Tutorial 13:
Building Your Observation Files

4D-Var Observations NetCDF File

During 4D-Var, the input observations data are used in the following ROMS routines:

- **Utility/obs_initial.F**
- **Utility/obs_read.F**
- **Utility/obs_write.F**
- **Utility/obs_scale.F**
- **Utility/obs_depth.F**
- **Utility/extract_obs.F**
- **Adjoint/ad_extract_obs.F**
- **Adjoint/ad_htobs.F**
- **Adjoint/ad_misfit.F**

Observation Data Sources

Sea Surface Temperature

NOAA PFEG Coastwash OpenDAP (1/10 degree, 5 day composite). It is a blended product including microwave AMSR-E, AVHRR, MODIS, and GOES satellites (load_sst_pfeg.m).

http://thredds1.pfeg.noaa.gov:8080/thredds/dodsC/satellite/BA/ssta/5day

Sea Surface Height

Gridded altimetry data from AVISO (load_ssh_aviso.m).

http://opendap.aviso.oceanobs.com/thredds/dodsCduacs_global_nrt_msla_merged_h?%s

Hydrographic data

UK Met Office observations datasets

http://hadobs.metoffice.com/en3

Metadata

Dimensions:

survey	Number of unique data surveys
state_variable	Number of ROMS state variables
datum	Observations counter, unlimited dimension

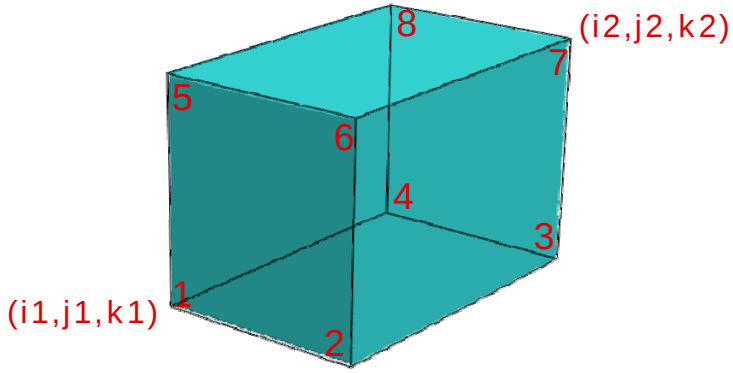
Variables:

Nobs(survey)	Number of observations per survey
survey_time(survey)	Survey time (days)
obs_variance(state_variable)	global time and space observation variance
obs_type(datum)	State variable ID associated with observation
obs_provenance(datum)	observation origin
obs_time(datum)	Time of observation (days)
obs_lon(datum)	Longitude of observation (degrees_east)
obs_lat(datum)	Latitude of observation (degrees_north)
obs_depth(datum)	Depth of observation (meters or level)
obs_Xgrid(datum)	X-grid observation location (nondimensional)
obs_Ygrid(datum)	Y-grid observation location (nondimensional)
obs_Zgrid(datum)	Z-grid observation location (nondimensional)
obs_error(datum)	Observation error, assigned weight
obs_value(datum)	Observation value

Observations NetCDF

```
dimensions:

        survey = 13 ;
        state_variable = 7 ;
        datum = UNLIMITED ; // (6815 currently)

variables:

        int spherical ;
                spherical:long_name = "grid type logical switch" ;
                spherical:flag_values = "0, 1" ;
                spherical:flag_meanings = "Cartesian spherical" ;
        int Nobs(survey) ;
                Nobs:long_name = "number of observations with the same survey time" ;
        double survey_time(survey) ;
                survey_time:long_name = "survey time" ;
                survey_time:units = "days since 1968-05-23 00:00:00 GMT" ;
                survey_time:calendar = "gregorian" ;
        double obs_variance(state_variable) ;
                obs_variance:long_name = "global time and space observation variance" ;
        int obs_type(datum) ;
                obs_type:long_name = "model state variable associated with observation" ;
                obs_type:flag_values = 1, 2, 3, 4, 5, 6, 7 ;
                obs_type:flag_meanings = "zeta ubar vbar u v temperature salinity" ;
        int obs_provenance(datum) ;
                obs_provenance:long_name = "observation origin" ;
                obs_provenance:flag_values = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ;
                obs_provenance:flag_meanings = "gridded_AVISO_SLA ..." ;
        double obs_time(datum) ;
                obs_time:long_name = "time of observation" ;
                obs_time:units = "days since 1968-05-23 00:00:00 GMT" ;
                obs_time:calendar = "gregorian" ;
        double obs_lon(datum) ;
                obs_lon:long_name = "observation longitude" ;
                obs_lon:units = "degrees_east" ;
                obs_lon:standard_name = "longitude" ;
        double obs_lat(datum) ;
                obs_lat:long_name = "observation latitude" ;
                obs_lat:units = "degrees_north" ;
                obs_lat:standard_name = "latitude" ;
        double obs_depth(datum) ;
                obs_depth:long_name = "depth of observation" ;
                obs_depth:negative = "downwards" ;
        double obs_Xgrid(datum) ;
                obs_Xgrid:long_name = "x-grid observation location" ;
        double obs_Ygrid(datum) ;
                obs_Ygrid:long_name = "y-grid observation location" ;
        double obs_Zgrid(datum) ;
                obs_Zgrid:long_name = "z-grid observation location" ;
        double obs_error(datum) ;
                obs_error:long_name = "observation error covariance" ;
        double obs_value(datum) ;
                obs_value:long_name = "observation value" ;
```

Variable	ID
ζ	1
$\bar{u}$	2
$\bar{v}$	3
u	4
v	5
temp	6
salt	7



Global Attributes

```
variables:

. . .

        int obs_provenance(datum) ;
                obs_provenance:long_name = "observation origin" ;
                obs_provenance:flag_values = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ;
                obs_provenance:flag_meanings = "gridded_AVISO_SLA blended_SST XBT_Met_Office CTD_temperature_Met_Office CTD_salinity_Met_Office
ARGO_temperature_Met_Office ARGO_salinity_Met_Office CTD_temperature_CalCOFI CTD_salinity_CalCOFI CTD_temperature_GLOBEC CTD_salinity_GLOBEC" ;

. . .


// global attributes:
                :type = "ROMS observations" ;
                :title = "California Current System, 1/3 degree resolution (WC13)" ;
                :Conventions = "CF-1.4" ;
                :state_variables = "\n",
                        "1: free-surface (m) \n",
                        "2: vertically integrated u-momentum component (m/s) \n",
                        "3: vertically integrated v-momentum component (m/s) \n",
                        "4: u-momentum component (m/s) \n",
                        "5: v-momentum component (m/s) \n",
                        "6: potential temperature (Celsius) \n",
                        "7: salinity (nondimensional)" ;
                :obs_provenance = "\n",
                        " 1: gridded AVISO sea level anomaly \n",
                        " 2: blended satellite SST \n",
                        " 3: XBT temperature from Met Office \n",
                        " 4: CTD temperature from Met Office \n",
                        " 5: CTD salinity from Met Office \n",
                        " 6: ARGO floats temperature from Met Office \n",
                        " 7: ARGO floats salinity from Met Office \n",
                        " 8: CTD temperature from CalCOFI \n",
                        " 9: CTD salinity from CalCOFI \n",
                        "10: CTD temperature from GLOBEC \n",
                        "11: CTD salinity from GLOBEC" ;
                :variance_units = "squared state variable units" ;
                :obs_sources = "\n",
                        "http://opendap.aviso.oceanobs.com/thredds/dodsC \n",
                        "http://hadobs.metoffice.com/en3" ;
                :history = "4D-Var observations, Wednesday - July 7, 2010 - 12:20:37.2629 AM" ;
```

Longitude and Latitude Locations (Optional)

- **The obs_lon and obs_lat values are only necessary to compute the fractional grid locations (obs_Xgrid, obs_Ygrid) during pre-processing using obs_ijpos.m**

- **The obs_lon and obs_lat are not used directly in ROMS when running the 4D-Var algorithms for efficiency and because of the complexity of curvilinear grids. The fractional grid locations obs_Xgrid and obs_Ygrid are used instead.**

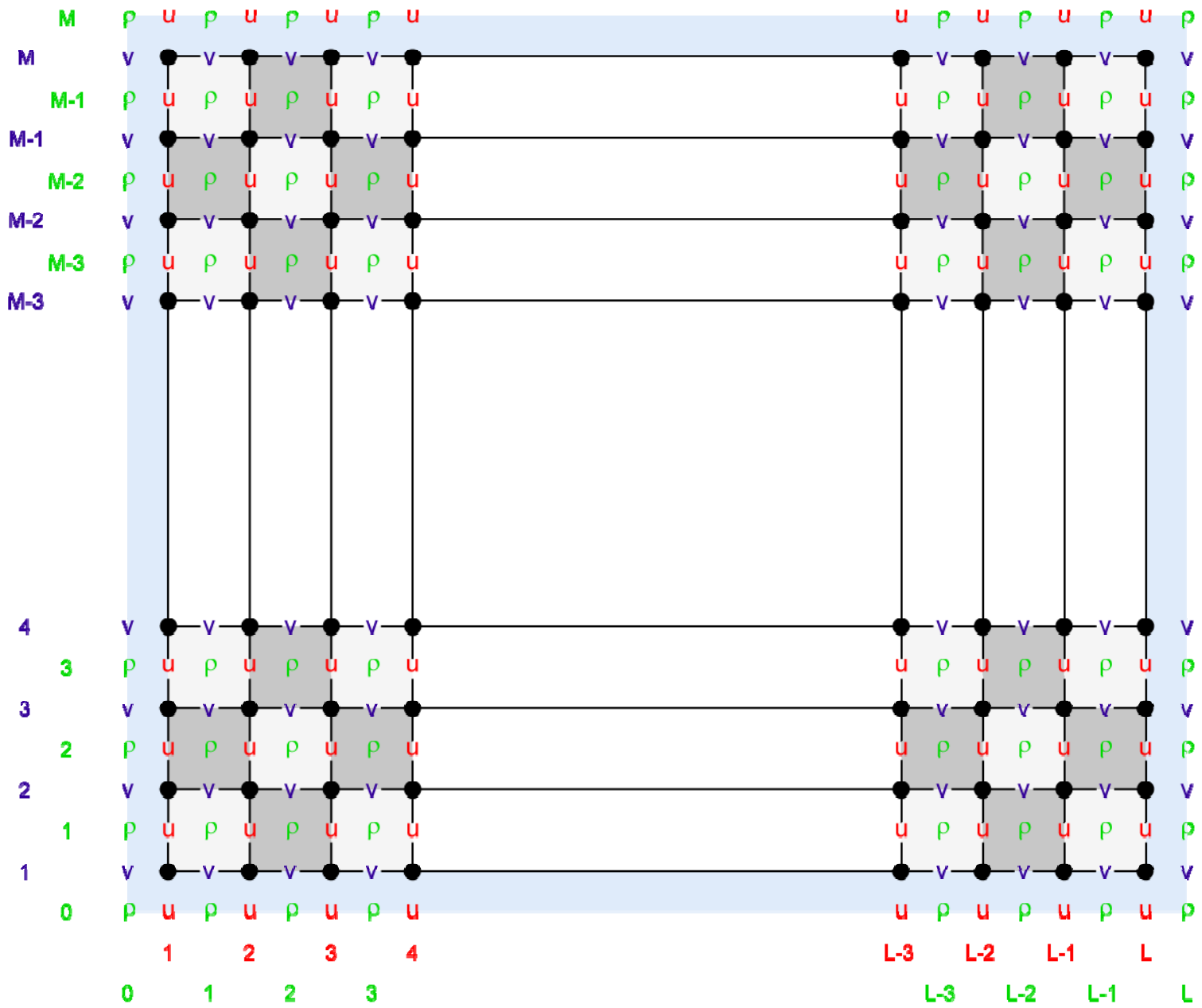
- **However, they are useful during post-processing and plotting results into maps or when changing application grid.**

- **The size of the observation NetCDF can be an issue, it is smaller if obs_lon and obs_lat are omitted.**

ROMS GRID

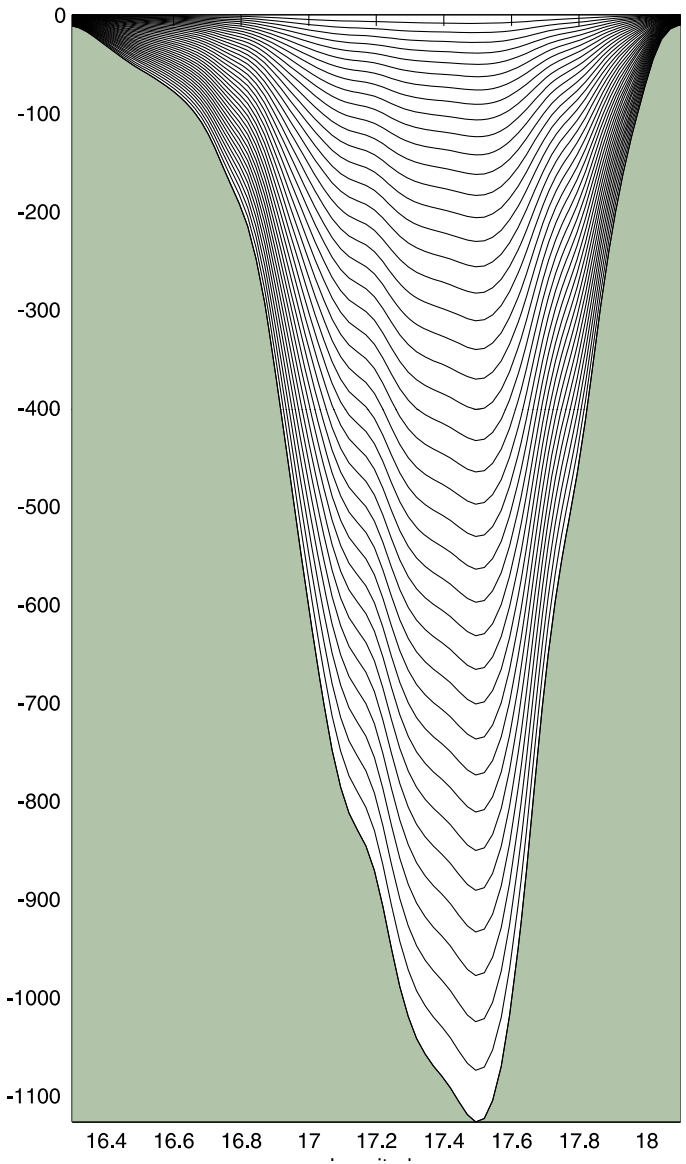
- **Horizontal curvilinear orthogonal coordinates on an Arakawa C-grid.**
- **Terrain-following coordinates on a staggered vertical grid.**
- **Coarse-grained parallel tile partitions.**
- **Only distributed-memory (MPI) is possible in all the adjoint-based algorithms. Shared-memory (OpenMP) is not possible because of the way that the adjoint model is coded.**
- **In distributed-memory, we need the adjoint of the MPI exchanges between the tiles.**

Staggered C-Grid, RHO-points

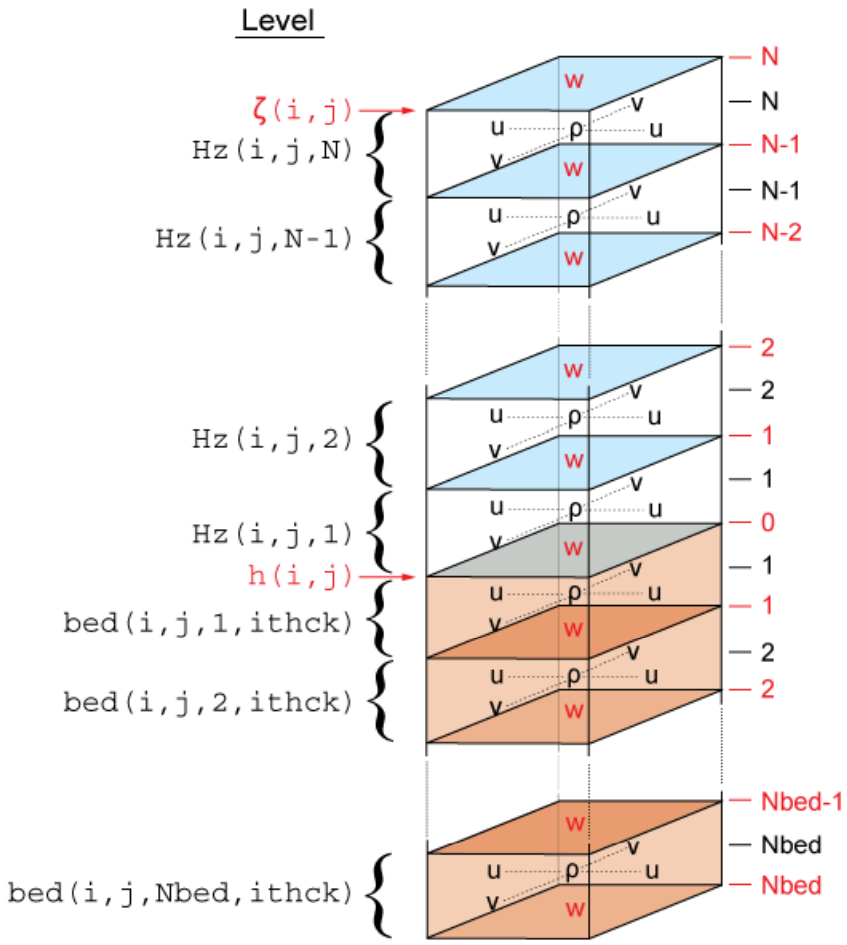


Vertical Terrain-following Coordinates

Vieste (Italy)

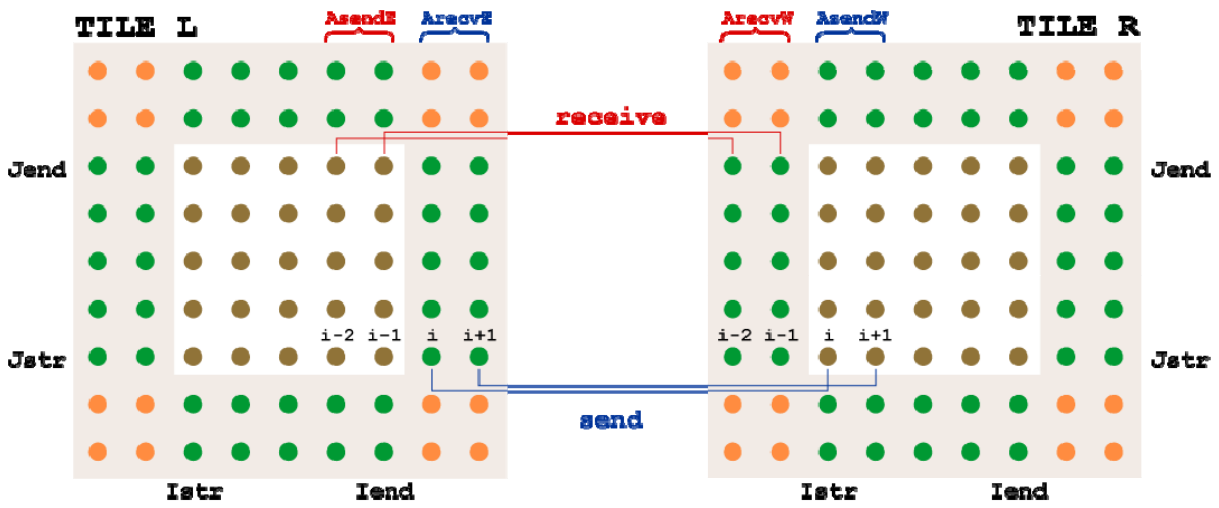


Dubrovnik (Croatia)



East-West MPI Communications

Nonlinear

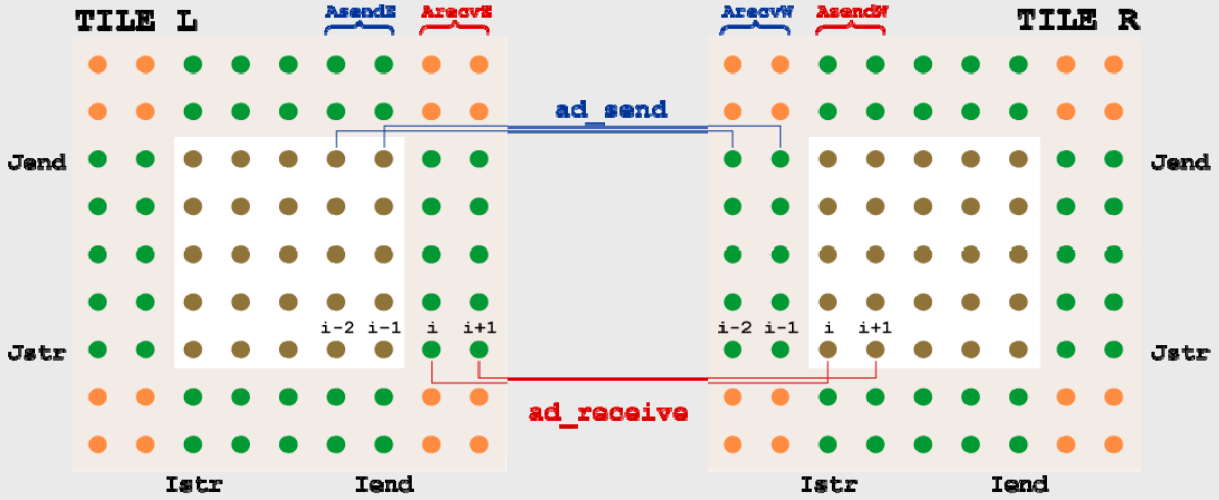


$$v^L_i = v^R_i \qquad v^R_{i-2} = v^L_{i-2}$$

$$v^L_{i+1} = v^R_{i+1} \qquad v^R_{i-1} = v^L_{i-1}$$

With Respect To Tile R

Adjoint



$$ad\_v^L_{i-2} = ad\_v^L_{i-2} + ad\_v^R_{i-2} \; ; v^R_{i-2} = 0 \qquad ad\_v^R_i = ad\_v^R_i + ad\_v^L_i \; ; v^L_i = 0$$

$$ad\_v^L_{i-1} = ad\_v^L_{i-1} + ad\_v^R_{i-1} \; ; v^R_{i-1} = 0 \qquad ad\_v^R_{i+1} = ad\_v^R_{i+1} + ad\_v^L_{i+1} \; ; v^L_{i+1} = 0$$

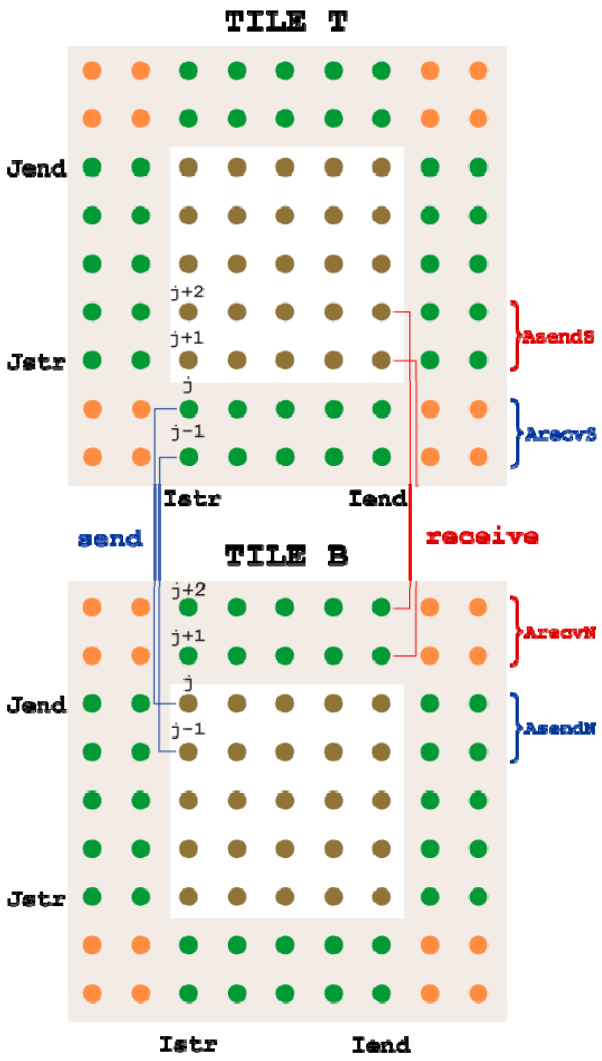
With Respect To Tile R

North-South MPI Communications

Nonlinear

$V^B_{j+2} = V^T_{j+2}$

$V^B_{j+1} = V^T_{j+1}$



$V^T_j = V^B_j$

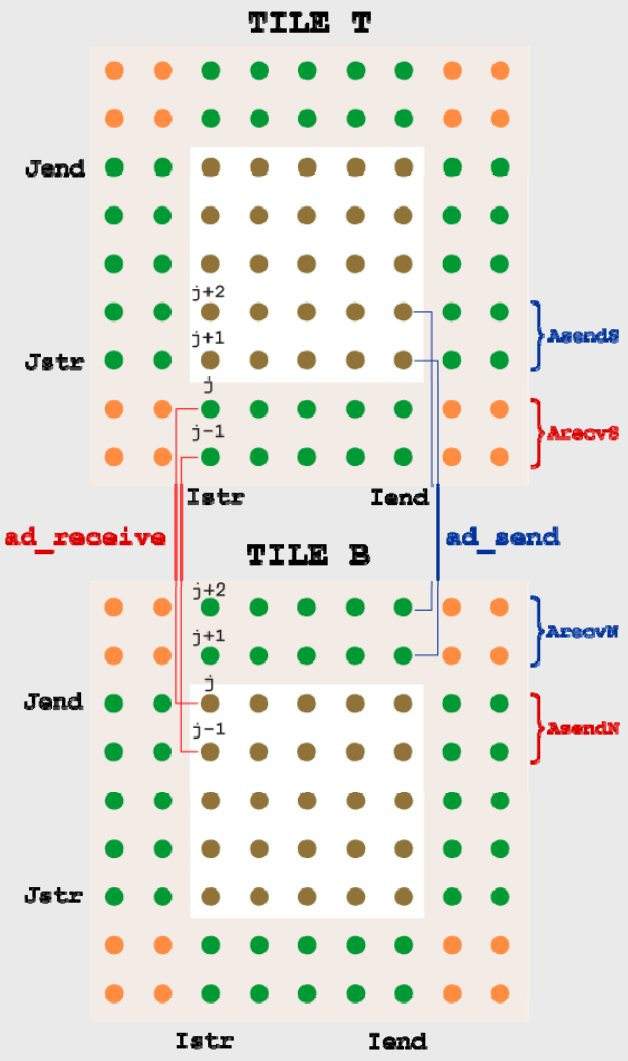
$V^T_{j-1} = V^B_{j-1}$

With Respect to Tile B

Adjoint

$ad\_V^T_{j+2} = ad\_V^T_{j+2} + ad\_V^B_{j+2}\ ;\ ad\_V^B_{j+2} = 0$

$ad\_V^T_{j+1} = ad\_V^T_{j+1} + ad\_V^B_{j+1}\ ;\ ad\_V^B_{j+1} = 0$



$ad\_V^B_j = ad\_V^B_j + ad\_V^T_j\ ;\ ad\_V^T_j = 0$

$ad\_V^B_{j-1} = ad\_V^B_{j-1} + ad\_V^T_{j-1}\ ;\ ad\_V^T_{j-1} = 0$

Processing

- **Compute observation's fractional (ξ, η) grid coordinates locations from its (obs_lon, obs_lat).**
- **Find how many unique survey times occur within the data set (survey dimension)**
- **Count observations available per survey (Nobs) and assign their times (survey_time)**
- **Sort the observation in ascending time order and observation type for efficiency**
- **Save a copy of the observation file**
- **Several matlab scripts to process observations**

Matlab Scripts

C_observations.m	- Creates 4D-Var observation NetCDF file.
d_observations.m	- Driver to process 4D-Var observation NetCDF file.
Obs_merge.m	- Merges specified 4D-Var observation NetCDF files.
obs_read.m	- Reads observation NetCDF file and loads all data into a structure array.
obs_write.m	- Writes all observation data in structure array into an existing NetCDF file.
inside.m	- Checks if a point is strictly inside of the region defined by a polygon. This is an old Matlab function which is no longer supported. It is very useful in finding outliers in observations (inpolygon).
obs_ijpos.m	- Computes observation locations in ROMS fractional coordinates. It uses the deprecated inside.m Matlab function.
super_obs.m	- Checks the provided observation data (4D-Var NetCDF file or structure) and creates super observations if there is more than one datum of the same observation type per grid cell.

Matlab Scripts

plot_super.m	- Sample script to compute and plot super observations from specified 4D-Var observations NetCDF file.
d_ssh_obs.m	- Driver template to extract SSH observations for AVISO. It creates and writes an observation file. Then, it computes super observations and creates and writes a super observations NetCDF file.
d_sst_obs.m	- Driver template to extract SST observations from satellite data. It creates and writes an observation file. Then, it computes super observations and creates and writes a super observations NetCDF file.
load_ssh_aviso.m	- Extracts AVISO sea level anomaly for the specified region and period of interest from ROMS Grid file.
load_sst_pfeg.m	- Extracts satellite sea surface temperature for the specified region and period of interest from ROMS Grid file. The SST data is from the OpenDAP catalog maintained by NOAA PFEG Coastwatch in California. The resolution is 0.1 degree global 5-day average composite.

Matlab Observation Structure

Observations data structure S:

S.ncfile	NetCDF file name (string)
S.Nsurvey	number of observations surveys
S.Nstate	number of state variables
S.Ndatum	total number of observations
S.spherical	spherical grid switch
S.Nobs	number of observations per survey
S.survey_time	time for each survey time
S.variance	global variance per state variable
S.type	state variable associated with observation
S.time	time for each observation
S.depth	depth of observation
S.Xgrid	observation fractional x-grid location
S.Ygrid	observation fractional y-grid location
S.Zgrid	observation fractional z-grid location
S.error	observation error
S.value	observation value

Matlab Observation Structure

The following optional variables will be read if available:

S.provenance	observation origin
S.lon	observation longitude
S.lat	observation latitude

The following variable attributes will be read if available:

S.state_flag_values	obs_type flag_values attribute
S.state_flag_meanings	obs_type flag_meanings attribute
S.origin_flag_values	obs_provenance flag_values attribute
S.origin_flag_meanings	obs_provenance flag_meaning attribute

The following global attributes will be read if available:

S.global_variables	state_variables global attribute
S.global_provenance	obs_provenance global attribute
S.global_sources	obs_sources global attribute

obs_ijpos.m

```
function [Xgrid,Ygrid]=obs_ijpos(GRDname,obs_lon,obs_lat,Ccorrection);

%
% OBS_IJPOS:  Computes observation locations in ROMS fractional coordinates
%
% [Xgrid,Ygrid]=obs_ijpos(GRDname,obs_lon,obs_lat,Ccorrection)
%
% This function computes the observation locations (Xgrid,Ygrid) in terms
% of ROMS fractional (I,J) coordinates. This is done to facilitate the
% processing of the observation operators inside ROMS. All the observations
% are assumed to be located at RHO-points.
%
% On Input:
%
%    GRDname       NetCDF grid file name (string)
%    obs_lon       observation longitude (positive, degrees_east)
%    obs_lat       observation latitude  (positive, degrees_north)
%    Ccorrection   switch to apply small correction in curvilinear
%                    grids (0: no, 1: yes)
%
% On Ouput:
%
%    Xgrid         observation fractional x-grid location
%    Ygrid         observation fractional y-grid location
%
% Notice:  Outlier observations outside of ROMS grid has an NaN value in
%          Xgrid and Ygrid.
%

% svn $Id: obs_ijpos.m 485 2010-07-07 18:10:13Z arango $
%=========================================================================%
%  Copyright (c) 2002-2010 The ROMS/TOMS Group                            %
```

super_obs.m

```
function [Sout]=super_obs(Sinp);

%
% SUPER_OBS:  Creates super observations when necessary
%
% [Sout]=super_obs(Sinp)
%
% This function checks the provided observation data and creates
% super observations if there is more than one datum of the same
% observation type per grid cell. At input, Sinp is either a
% 4D-Var observation NetCDF file or data structure.
%
% On Input:
%
%    Sinp    Observations data structure or NetCDF file name
%
%
% On Output:
%
%    Sout    Binned observations data (structure array):
%
%              Sout.ncfile       NetCDF file name (string)
%              Sout.Ndatum       total number of observations
%              Sout.spherical    spherical grid switch
%              Sout.Nobs         number of observations per survey
%              Sout.survey_time  time for each survey time
%              Sout.variance     global variance per state variable
%              Sout.type         state variable associated with observation
%              Sout.time         time for each observation
%              Sout.depth        depth of observation
%              Sout.Xgrid        observation fractional x-grid location
%              Sout.Ygrid        observation fractional y-grid location
```

Assumptions

- **All scalar observations are assumed to be at RHO-points.**
- **All vector observations are assumed to be rotated to ROMS curvilinear grid, if applicable. Vector observations are always measured at the same location.**
- **All observation horizontal locations are assumed to be in fractional curvilinear grid coordinates.**
- **Vertical locations can be in fractional levels (1:N) or actual depths (negative values).**
- **Removal of tidal signal?**
- **Filtering of non-resolved processes?**

Observation Operators

- **Currently, all observations must be in terms of model state variables (same units):**
  - **2D configuration: zeta, ubar, vbar**
  - **3D configuration: zeta, u, v, T, S, …**
- **There is no time interpolation of the model solution at observation times:**

  time - 0.5*dt < ObsTime < time + 0.5*dt
- **There is no observation quality control (screening) inside ROMS, except ObsScale.**
- **No observation constraints are implemented (Satellite SST measurements)**

Observation Interpolation

- **Only spatial linear interpolation is coded.**
- **If land/sea masking, the interpolation coefficients are weighted by the mask.**
- **If shallower than z_r(:,:,N), observations are assigned to the surface level.**
- **If deeper than z_r(:,:,1), observations are assigned to bottom level.**

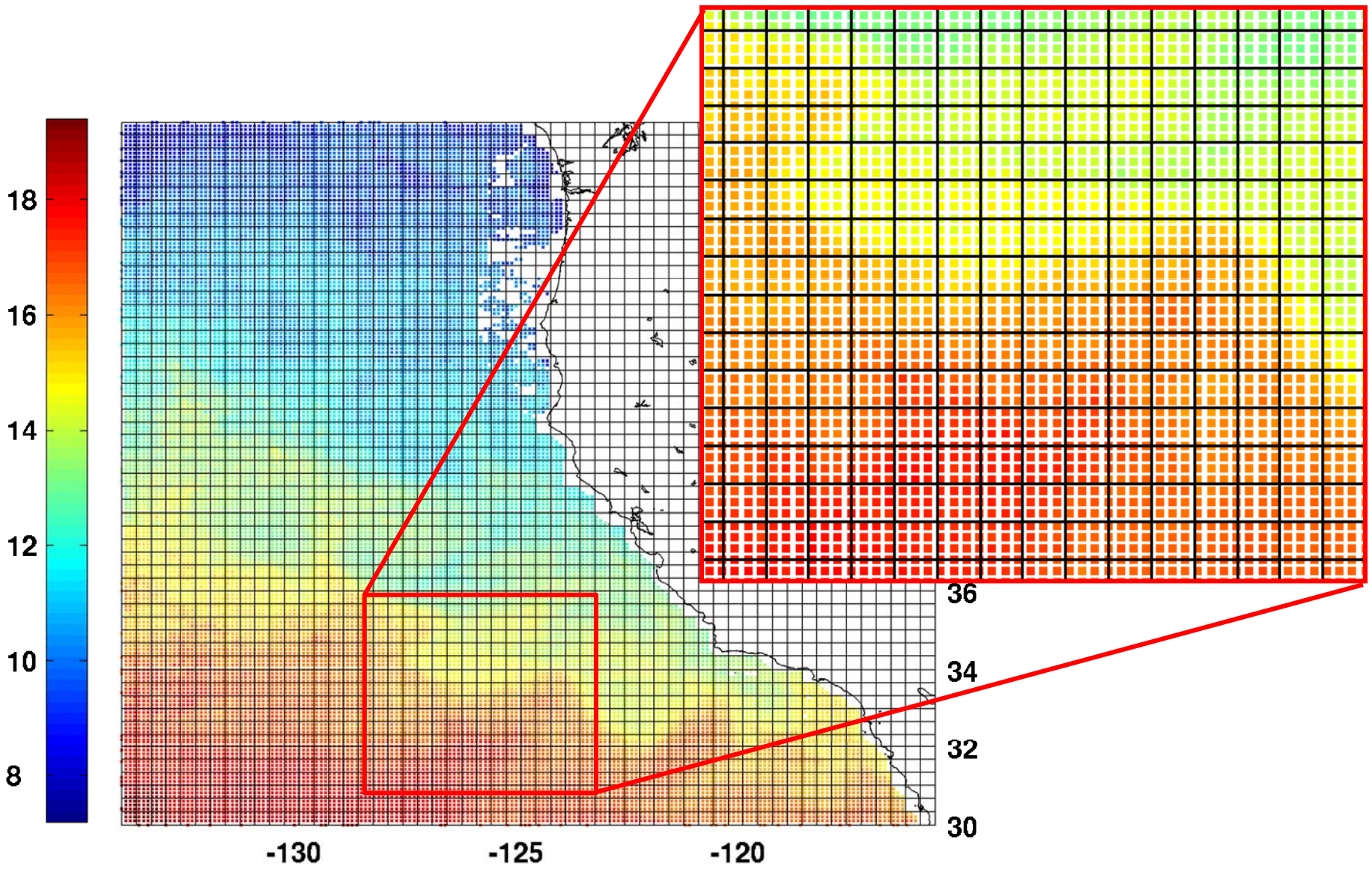
Recommendations

- **Create a NetCDF file for each observation type.**
- **Use a processing program to meld NetCDF observation files (obs_meld.m).**
- **Keep a master copy of each observation file, in case you are running your application at different resolutions.**
- **Decimation of observations. Finite volume representation super observations (super_obs.m).**

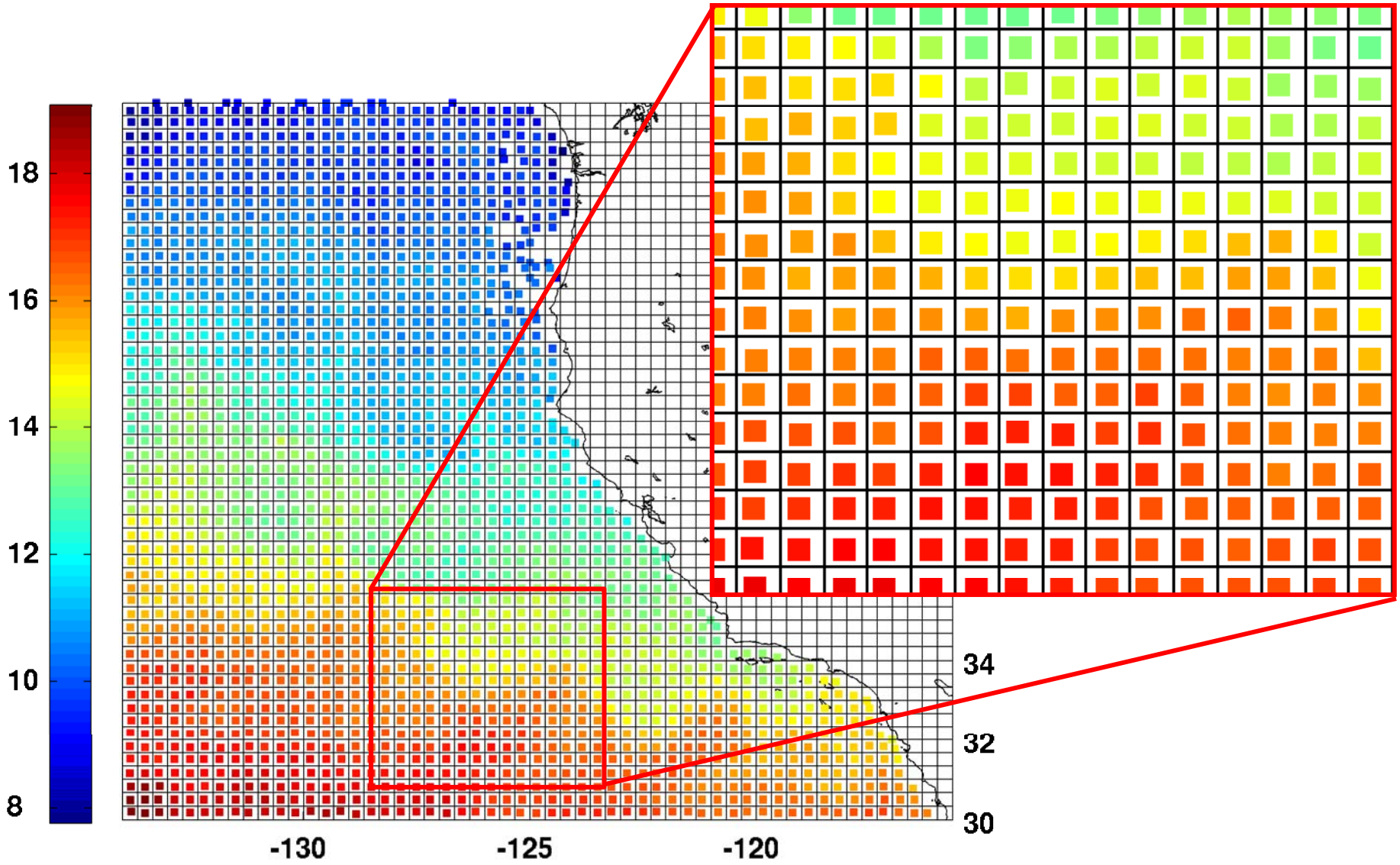
d_sst_obs.m

```
%
%  D_SST_OBS:  Driver script to create a 4D-Var SST observations file.
%
%  This a user modifiable script that can be used to prepare ROMS 4D-Var
%  SST observations NetCDF file. The SST data is extracted from the
%  extensive OpenDAP catalog maintained by NOAA PFEG Coastwatch in
%  California using script 'load_sst_pfeg.m'. The SST are 0.1 degree
%  global 5-day average composite. USERS can use this as a prototype
%  for their application.
%

% svn $Id: d_sst_obs.m 490 2010-07-11 03:58:35Z arango $
%=========================================================================%
%  Copyright (c) 2002-2010 The ROMS/TOMS Group                            %
%    Licensed under a MIT/X style license                                 %
%    See License_ROMS.txt                          Hernan G. Arango       %
%=========================================================================%

%  Set input/output NetCDF files.

 my_root = '/home/arango/ocean/toms/repository/test';

 GRDfile = fullfile(my_root, 'WC13/Data', 'wc13_grd.nc');
 OBSfile = 'wc13_sst_obs.nc';
 SUPfile = 'wc13_sst_super_obs.nc';

%  Set ROMS state variable type classification.

Nstate=7;            % number of ROMS state variables

state.zeta = 1;      % free-surface
state.ubar = 2;      % vertically integrated u-momentum
```

SST Observations (1/10 degree)



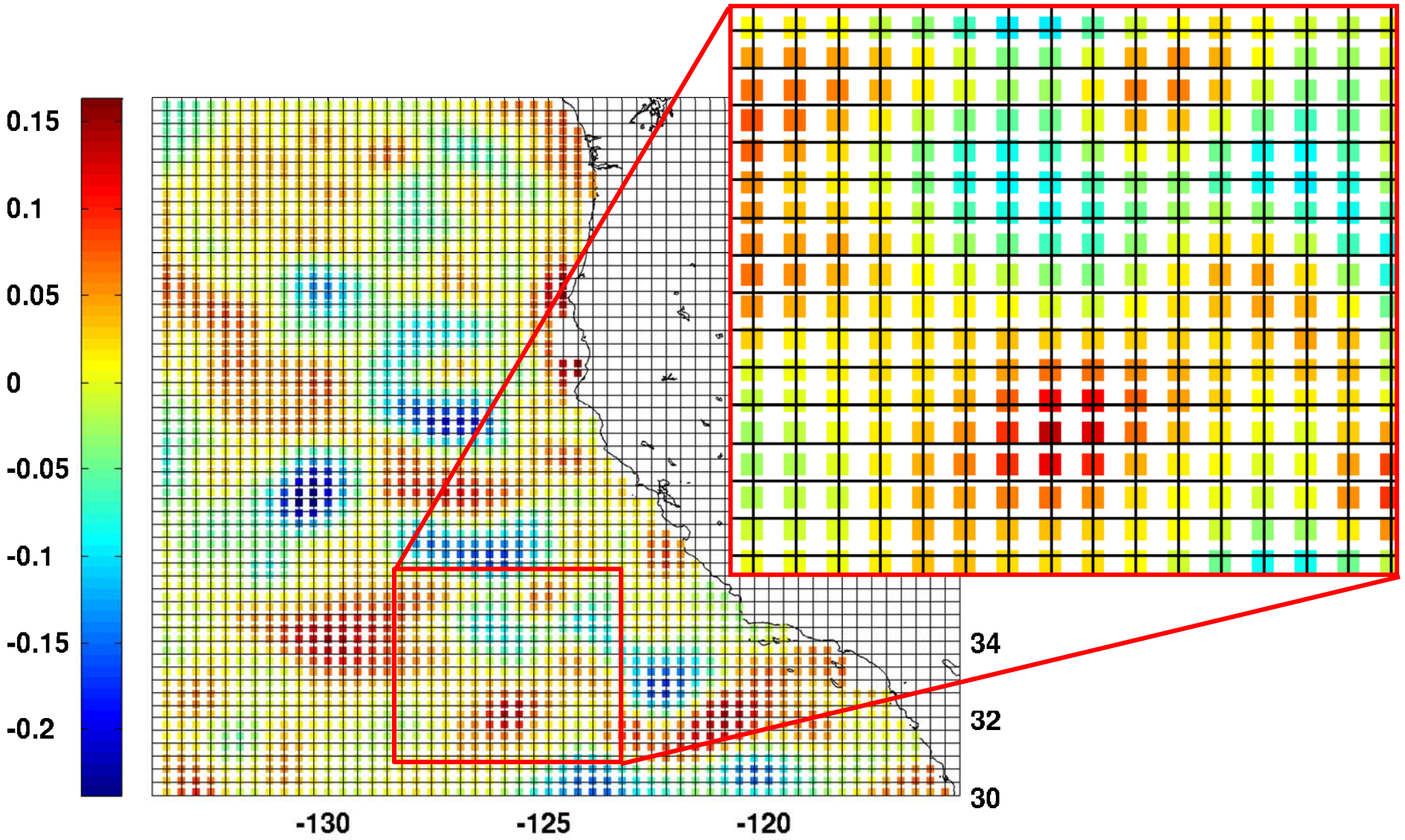
SST Super Observations



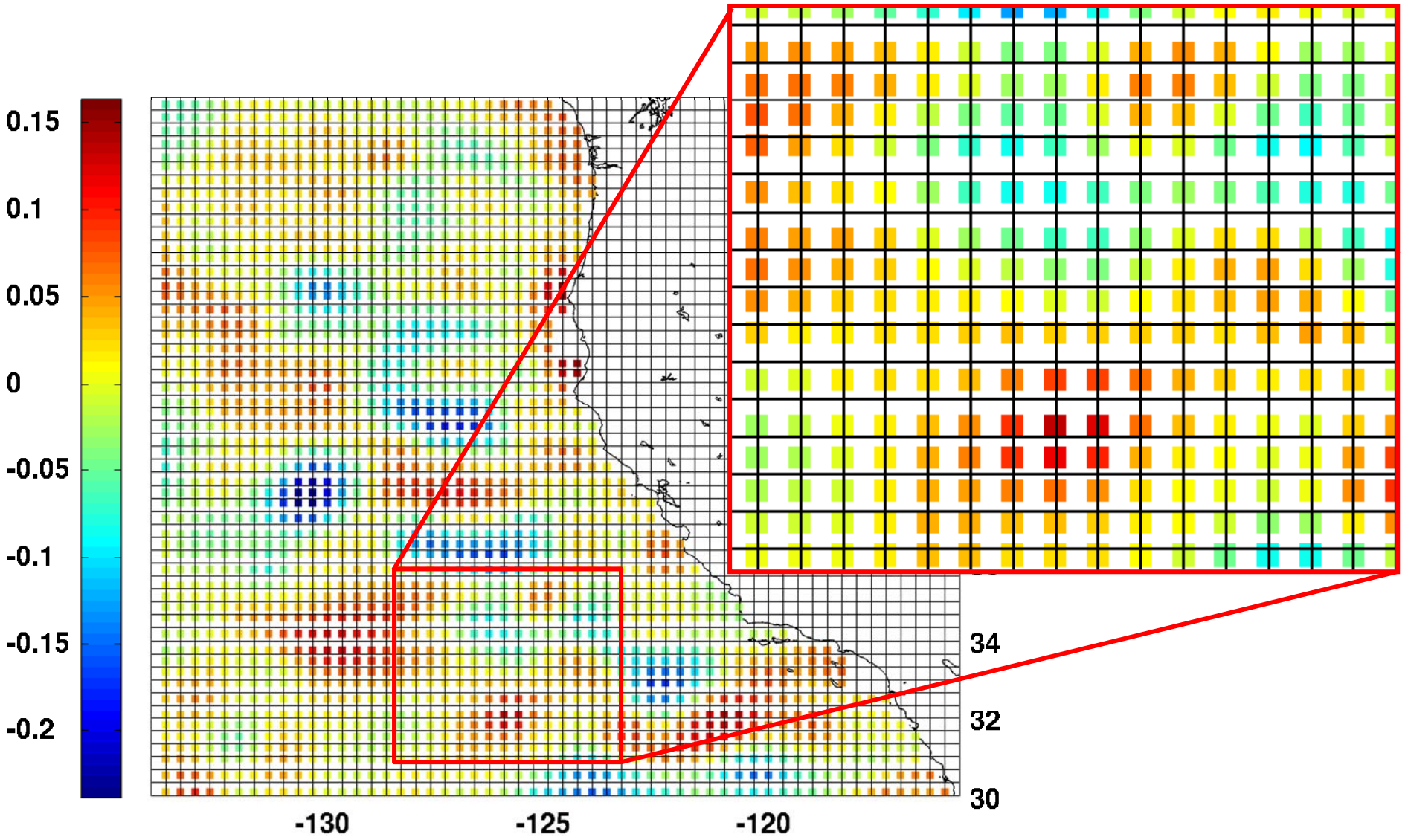
d_ssh_obs.m

```
%
%  D_SSH_OBS:  Driver script to create a 4D-Var SSH observations file.
%
%  This a user modifiable script that can be used to prepare ROMS 4D-Var
%  SSH observations NetCDF file. The SSH observations are extracted from
%  AVISO dataset using script 'load_ssh_aviso.m'. USERS can use this as
%  a prototype for their application.
%

% svn $Id: d_ssh_obs.m 490 2010-07-11 03:58:35Z arango $
%=========================================================================%
%  Copyright (c) 2002-2010 The ROMS/TOMS Group                            %
%    Licensed under a MIT/X style license                                 %
%    See License_ROMS.txt                          Hernan G. Arango       %
%=========================================================================%

%  Set input/output NetCDF files.

 my_root = '/home/arango/ocean/toms/repository/test';

 GRDfile = fullfile(my_root, 'WC13/Data', 'wc13_grd.nc');
 OBSfile = 'wc13_ssh_obs.nc';
 SUPfile = 'wc13_ssh_super_obs.nc';

%  Set ROMS state variable type classification.

Nstate=7;            % number of ROMS state variables

state.zeta = 1;      % free-surface
state.ubar = 2;      % vertically integrated u-momentum
state.vbar = 3;      % vertically integrated v-momentum
state.u    = 4;      % u-momentum
```

SSH Observations (AVISO)



SSH Super Observations



WC13 Observations for Exercises