Tutorial 10:
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: Determines if there are observations available for the current model window, and sets switches and parameters for processing the data.
- Utility/obs_read.F: Reads observations for current survey time for either forward or backward integration, and stores data in the appropriate ROMS variables.
- Utility/obs_write.F: Interpolates model (NLM or TLM) at observation the locations (H operator) when appropriate; sets screening and quality control flag obs_scale, and writes data into output MODname NetCDF file.
- Utility/obs_depth.F: If observations vertical coordinate is in meters (negative values), it initializes internal ROMS variable Zobs to facilitate collective MPI exchanges between tiles.
- Utility/obs_k2z.F: Converts observations vertical coordinate from fractional k-level to depth in meters (negative) for the DART Ensemble Kalman Filter.
- Utility/extract_obs.F: Generic 2D and 3D routines to interpolate observations to the appropriate ROMS NLM or TLM state variable.
- Adjoint/ad_extract_obs.F: Generic adjoint 2D and 3D routines to interpolate observations to the appropriate ROMS ADM state variable.
- Adjoint/ad_htobs.F: Loads adjoint observation operator ($H^T X$) to ADM forcing arrays in the 4D-Var dual formulation.
- Adjoint/ad_misfit.F: Computes the ADM misfit forcing in the 4D-Var primal formulation.

Metadata

Dimensions:

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

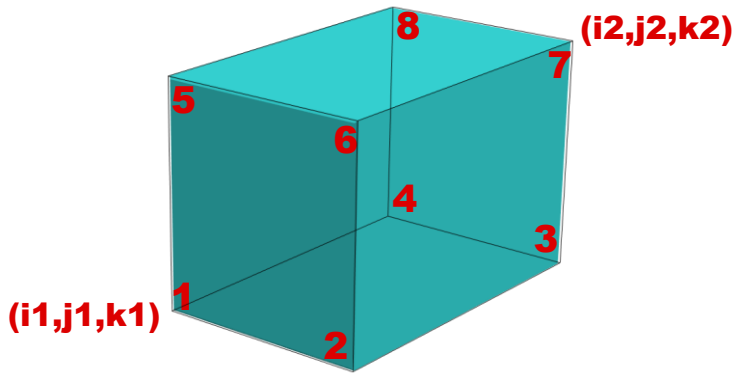
Variables:

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
obs_meta(datum)	Observation meta value

Observations NetCDF

```
dimensions:

        survey = 13 ;
        state_variable = 7 ;
        datum = 6815

        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" ;
        double obs_meta(datum) ;
                obs_value:long_name = "observation meta 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" ;
```

obs_value and obs_meta

```
variables:

. . .

        double obs_value(datum) ;
                obs_value:long_name = "observation value" ;
                obs_value:units = "associated state variable units" ;
        double obs_meta(datum) ;
                obs_meta:long_name = "observation meta value" ;
                obs_meta:units = "associated state variable units" ;
```

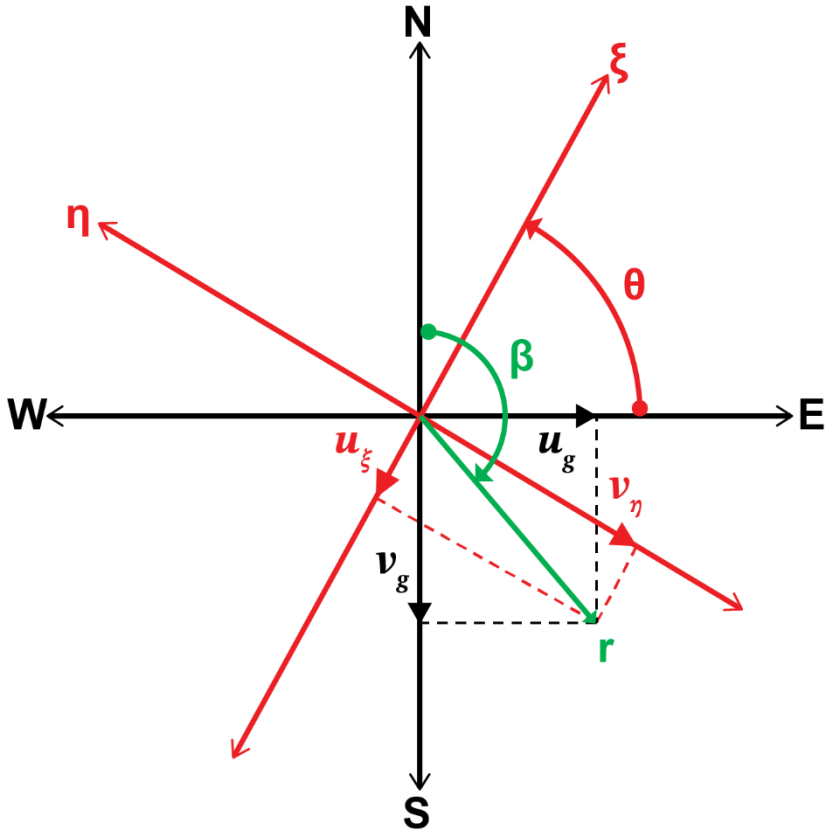
The obs_meta variable contains additional data qualifiers to obs_value for extra-observation operators that are not in a one-to-one correspondence with ROMS state variables.

For Example, in surface HF radial observations:

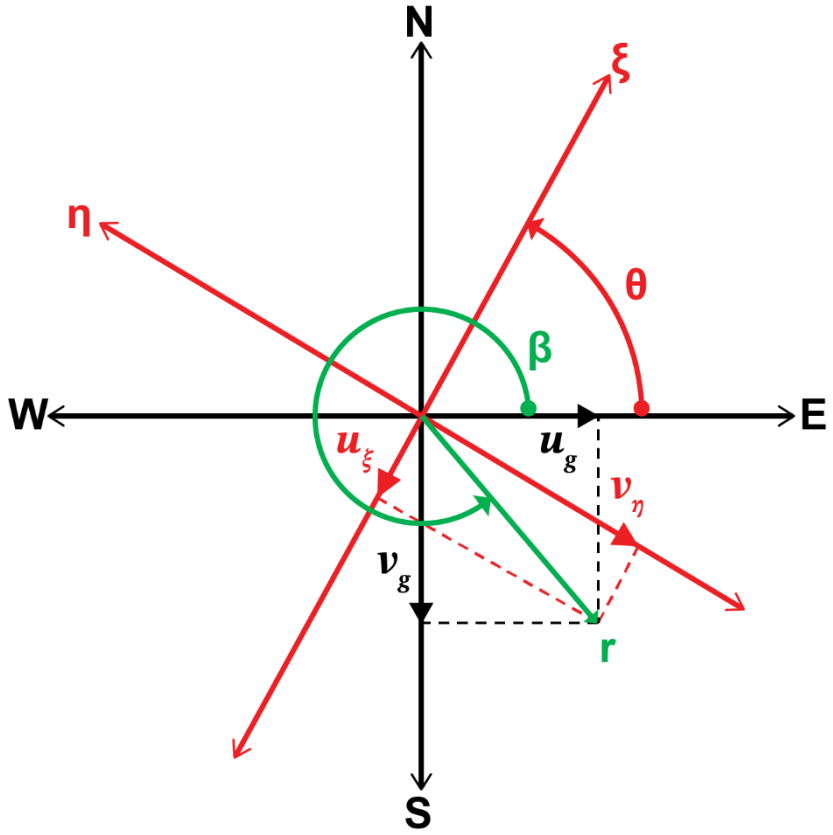
obs_value(:) HF radial speed (m/s)
obs_meta(:) HF radial heading angle (radians)

HF Radar Observations

Navigational Convention
(Default)



Mathematical Convention
(RADIAL_ANGLE_CCW_EAST)



r HF radial speed (obs_value)
β HF radial heading angle (obs_meta)
θ ROMS curvilinear angle (counterclockwise from East)

u_g Geographical zonal velocity
v_g Geographical meridional velocity
u_ξ ROMS radial ξ-velocity component
v_η ROMS radial η-velocity component

ROMS Observation Classes

- ROMS State Variables Operator, one-to-one correspondence with observation data:
  - Reserved enumeration indices, 1 to NstateVar where NstateVar = 7+2*NT and NT is the number of active and passive tracers. It includes zeta, ubar, vbar, u, v, t(1:NT), sustr, svstr, and stflx(1:NT).
- ROMS Extra-Observation Operators, multiple ROMS state variables dependency:
  - HF radials observations: u and v (available).
  - Pressure observations.
  - Travel time observations: temperature over large distances (acoustic tomography).
- In any ROMS application, the number of observation classes is:
  - NobsVar = NstateVar + NextraObs
- In s4dvar.in, we have:

```
!  NextraObs values are expected for keywords ExtraIndex and ExtraName.
!  If NextraVar > 1, enter one observation type names per line and
!  use a backslash as the continuation.

      NextraObs = 0
     ExtraIndex = 20, 30
      ExtraName = radial \
                  pressure
```

- ROMS internal mapping indices:
  - ObsState2Type: state variable index to observation type (obs_type)
  - ObsState2Type(isTvar(itemp)) = 6
  - ObsType2State: observation type (obs_type) to state variable index
  - ivar = ObsType2State(obs_type(iobs))

Observation Provenance

The observation provenance is an integer ID indicating the source of each datum, and it can be used to quantify impact and sensitivity of each observation in the 4D-Var analysis. The enumeration is arbitrary, and the user needs to be consistent with the provenance codes when processing the data.

For MARACOOS, the provenance codes range from 300-800:

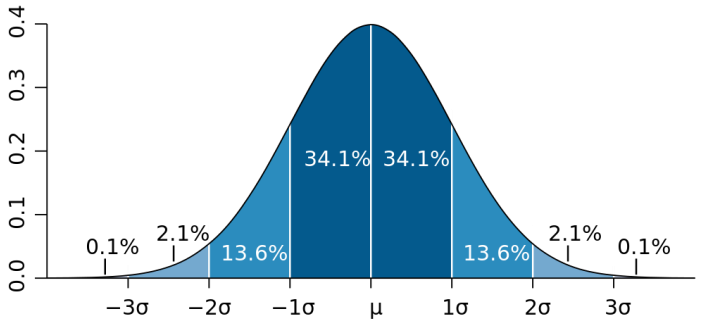
ALTIKA_SSH	426
Altimeter_superobs	400
AMSR_microwave_SST	306
AVHRR_IR_SST	302
CODAR_velocities	500
Cryosat_SSH	424
ECOMON_CTD	730
Envisat_SSH	423
GOES_geostationary_IR_SST	303
IOOS_gliders	702
Jason_1_SSH	401
Jason_2_SSH	402
Jason_3_SSH	403
MetOffice_Argo	722
MetOffice_Bouys	723
MetOffice_CTD_XCTD_TESAC	721
MetOffice_XBT	720
NERACOOS_CTD	740
OSMC_GTS_floats	713
OSMC_GTS_gliders	711
OSMC_GTS_moored_bouys	716
OSMC_GTS_other	710
OSMC_GTS_ships_generic	717
OSMC_GTS_VOShips	714
Pioneer_Array_CP01CNSM	601
Pioneer_Array_CP01CNSP	602
Pioneer_Array_CP02PMCI	603
Pioneer_Array_CP02PMCO	606
Pioneer_Array_CP02PMUI	605
Pioneer_Array_CP02PMUO	610
Pioneer_Array_CP03ISSM	607
Pioneer_Array_CP03ISSP	608
Pioneer_Array_CP04OSPM	609
Pioneer_Array_CP04OSSM	611
Pioneer_Array_GL003	641
Pioneer_Array_GL335	642
Pioneer_Array_GL336	643
Pioneer_Array_GL339	644
Pioneer_Array_GL340	645
Pioneer_Array_GL374	646
Pioneer_Array_GL376	647
Pioneer_Array_GL379	648
Pioneer_Array_GL380	649
Pioneer_Array_GL387	650
Pioneer_Array_GL388	651
Pioneer_Array_GL389	652
Pioneer_Array_GL390	653
Pioneer_Array_PG564	654
RU_MARACOOS_gliders	701
Sentinel_3a_SSH	441
Sentinel_3b_SSH	442
SST_super_observations	300
TRMM_microwave_SST	304
VIIRS_IR_SST	307
WSAT_microwave_SST	305

Background Quality Control

Use CPP option BGQC to activate a background quality control of the observations in terms of ROMS state variable index or provenance.

In s4dvar.in we have:

```
! Select flag for BackGround Quality Control (BGQC) of observations:
!
!      [1] Quality control in terms of state variable indices
!      [2] Quality control in terms of observation provenance

      bgqc_type

! If BGQC is in terms of state variables, set number of standard deviations
! threshold to use in the quality control rejection of observations.
!
!   Use a large value (say, 1.0d+5) if you do not want to reject observations
!   associated with a particular state variable.
!
!   Use a small value (typically, 4 standard deviations) to perform quality
!   control of observations for a particular state variable.

 S_bgqc(isFsur)                                          ! free-surface
 S_bgqc(isUbar)                                          ! 2D U-momentum
 S_bgqc(isVbar)                                          ! 2D V-momentum
 S_bgqc(isUvel)                                          ! 3D U-momentum
 S_bgqc(isVvel)                                          ! 3D V-momentum
 S_bgqc(isTvar)                                          ! 1:NT tracers

! If BGQC is in terms of observation provenance, set number of standard
! deviations threshold to use in the quality control rejection of
! observations.
!
!   Use a small value (say, 4 standard deviations) to perform quality
!   control for the desired observation provenance(s).
!
!   Nprovenance:   Number of observation provenances to quality control
!   Iprovenance:   Observation provenance indices to process [1:Nprovenance]
!   P_bgqc:        Standard deviation threshold [1:Nprovenance]

    Nprovenance
    Iprovenance                          ! ARGO Temperature and Salinity

         P_bgqc
```



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. Their values range are:

  obs_Xgrid: 0.5 to L – 0.5
  obs_Ygrid: 0.5 to M – 0.5

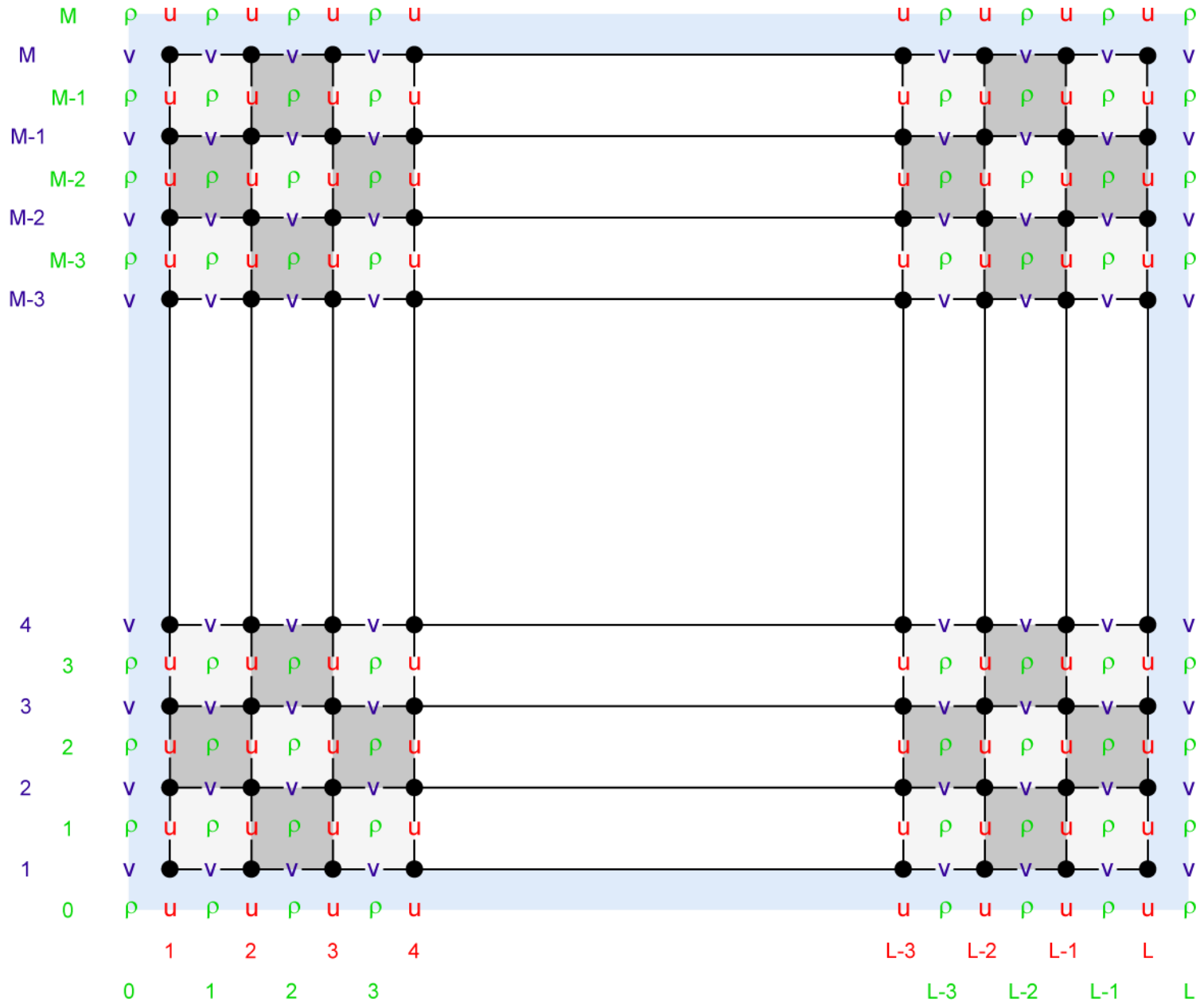
- However, they are useful during post-processing and plotting results into maps or when changing the application grid.

- If land/sea masking, observations located on land grid cell are rejected with ObsScale(iobs) = 0.0. Viable observations have a value of ObsScale(iobs) = 1.0.

ROMS GRID

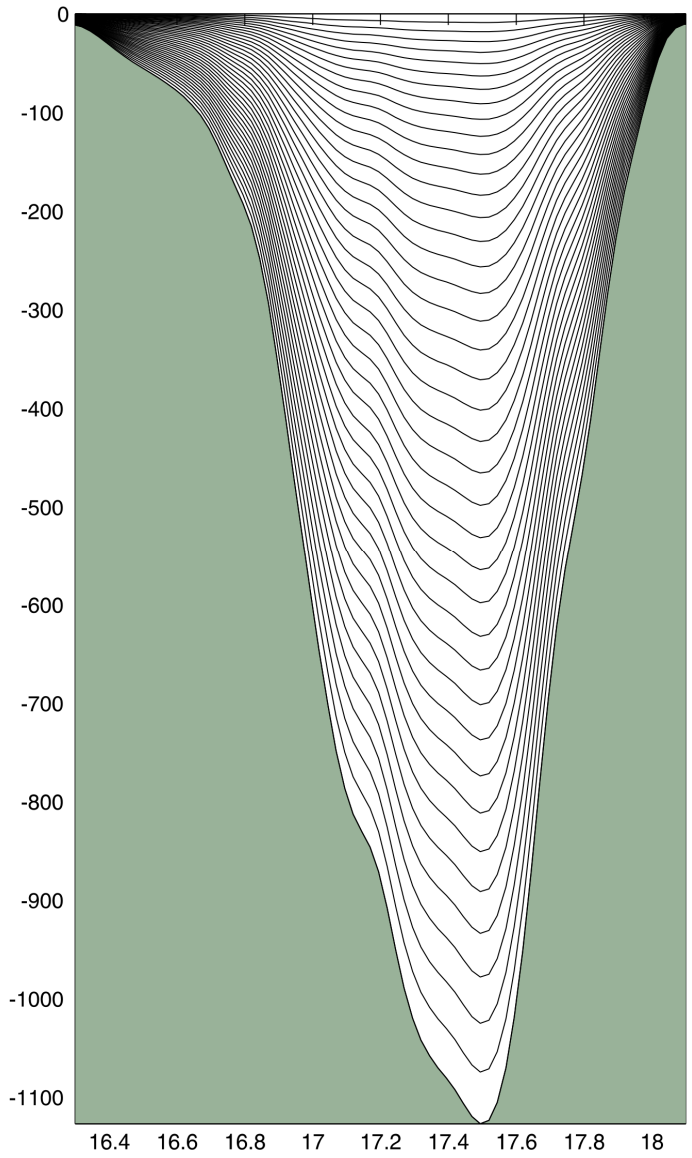
- Horizontal curvilinear orthogonal coordinates on an Arakawa C-grid.
- Terrain-following coordinates on a staggered vertical grid.
- In 4D-Var, all the observations locations (obs_Xgrid, obs_Ygrid) are in the fractional (i, j, k) coordinates to facilitate efficient spatial interpolations.
- Coarse-grained parallelization with horizontal 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

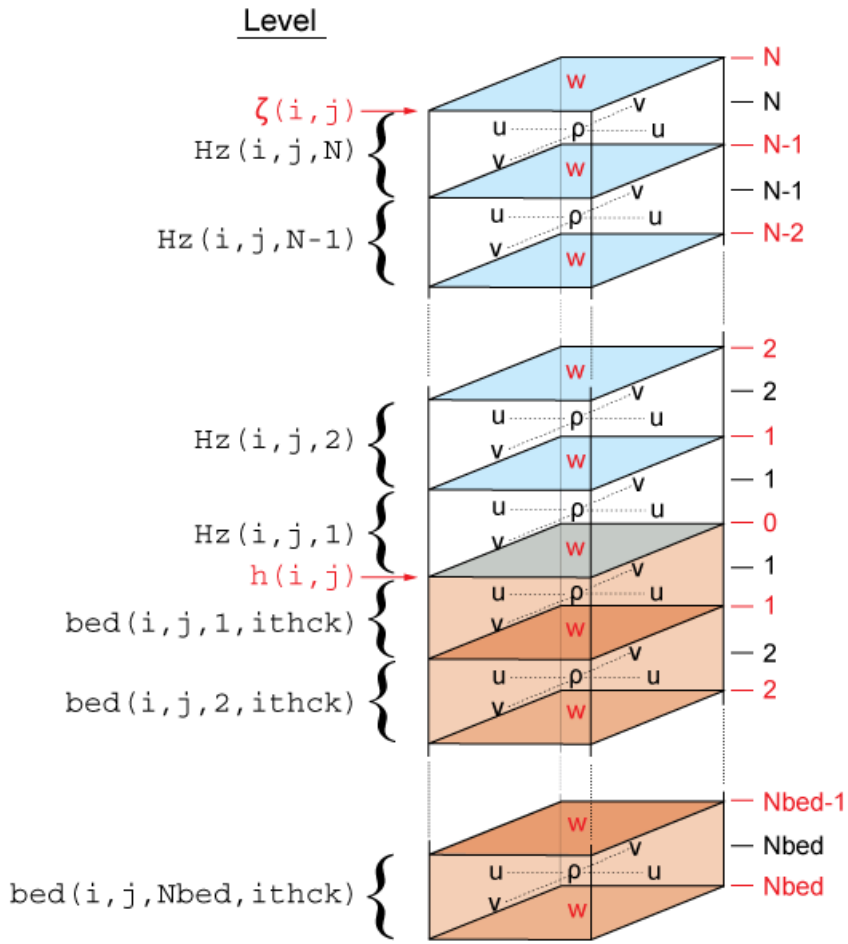
Vieste (Italy)



Depth (m)

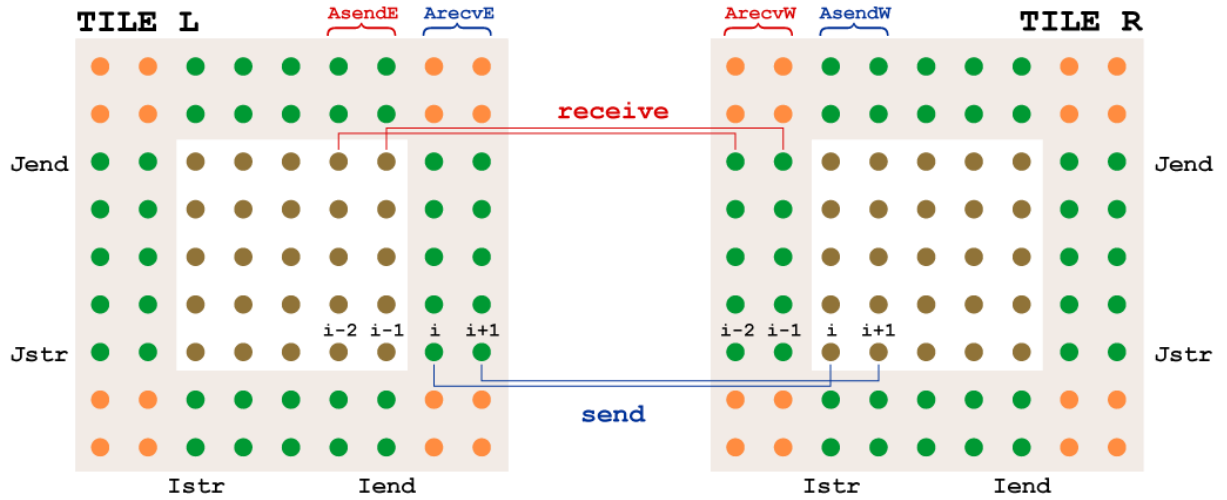
Longitude

Dubrovnik (Croatia)



East-West MPI Communications

Nonlinear

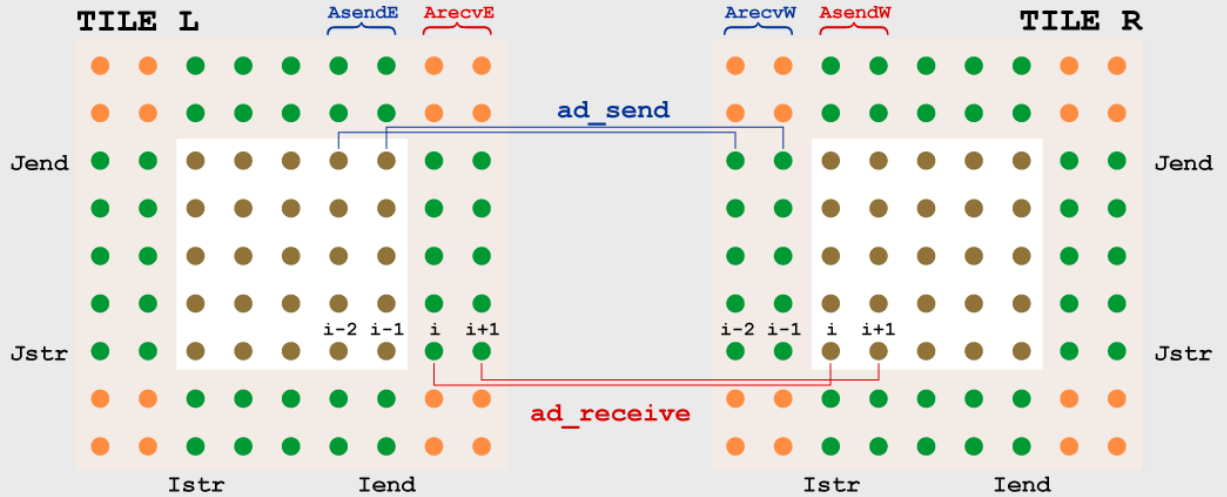


$$v^L_i = v^R_i \qquad v^R_{i-2} = v^L_{i-2}$$

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

With Respect To Tile R

Adjoint



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

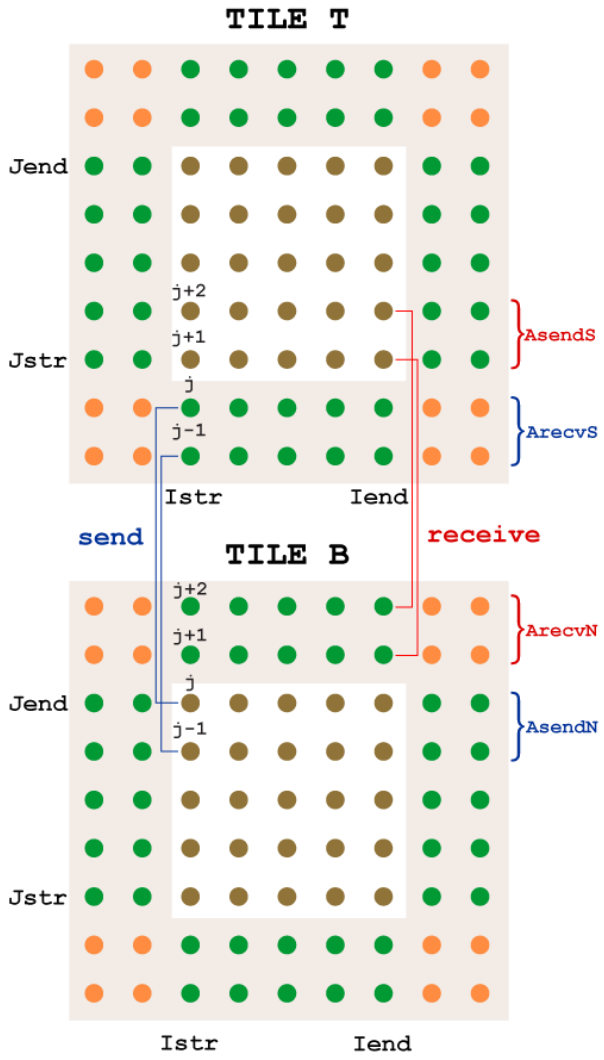
$$ad\_v^L_{i-1} = ad\_v^L_{i-1} + ad\_v^R_{i-1} \;;\; v^R_{i-1} = 0 \qquad ad\_v^R_{i+1} = ad\_v^R_{i+1} + ad\_v^L_{i+1} \;;\; v^L_{i+1} = 0$$

With Respect To Tile R

North-South MPI Communications

Nonlinear

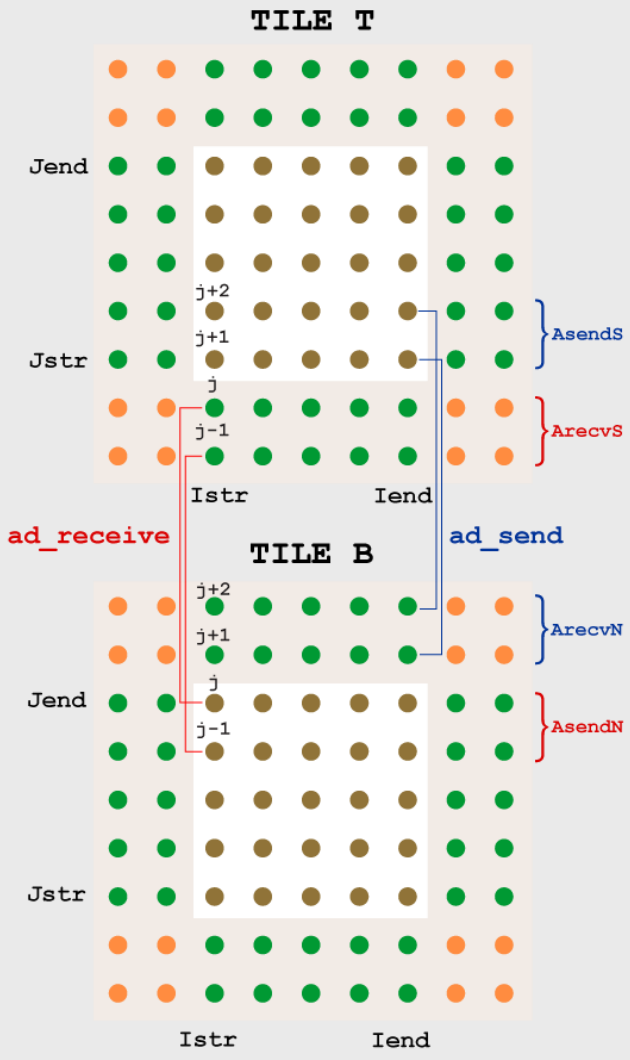
$$V^B_{j+2} = V^T_{j+2}$$
$$V^B_{j+1} = V^T_{j+1}$$



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

With Respect to Tile B

$$ad\_V^T_{j+2} = ad\_V^T_{j+2} + ad\_V^B_{j+2} \; ; \; ad\_V^B_{j+2} = 0$$
$$ad\_V^T_{j+1} = ad\_V^T_{j+1} + ad\_V^B_{j+1} \; ; \; ad\_V^B_{j+1} = 0$$



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

Adjoint

Processing

- Compute observation's fractional grid coordinates (obs_Xgrid, obs_Ygrid) locations from its (obs_lon, obs_lat).
- Find how many different survey times occur within the data set. Assign such value to the survey dimension, and store unique time values in survey_time(1:survey).
- Count the number of observations available per each survey, and assign their values to Nobs(1:survey).
- Sort the observation in ascending time order to facilitate processing in the forward integration of ROMS NLM and TLM kernels and backward integration of ROMS ADM kernel.
- In addition to sorting in time, it is recommended to sort observations in ascending obs_type index for efficiency. That is, to process chunks of data in continuous computer memory.
- Save a copy of the observation file since ROMS modifies that NetCDF file.
- There are several Matlab scripts available to process the 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
S.Meta	observation meta 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, ...
                                  Correction, obc_edge, ...
                                  Ioffset, Joffset);

%
% OBS_IJPOS:  Computes observation locations in ROMS fractional coordinates
%
% [Xgrid,Ygrid]=obs_ijpos(GRDname,obs_lon,obs_lat,Correction,obc_edge,
%                         Ioffset,Joffset)
%
% 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.  If the Ioffset and Joffset
% vectors are provided, the polygon defined by the application grid is
% smaller by the number of grid points provided in the offset.  This is
% done to avoid processing next to the applications boundary edges.
%
% On Input:
%
%    GRDname       NetCDF grid file name (string)
%    obs_lon       observation longitude (positive, degrees_east)
%    obs_lat       observation latitude  (positive, degrees_north)
%    Correction    switch to apply correction due to spherical/curvilinear
%                    grids (false, true)
%    obc_edge      switch to include observations on open boundary edges
%                    (false, true)
%    Ioffset       Application I-grid offset when defining polygon (vector):
%                    Ioffset(1):  I-grid offset on the edge where Istr=1
%                    Ioffset(2):  I-grid offset on the edge where Iend=Lm
```

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 when there are more than one meassurement of
% the same state variable per grid cell. At input, Sinp is either a
% 4D-Var observation NetCDF file or data structure.
%
% An additional field (Sout.std) is added to the output observation
% structure containing the standard deviation of the binning which
% can be used as observation error. You may choose to assign this
% value as observation error before writing to NetCDF file:
%
%    Sout.error = max(Sout.error, Sout.std)
%
% That is, the larger observation variace is chosen. A zero value
% for "Sout.std" indicates that the observation did not required
% binning.  Use "c_observations.m" to create NetCDF file and
% "obs_write" to write out data.
%
% On Input:
%
%    Sinp    Observations data structure or NetCDF file name
%
%
% On Output:
```

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 (i,j) indices.
- Vertical locations can be in fractional levels (1:N) or actual depths (negative values).
- The observation error usually includes:
  - Instrument error (uncorrelated)
  - Error of representation (true ocean state versus model discrete state)
  - Observation processing error: thinning, averaging, filtering, and interpolation
  - Observation operator error
- 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
- Screening and background quality control (BGQC) inside ROMS via the ObsScale factor (0: reject, 1: accept)
- 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 rejected. It is advantageous to provide observations vertical position in fractional coordinates (1:N). Assign level N to satellite observations.
- If deeper than z_r(:,:,1), observations are rejected.

Recommendations

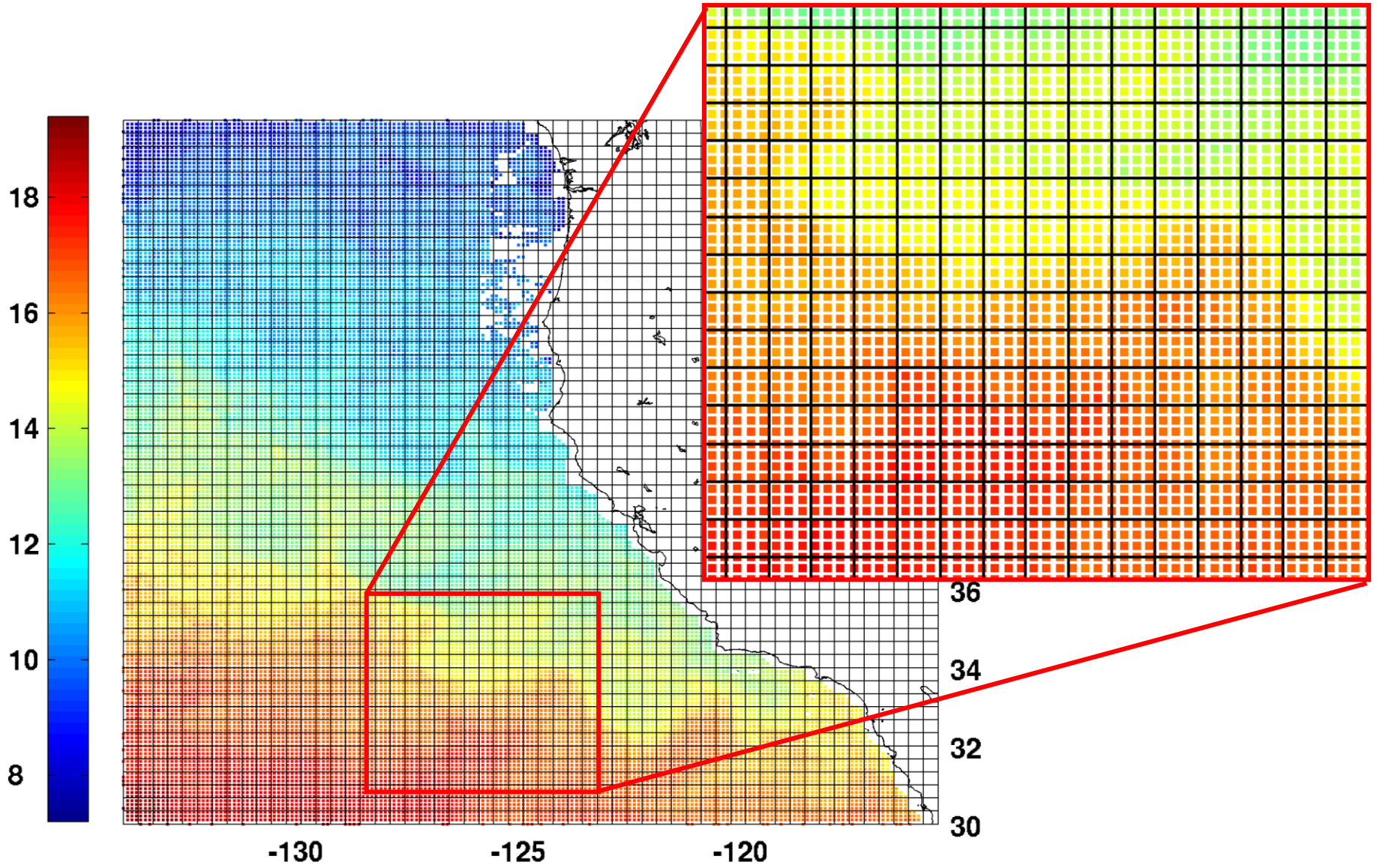
- Create a NetCDF file for each observation type.
- Use a processing program to meld NetCDF observation files (obs_merge.m).
- Keep a master copy of each observation file, in case you are running your application at different resolutions.
- Decimation of observations. Finite volume representation super 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 Global
%  using script 'load_sst_pfeg.m'.  USERS can use this as a prototype
%  for their application.
%

% svn $Id: d_sst_obs.m 938 2019-01-28 06:35:10Z arango $
%=========================================================================%
%  Copyright (c) 2002-2019 The ROMS/TOMS Group                            %
%    Licensed under a MIT/X style license                                 %
%    See License_ROMS.txt                           Hernan G. Arango      %
%=========================================================================%

%  Set SST product to use from CoastWatch OpenDAP server:

CoastWatch = 'http://oceanwatch.pfeg.noaa.gov/thredds/dodsC/satellite';

%  sst_URL = strcat(CoastWatch, '/AA/ssta/1day');     % SST, Aqua AMSR-E,
%  sst_URL = strcat(CoastWatch, '/AA/ssta/3day');     % Global
%  sst_URL = strcat(CoastWatch, '/AA/ssta/5day');
%  sst_URL = strcat(CoastWatch, '/AA/ssta/8day');
%  sst_URL = strcat(CoastWatch, '/AA/ssta/14day');
%  sst_URL = strcat(CoastWatch, '/AA/ssta/mday');

   sst_URL = strcat(CoastWatch, '/BA/ssta/5day');     % SST, Blended, Global
%  sst_URL = strcat(CoastWatch, '/BA/ssta/8day');     % Experimental
```

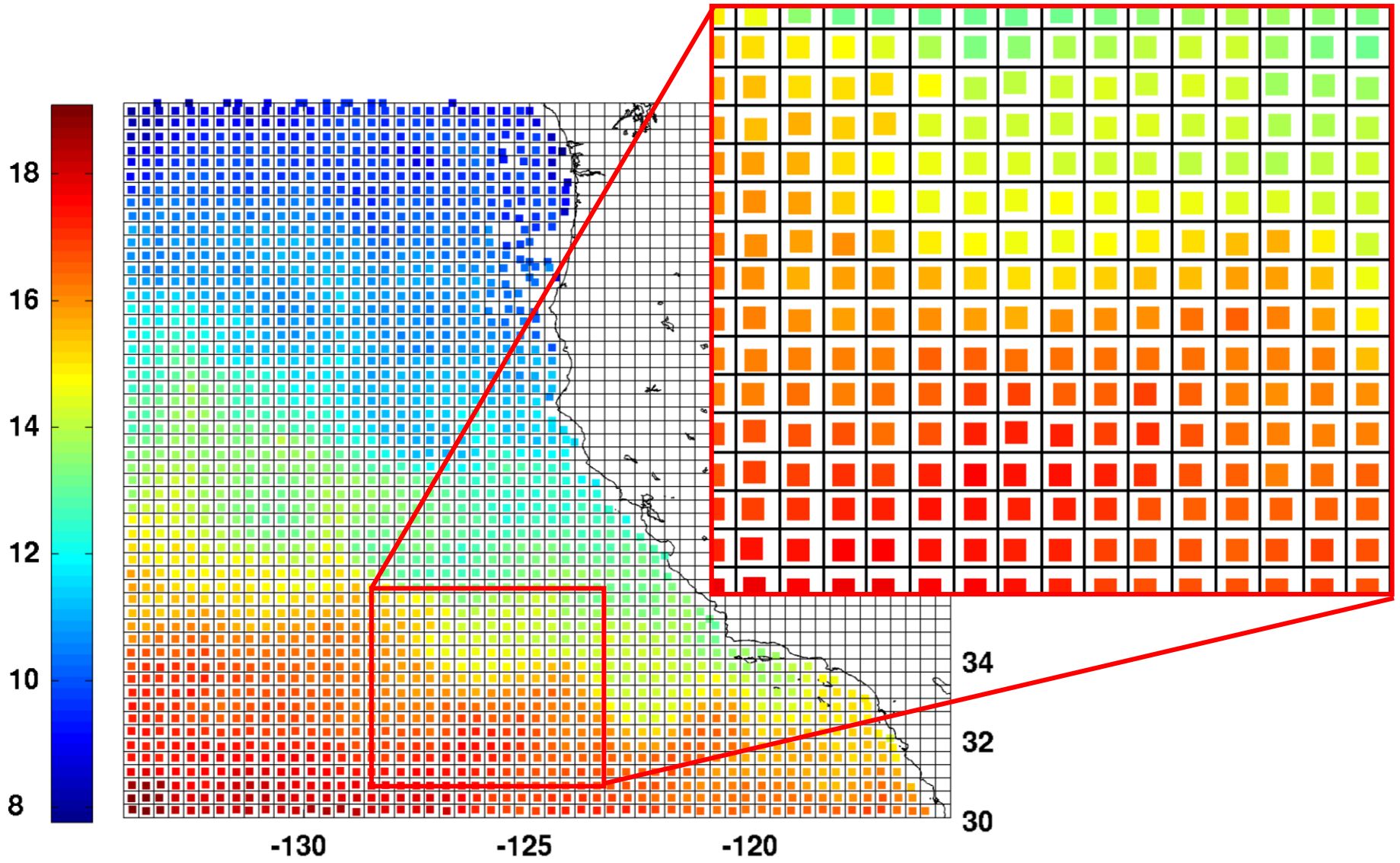
SST Observations (1/10 degree)

Jan 1, 2004



SST Super Observations

Jan 1, 2004

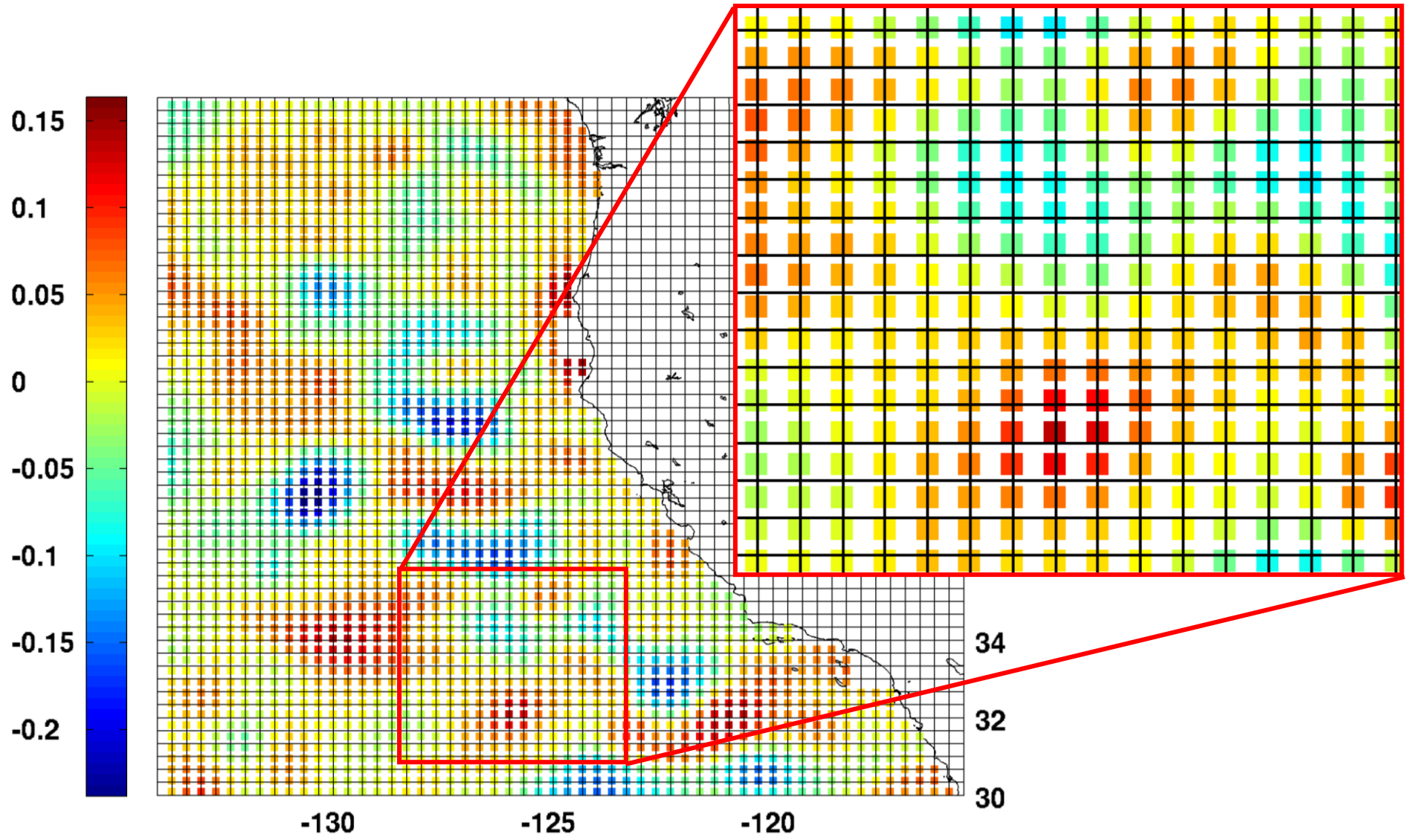


d_ssh_obs.m

```
%
%  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.
%
%  The AVISO dataset is password protected and users need to request
%  access by filling the following form from:
%
%  http://www.aviso.oceanobs.com/en/data/registration-form
%

% svn $Id: d_ssh_obs.m 938 2019-01-28 06:35:10Z arango $
%=========================================================================%
%  Copyright (c) 2002-2019 The ROMS/TOMS Group                            %
%    Licensed under a MIT/X style license                                 %
%    See License_ROMS.txt                           Hernan G. Arango      %
%=========================================================================%

%  Set SSH product to use from AVISO OpenDAP server.  You need to edit
%  the USERNAME and PASSWORD in the link below:

% Dir = 'http://USERNAME:PASSWORD@opendap.aviso.oceanobs.com/thredds/dodsC';

% ssh_URL = strcat(Dir, '/dataset-duacs-nrt-over30d-global-merged-msla-h');
  ssh_URL = strcat(Dir, '/dataset-duacs-dt-ref-global-merged-msla-h-daily');
% ssh_URL = strcat(Dir, '/dataset-duacs-dt-upd-global-merged-msla-h-daily');
```

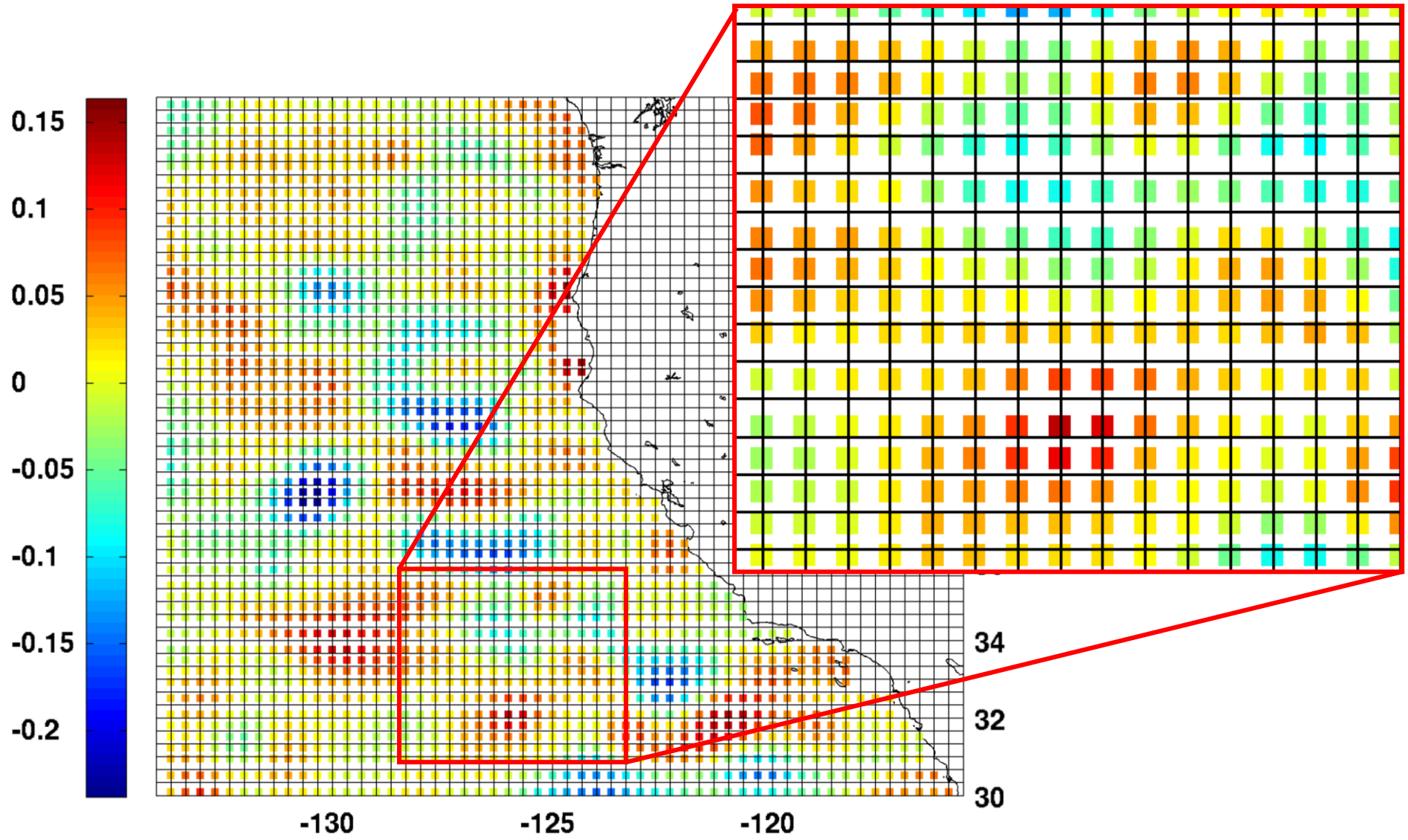
SSH Observations (AVISO)

Jan 1, 2004

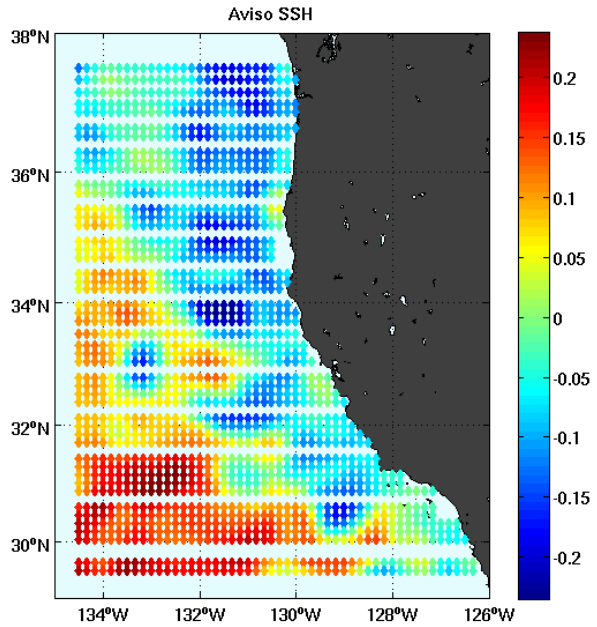


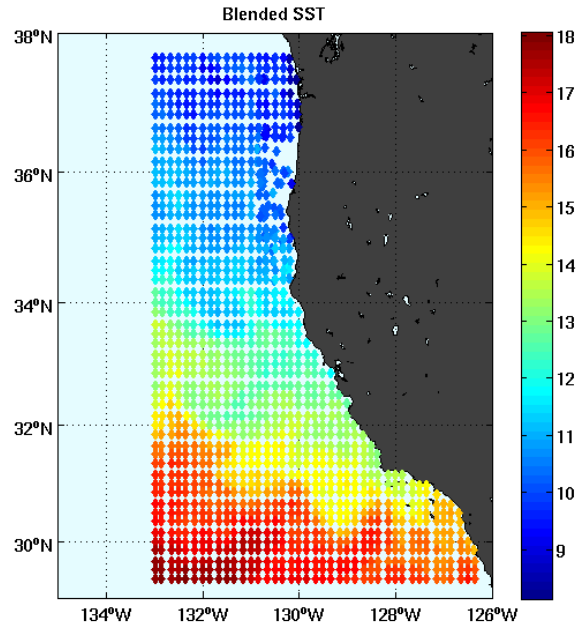
SSH Super Observations

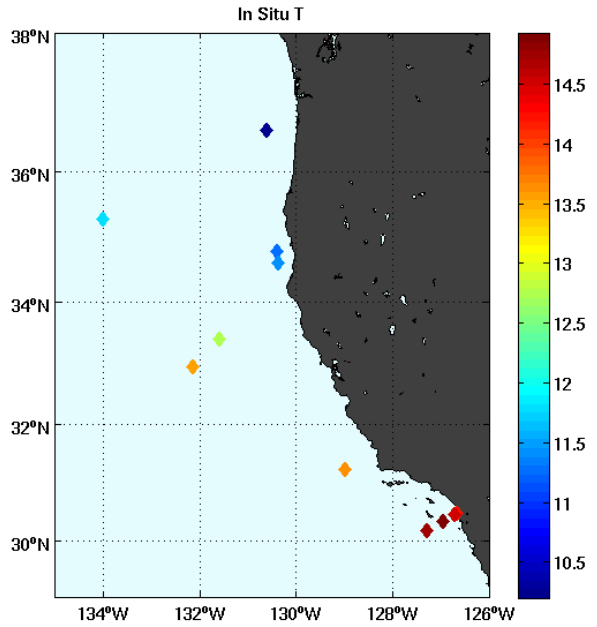
Jan 1, 2004

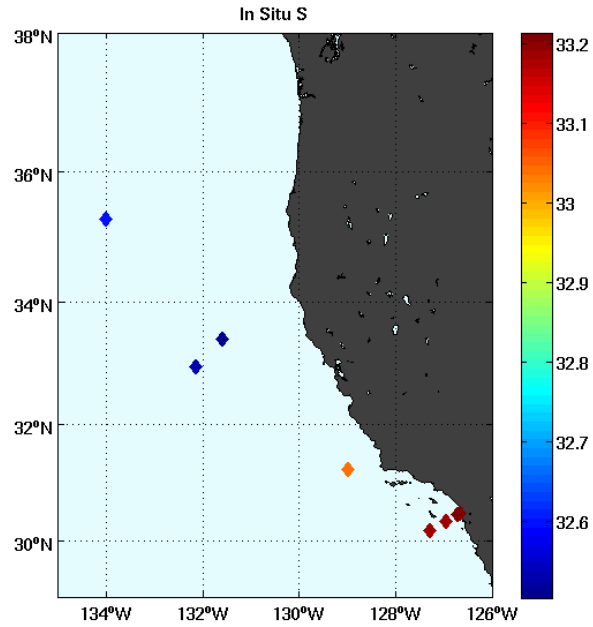


WC13 Observations for Exercises









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