lon
; the task is to average values near the equator from y_min to y_max
; tarea has the area weights on the T grid
; anom is time x lon averaged over lat
; conforming the dimensions of tarea with the field variable allows
; NCL to perform element by element array arithmetic and avoids loops
; However, note that the conform_dims  function creates an array with nt
; redundant copies of the same data. The temporary array then needs to be
; deleted.

sub_y = y_max - y_min + 1
tarea_conform = conform_dims((/ nt, sub_y, nx /), \
                               tarea(y_min:y_max,:), (/ 1, 2 /))
subfield = tarea_conform * field(:,y_min:y_max,:) ; time * lat * lon
anom = dim_sum_n_Wrap(subfield,1)
tarea_anom = dim_sum_n_Wrap(tarea_conform,1)
anom = anom / tarea_anom
delete(tarea_conform)
delete(subfield)
```

(the bad)

Smoothing code for mixed layer depth value (IDL)

```
; a more complicated code with multiple nested loops that requires access to
; adjacent cells along 2 dimensions during each pass.
; Only the beginning shown here

for ns=1,ns_max do begin

  print, ' smoothing pass .... ', ns

  field_temp_1 = MLD

  for j=1,ny-2 do begin
    for i=0,nx-1 do begin

    im1 = i-1
    ip1 = i+1
    if ( i eq 0    ) then  im1 = nx-1
    if ( i eq nx-1 ) then  ip1 = 0

      cc = double(tarea(i  ,j  ))
      ce = double(tarea(ip1,j  ))
      cw = double(tarea(im1,j  ))
      cn = double(tarea(i  ,j+1))
      cs = double(tarea(i  ,j-1))
      sum = cc + ce + cw + cn + cs
      cc = cc / sum
      ce = ce / sum
…
```

Smoothing code for mixed layer depth value (NCL)

```
; Sample lines of my attempt to recreate this code in NCL eliminating loops.
; Eventually I got it to work more or less, but it still did not have the
; desired performance and it just looks too complicated to be maintainable.

tarea_sum = tarea
tarea_sum(1:ny-2,1:nx-2) = \
        tarea(1:ny-2,1:nx-2) + tarea(1:ny-2,:nx-3) + tarea(1:ny-2,2:nx-1) + \
        tarea(:ny-3,1:nx-2) + tarea(2:ny-1,1:nx-2)
tarea_sum(1:ny-2,0) = \
        tarea(1:ny-2,0) +tarea(1:ny-2,nx-1) + tarea(1:ny-2,1) + \
        tarea(:ny-3,0) + tarea(2:ny-1,0)
…

MLD_new(:,:,1:ny-2,1:nx-2) = \
        MLD(:,:,1:ny-2,1:nx-2) * cc_c(:,:,1:ny-2,1:nx-2) + \
        MLD(:,:,1:ny-2,:nx-3) * cw_c(:,:,1:ny-2,1:nx-2) + \
        MLD(:,:,1:ny-2,2:nx-1) * ce_c(:,:,1:ny-2,1:nx-2) + \
        MLD(:,:,:ny-3,1:nx-2) * cn_c(:,:,1:ny-2,1:nx-2) + \
        MLD(:,:,2:ny-1,1:nx-2) * cs_c(:,:,1:ny-2,1:nx-2)
; etc.
```

(the ugly)

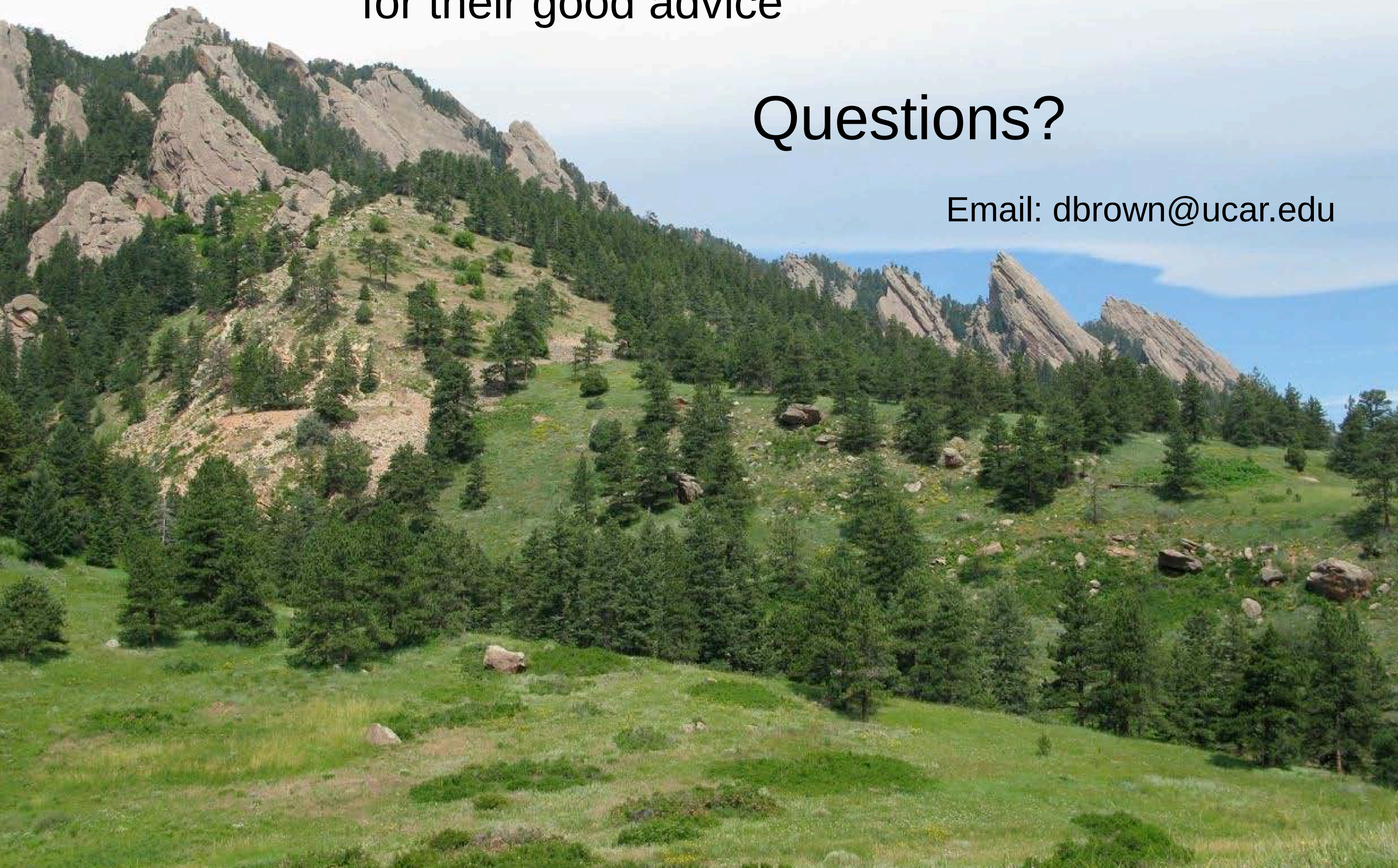
When the effort to avoid looping in NCL means the code starts looking like this, it's probably time to switch to Fortran and create a shared object.

Summary

- New OMWG diagnostic suite verified and available by the end of the year
- Freely distributable open source
- Performance and graphics similar to existing suite
- Future improvements possible
- Suggestions welcome

NCL: http://www.ncl.ucar.edu

Thanks to Susan Bates and Gokhan Danabasoglu for their support of this project and Dennis Shea and Mary Haley for their good advice



Questions?

Email: dbrown@ucar.edu