Some ROMS Algorithms

Hernan G. Arango
DMCS, Rutgers University

COAWST Modeling System Training
WHOI, Woods Hole, MA
August 16, 2016

ROMS VIP'S

Andrew M. Moore
U. California Santa Cruz
Adjoint-Based Algorithms

Alexander F. Shchepetkin
U. California Los Angeles
Nonlinear Kernel

Kate S. Hedstrom
U. Alaska, Fairbanks
User Community Forum

John C. Warner
USGS, WHOI
Sediment Transport, Nesting
COAWST

John L. Wilkin
IMCS, Rutgers University
Supreme Beta Tester

David J. Robertson
IMCS, Rutgers University
Cyber Infrastructure

Latest Releases

- **Revised wetting and drying algorithm:**
  www.myroms.org/projects/src/ticket/648
- **Replaced the SPLINES option:**
  www.myroms.org/projects/src/ticket/681
- **Corrected tracer horizontal diffusion algorithms:**
  www.myroms.org/projects/src/ticket/689
- **Added Red Tide Ecosystem Model (Gulf of Maine):**
  www.myroms.org/projects/src/ticket/694
- **Added Ensemble Kalman filter (EnKF) using the Data Assimilation Research Testbed (DART) developed at NCAR:**
  www.myroms.org/projects/src/ticket/697
- **Added Staggered Grid (SGRID) data model conventions:**
  www.myroms.org/projects/src/ticket/701
- **Major Update to all 4D-Var algorithms:**
  www.myroms.org/projects/src/ticket/702
- **Added Quicksave output NetCDF file:**
  www.myroms.org/projects/src/ticket/704

SPLINES Option

The SPLINES option was removed and replaced with following three options for more flexibility:

- SPLINES_VDIFF: conservative, parabolic splines reconstruction for vertical diffusion on active and passive tracers (step3d_t.f).
- SPLINES_VVISC: conservative, parabolic splines reconstruction for vertical diffusion on active and passive tracers (step3d_uv.f).
- RI_SPLINES: conservative, parabolic splines reconstruction for vertical velocity shear used in the Richardson Number (gls_corstep.F and my25_corstep.F) and Bulk Richardson Number (lmd_bkpp.F, lmd_skpp.F, and lmd_vmis.F).

It was been reported that the SPLINES option violates the stress condition:

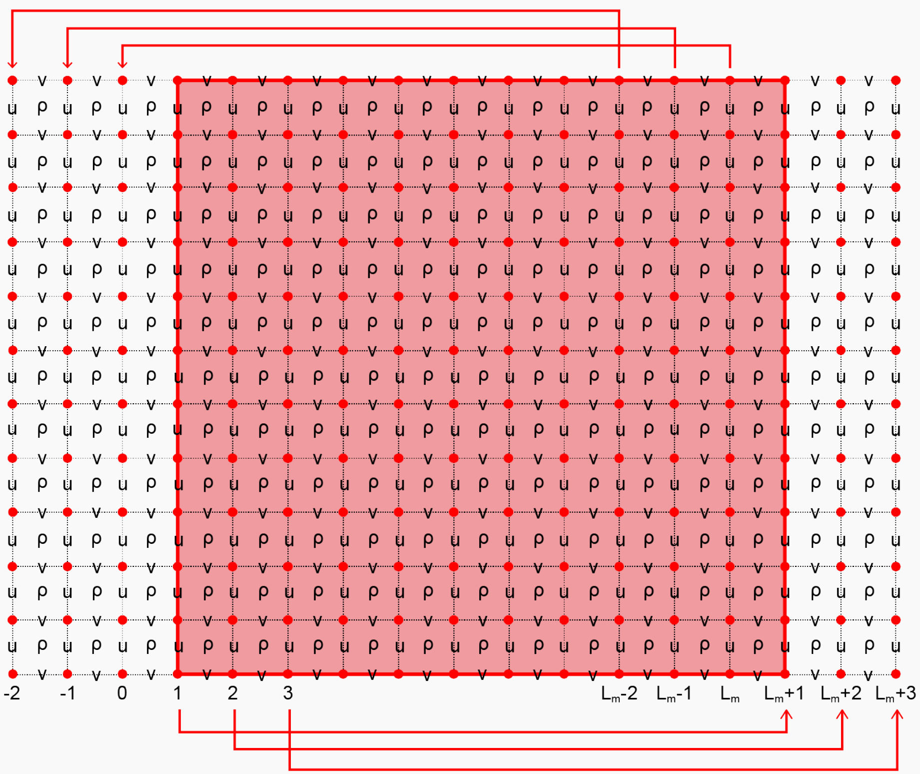
sustr = Akv * du/dz

svstr = Akv * dv/dz

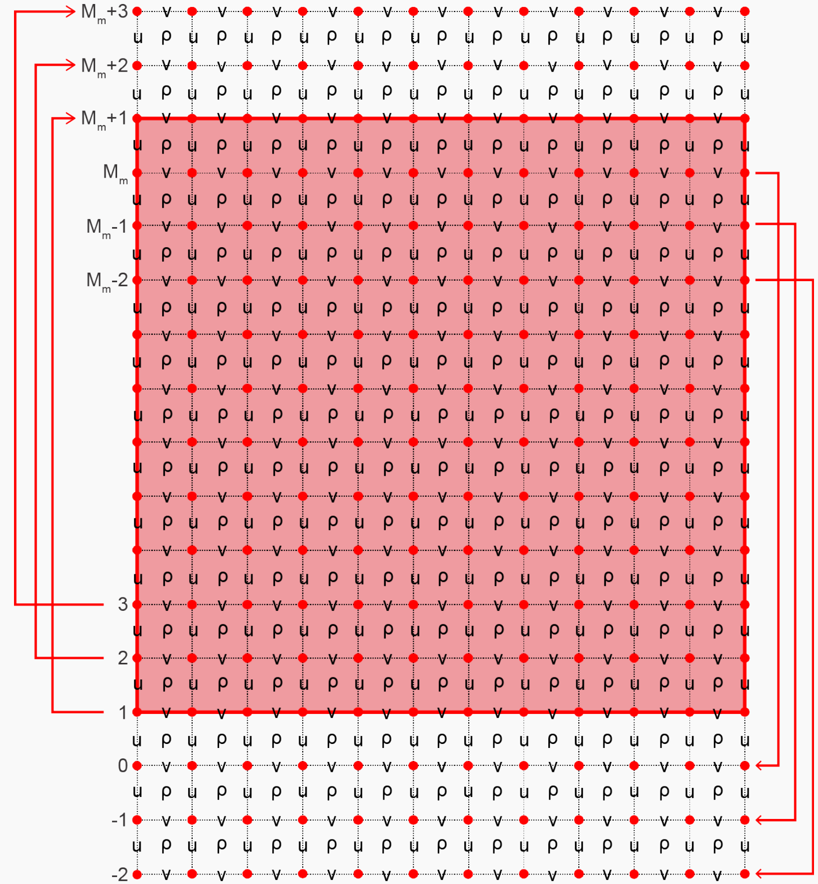
Check https://www.myroms.org/projects/src/ticket/681

ROMS Nesting is very unique: Inspiration

Nesting Inspiration

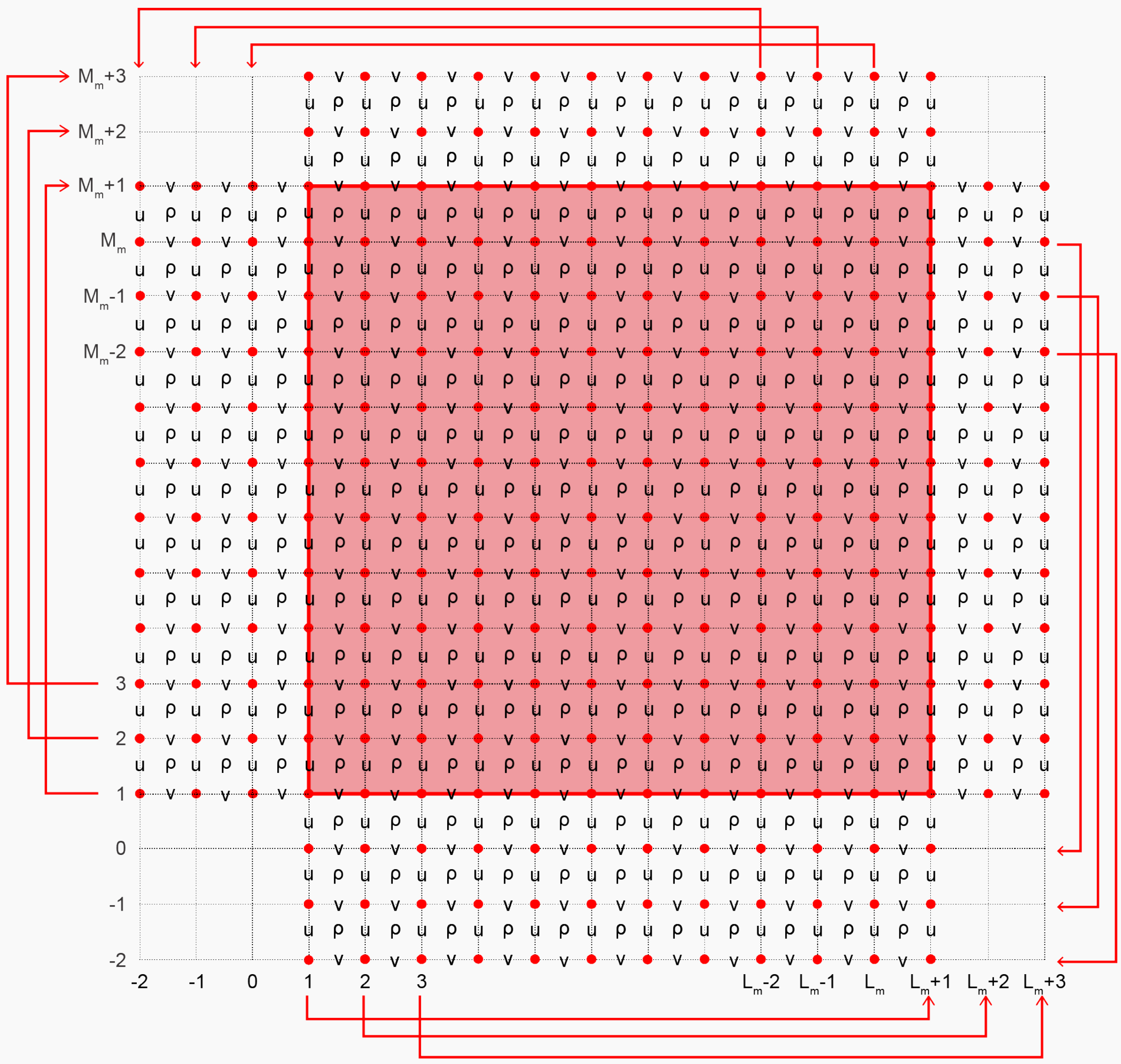


East-West Periodic



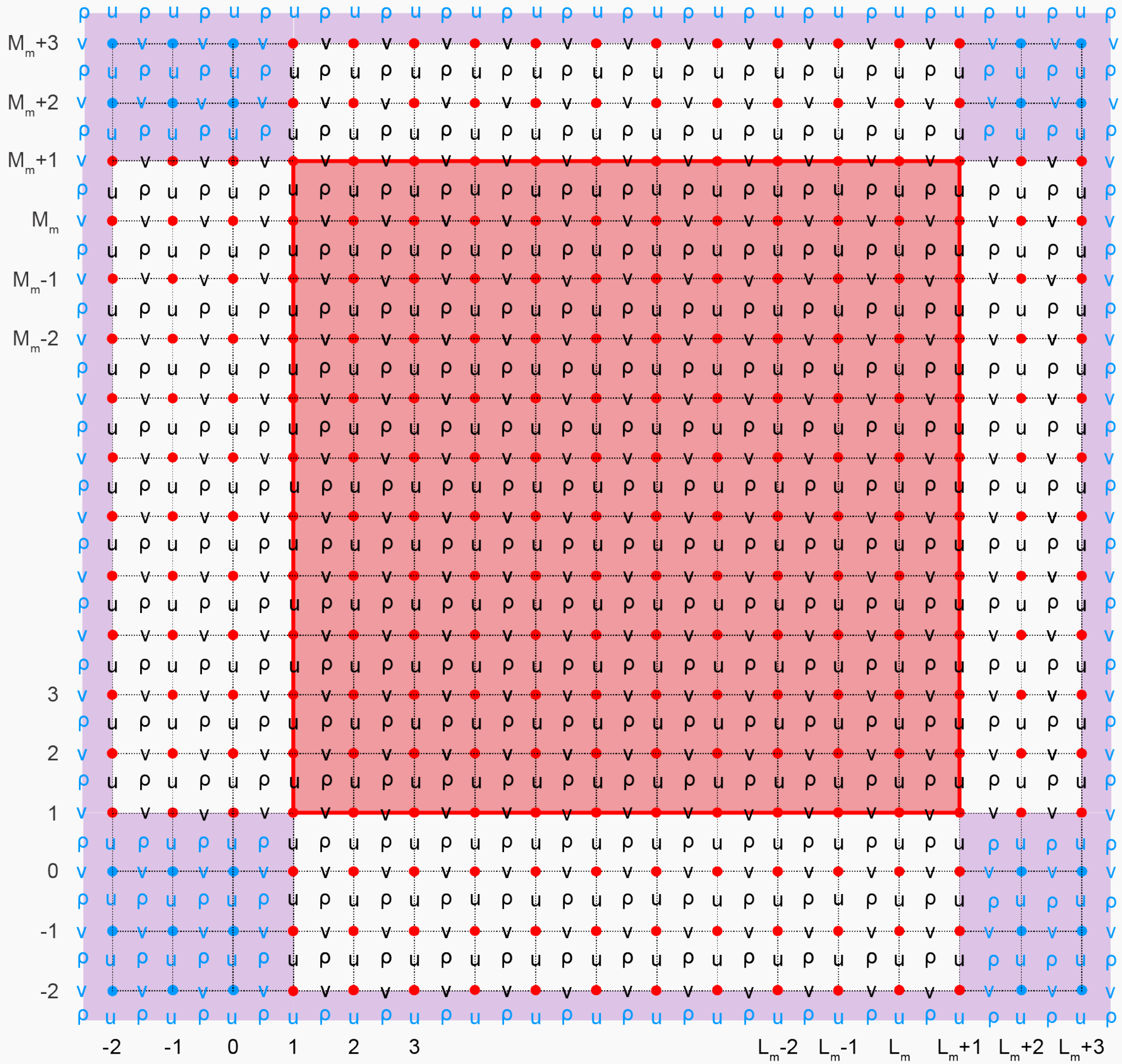
North-South Periodic

Nesting Inspiration



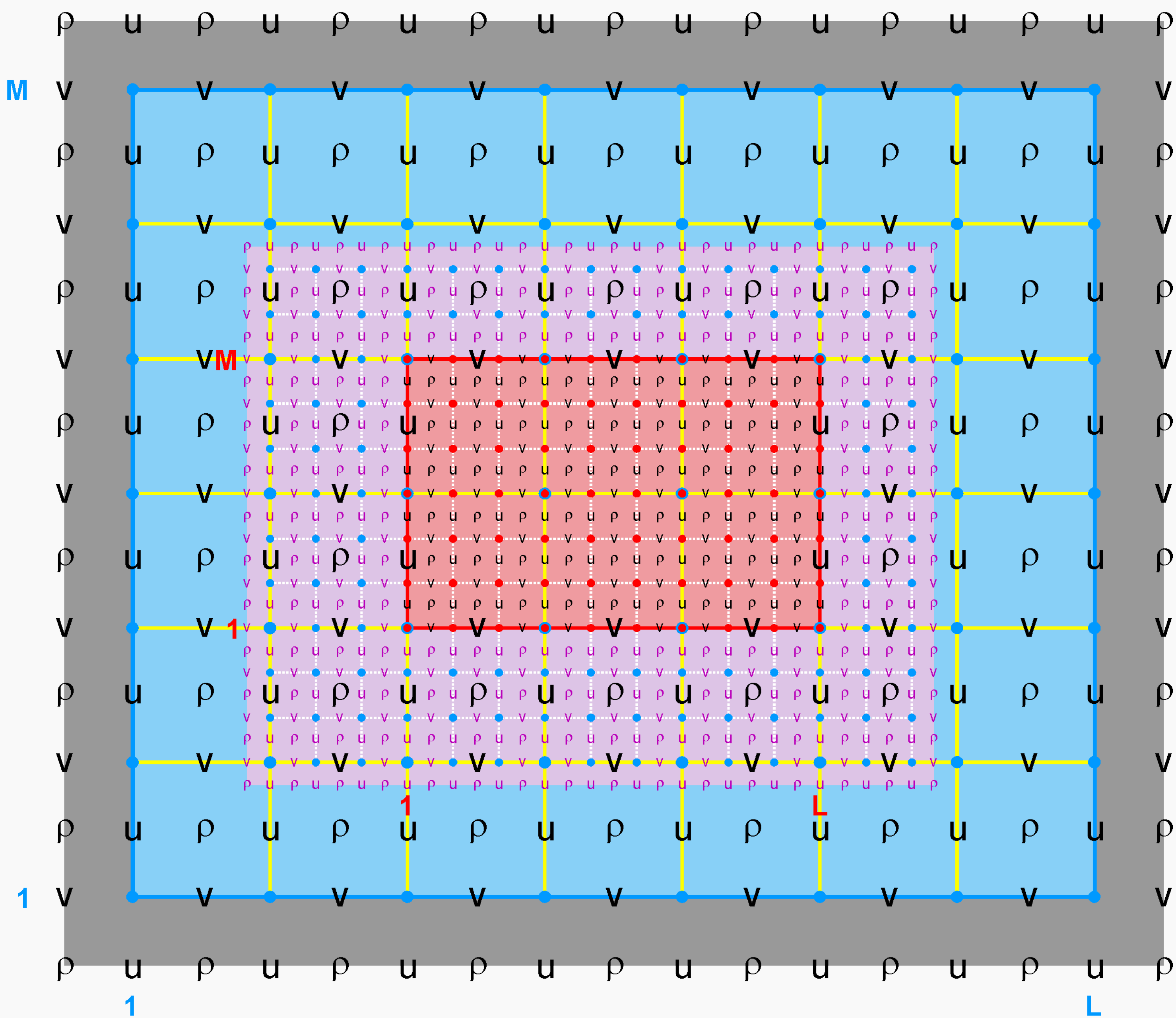
Double Periodic

Nesting Inspiration



Double Periodic to Refinement

Nested Grids: Refinement Class



Ngrids = 2
NestLayers = 2
GridsInLayer = 1 1
Ncontact = 2
Donor Grid = blue
Receiver Grid = red

Ncontact=(Ngrids-1)*2

Nesting Strategy

- The horizontal **i**- and **j**-ranges in the numerical kernel DO-loops are expanded to allow operations on various nested grid classes (refinement, mosaics, and composite) and nesting layers (refinement and composite combinations).

  This facilitates the computation of any horizontal operators (advection, diffusion, gradient, etc.) in the nesting overlap regions and avoids the need for cumbersome lateral boundary conditions on the model variables and their associated flux/gradient values. The advantage of this approach is that it is generic to any discrete horizontal operator. The overlap region is an extended section of the grid that overlays an adjacent grid.

  The strategy is to compute the full horizontal operator at the contact points between nested grids instead of specifying boundary conditions.

Nesting Strategy

- Nowadays, the lateral boundary conditions are set with logical switches (LBC structure) that depend on the nested grid.

  This facilitates, in a generic way, the processing or not of lateral boundary conditions in applications with nested grids. In nesting applications, the values at the lateral boundary points are computed directly in the overlap region by the numerical kernel.

  The logical switches allow different lateral boundary conditions types between active (temperature and salinity) and passive (biology, sediment, inert, etc.) tracers.

  The lateral boundary condition switches for each state variable and boundary edge are now specified in ROMS input script file, ocean.in.

Nesting Strategy

- The nesting calls appear only in the main time-stepping routines, main2d or main3d. The concept of nesting layers is introduced to allow applications with both composite grids and refinement grids. Several routines in module nesting_mod are used to process the information that it is required in the overlap region, what information needs to be exchanged from/to another grid, and when to exchange it.

  In mosaic and composed grids, the information is exchanged between each sub-time step call in main2d or main3d. For example, the data donor grid and the mosaic/composite grids need to sub-time step the 2D momentum equations before any of them start solving and coupling the 3D momentum equations.

  In refinement grids, the information at the contact points is processed at the end of the full time-step layer. The exchange between data donor and refinement grids is two-way.

Tile I- and J-Ranges

get_bounds.F

```
Istr    = BOUNDS(ng) % Istr    (tile)
IstrB   = BOUNDS(ng) % IstrB   (tile)
IstrM   = BOUNDS(ng) % IstrM   (tile)
IstrP   = BOUNDS(ng) % IstrP   (tile)
IstrR   = BOUNDS(ng) % IstrR   (tile)
IstrT   = BOUNDS(ng) % IstrT   (tile)
IstrU   = BOUNDS(ng) % IstrU   (tile)

Iend    = BOUNDS(ng) % Iend    (tile)
IendB   = BOUNDS(ng) % IendB   (tile)
IendP   = BOUNDS(ng) % IendP   (tile)
IendR   = BOUNDS(ng) % IendR   (tile)
IendT   = BOUNDS(ng) % IendT   (tile)

Jstr    = BOUNDS(ng) % Jstr    (tile)
JstrB   = BOUNDS(ng) % JstrB   (tile)
JstrM   = BOUNDS(ng) % JstrM   (tile)
JstrP   = BOUNDS(ng) % JstrP   (tile)
JstrR   = BOUNDS(ng) % JstrR   (tile)
JstrT   = BOUNDS(ng) % JstrT   (tile)
JstrV   = BOUNDS(ng) % JstrV   (tile)

Jend    = BOUNDS(ng) % Jend    (tile)
JendB   = BOUNDS(ng) % JendB   (tile)
JendP   = BOUNDS(ng) % JendP   (tile)
JendR   = BOUNDS(ng) % JendR   (tile)
JendT   = BOUNDS(ng) % JendT   (tile)
```

```
Istrm3  = BOUNDS(ng) % Istrm3  (tile)   Istr-3
Istrm2  = BOUNDS(ng) % Istrm2  (tile)   Istr-2
Istrm1  = BOUNDS(ng) % Istrm1  (tile)   Istr-1
IstrUm2 = BOUNDS(ng) % IstrUm2 (tile)   IstrU-2
IstrUm1 = BOUNDS(ng) % IstrUm1 (tile)   IstrU-1

Iendp1  = BOUNDS(ng) % Iendp1  (tile)   Iend+1
Iendp2  = BOUNDS(ng) % Iendp2  (tile)   Iend+2
Iendp2i = BOUNDS(ng) % Iendp2i (tile)   Iend+2  interior
Iendp3  = BOUNDS(ng) % Iendp3  (tile)   Iend+3

Jstrm3  = BOUNDS(ng) % Jstrm3  (tile)   Jstr-3
Jstrm2  = BOUNDS(ng) % Jstrm2  (tile)   Jstr-2
Jstrm1  = BOUNDS(ng) % Jstrm1  (tile)   Jstr-1
JstrVm2 = BOUNDS(ng) % JstrVm2 (tile)   JstrV-2
JstrVm1 = BOUNDS(ng) % JstrVm1 (tile)   JstrV-1

Jendp1  = BOUNDS(ng) % Jendp1  (tile)   Jend+1
Jendp2  = BOUNDS(ng) % Jendp2  (tile)   Jend+2
Jendp2i = BOUNDS(ng) % Jendp2i (tile)   Jend+2  interior
Jendp3  = BOUNDS(ng) % Jendp3  (tile)   Jend+3
```

Suffix:

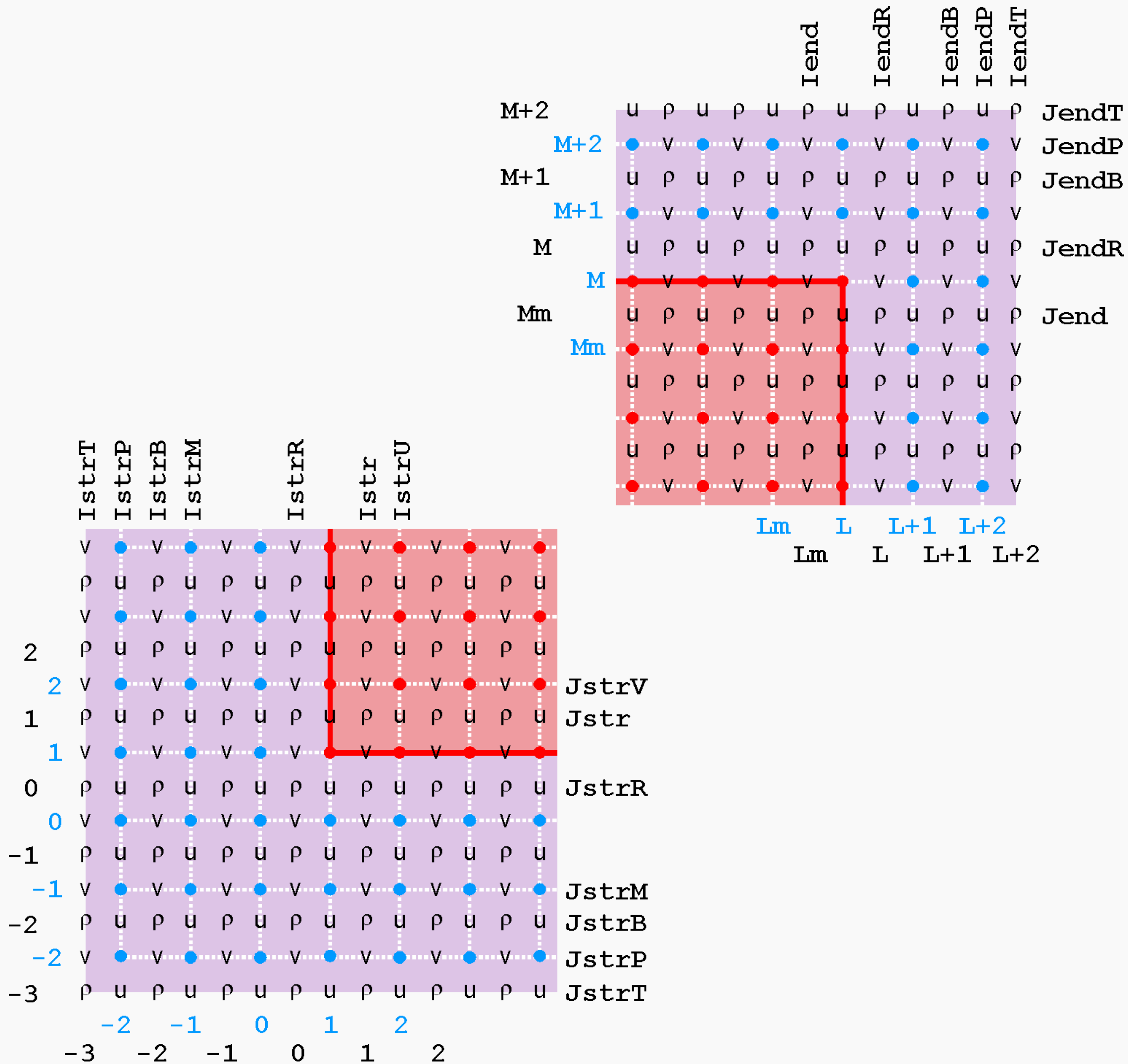
- R : tile RHO-points
- U : tile U-points
- V : tile V-points
- B : Boundary tile RHO- and V-points
- M : Boundary tile PSI- and U-points
- P : Nesting PSI-, U-, and V-points
- T : Nesting RHO-points

Boundary Tile Indices

If not nesting grids, the additional boundary tile indices associated with nesting are set to:

IstrT = IstrR	full range, starting I- direction (RHO-point)
IendT = IendR	full range, ending I- direction (RHO-point)
JstrT = JstrR	full range, starting J-direction (RHO-point)
JendT = JendR	full range, ending J-direction (RHO-point)
IstrP = Istr	full range, starting I- direction (PSI-, U-point)
IendP = Iend	full range, ending I- direction (PSI-point)
JstrP = Jstr	full range, starting J-direction (PSI-, V-point)
JendP = Jend	full range, ending J-direction (PSI-point)
IstrB = Istr	interior range, starting I- direction (RHO-, V-point)
IendB = Iend	interior range, ending I- direction (RHO-, V-point)
JstrB = Jstr	interior range, starting J-direction (RHO-, U-point)
JendB = Jend	interior range, ending J-direction (RHO-, U-point)
IstrM = IstrU	interior range, starting I- direction (PSI-, U-point)
JstrM = JstrV	interior range, starting J-direction (PSI-, V-point)

Boundary Tile Indices Locations



Lateral Boundary Conditions Structure

```
TYPE T_LBC
     logical :: acquire                  process lateral boundary data

     logical :: Chapman_explicit
     logical :: Chapman_implicit
     logical :: clamped
     logical :: closed
     logical :: Flather
     logical :: gradient
     logical :: nested
     logical :: nudging
     logical :: periodic
     logical :: radiation
     logical :: reduced
     logical :: Shchepetkin
END TYPE T_LBC

TYPE (T_LBC), allocatable :: LBC(:,:,:)
```

For example, for free-surface gradient boundary conditions we have:

```
     LBC(iwest,  isFsur, ng) % gradient
     LBC(ieast,  isFsur, ng) % gradient
     LBC(isouth, isFsur, ng) % gradient
     LBC(inorth, isFsur, ng) % gradient
```

Lateral Boundary Conditions Code

For Example, in zetabc.F the western boundary conditions are:

```
IF ( DOMAIN (ng) % Western_Edge(tile) ) THEN

  IF ( LBC (iwest, isFsur, ng) % radiation ) THEN
     …
  ELSE  IF ( LBC (iwest, isFsur, ng) % Chapman_explicit ) THEN
     …
  ELSE  IF ( LBC (iwest, isFsur, ng) % Chapman_implicit ) THEN
     …
  ELSE  IF ( LBC (iwest, isFsur, ng) % clamped ) THEN
     …
  ELSE  IF ( LBC (iwest, isFsur, ng) % gradient ) THEN
     …
  ELSE  IF ( LBC (iwest, isFsur, ng) % closed ) THEN

     DO  j = Jstr, Jend
        IF ( LBC_apply (ng) % west ( j ) ) THEN                    !  Allows both specified and
           zeta ( Istr-1, j, kout )  = zeta ( Istr, j, kout )      !  nested conditions
        END  IF
     END  DO

  END  IF

END  IF
```

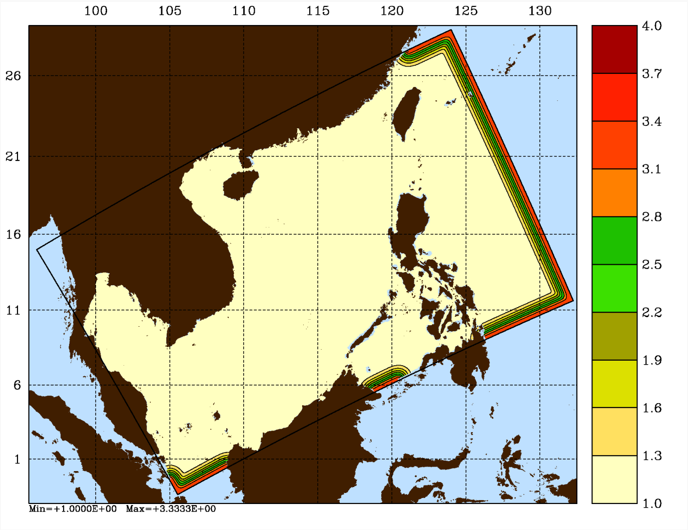
Viscosity and Diffusion Sponges

The horizontal viscosity is now computed as:

```
visc2_r(i,j) = visc_factor(i,j) * visc2_r(i,j)
visc4_r(i,j) = visc_factor(i,j) * visc4_r(i,j)
```

And the horizontal diffusion is now computed as:

```
diff2(i,j,itrc) = diff_factor(i,j) * diff2(i,j,itrc)
diff4(i,j,itrc) = diff_factor(i,j) * diff4(i,j,itrc)
```



The horizontal mixing coefficients (visc_factor and diff_factor) can be set with analytical functions using ANA_SPONGE or can be read from input GRID NetCDF file variables:

```
double visc_factor (eta_rho, xi_rho) ;
        visc_factor:long_name = "horizontal viscosity sponge factor" ;
        visc_factor:valid_min = 0. ;
        visc_factor:coordinates = "lon_rho lat_rho" ;

double diff_factor (eta_rho, xi_rho) ;
        diff_factor:long_name = "horizontal diffusivity sponge factor" ;
        diff_factor:valid_min = 0. ;
        diff_factor:coordinates = "lon_rho lat_rho" ;
```

The Matlab script add_sponge.m can be used to append sponge variables to the application GRID NetCDF file

Standard Input File: Rivers and Sponges

Logical switches (TRUE/FALSE) to activate horizontal momentum transport point Sources/Sinks (like river runoff transport) and mass point Sources/Sinks (like volume vertical influx), [1:Ngrids].

```
LuvSrc == 3*F          ! horizontal momentum transport
 LwSrc == 3*F          ! volume vertical influx
```

Logical switches (TRUE/FALSE) to activate tracers point Sources/Sinks (like river runoff) and to specify which tracer variables to consider: [1:NAT+NPT,Ngrids]. See glossary below for details.

```
LtracerSrc == 2*F 2*F 2*F          ! temperature, salinity, inert
```

Logical switches (TRUE/FALSE) to increase/decrease horizontal viscosity and/or diffusivity in specific areas of the application domain (like sponge areas) for the desired application grid.

```
   LuvSponge == 3*F                ! horizontal momentum
LtracerSponge == 2*F  2*F  2*F     ! temperature, salinity, inert
```

Standard Input File: Climatology and Nudging

Logical switches (TRUE/FALSE) to read and process climatology fields. See glossary below for details.

```
    LsshCLM == 3*F                 ! sea-surface height
     Lm2CLM == 3*F                 ! 2D momentum
     Lm3CLM == 3*F                 ! 3D momentum

 LtracerCLM == 2*F 2*F 2*F         ! temperature, salinity, inert
```

Logical switches (TRUE/FALSE) to nudge the desired climatology field(s). If not analytical climatology fields, users need to turn ON the logical switches above to process the fields from the climatology NetCDF file that are needed for nudging. See glossary below for details.

```
LnudgeM2CLM == 3*F                 ! 2D momentum
LnudgeM3CLM == 3*F                 ! 3D momentum

 LnudgeTCLM == 2*F 2*F 2*F         ! temperature, salinity, inert
```

Nudging Coefficients Metadata

The inverse (1/time) nudging coefficients can be set with analytical functions using ANA_NUDGCOEF or can be read from new input NUDNAME NetCDF file variables:

```
double M2_NudgeCoef (eta_rho, xi_rho) ;
        M2_NudgeCoef:long_name = "2D momentum inverse nudging coefficients" ;
        M2_NudgeCoef:units = "day-1" ;
        M2_NudgeCoef:coordinates = "xi_rho eta_rho " ;

double M3_NudgeCoef (s_rho, eta_rho, xi_rho) ;
        M3_NudgeCoef:long_name = "3D momentum inverse nudging coefficients" ;
        M3_NudgeCoef:units = "day-1" ;
        M3_NudgeCoef:coordinates = "xi_rho eta_rho s_rho " ;

double tracer_NudgeCoef (s_rho, eta_rho, xi_rho) ;
        tracer_NudgeCoef:long_name = "generic tracer inverse nudging coefficients" ;
        tracer_NudgeCoef:units = "day-1" ;
        tracer_NudgeCoef:coordinates = "xi_rho eta_rho s_rho " ;

double temp_NudgeCoef (s_rho, eta_rho, xi_rho) ;
        temp_NudgeCoef:long_name = "temp inverse nudging coefficients" ;
        temp_NudgeCoef:units = "day-1" ;
        temp_NudgeCoef:coordinates = "xi_rho eta_rho s_rho " ;

double salt_NudgeCoef (s_rho, eta_rho, xi_rho) ;
        salt_NudgeCoef:long_name = "salt inverse nudging coefficients" ;
        salt_NudgeCoef:units = "day-1" ;
        salt_NudgeCoef:coordinates = "xi_rho eta_rho s_rho " ;
```

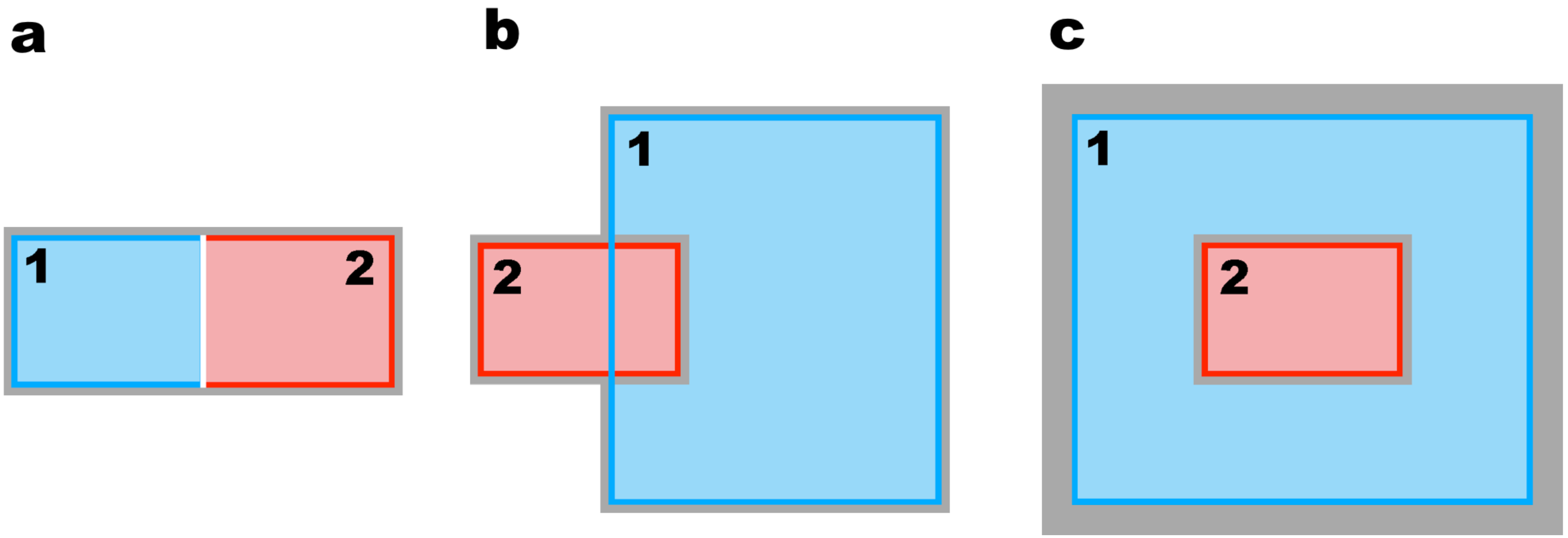
Nesting Configuration Types

- Composite Grids Super-Class:
    1. Mosaic Grids Sub-Class
    2. Composite Overlap Grids Sub-Class
    3. Complex Estuary Composite Grids Sub-Class
    4. Partial Boundary Composite Grids Sub-Class

- Refinement Grids Super-Class:
    1. Single Refinement Sub-Class
    2. Multiple Refinement Sub-Class

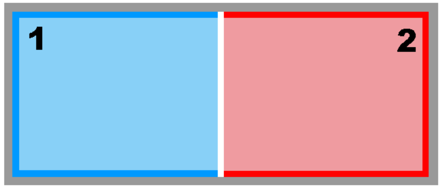
- Composite and Refinement Combination Super-Class:
    1. Refinement and Partial Boundary Composite Sub-Class
    2. Complex Estuary Refinement-Composite Sub-Class

Nesting Classes

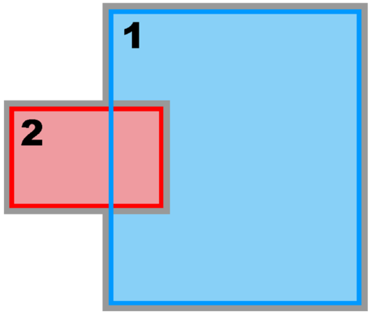


Mosaic Composite Refinement

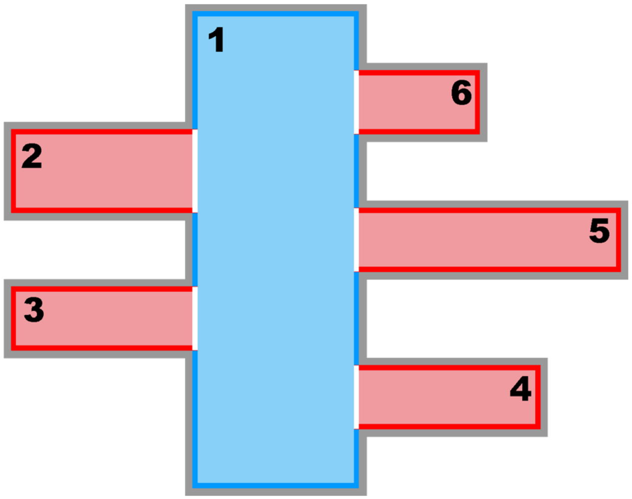
Composite Grid Sub-Classes



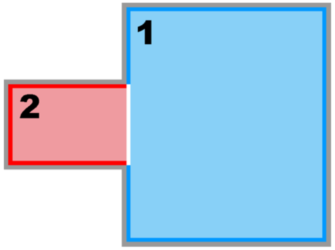
Mosaic Sub-Class:
Ngrids = 2
NestedLayers = 1
GridsInLayer = 2
Ncontact = 2



Composite Overlap Sub-Class:
Ngrids = 2
NestedLayers = 1
GridsInLayer = 2
Ncontact = 2

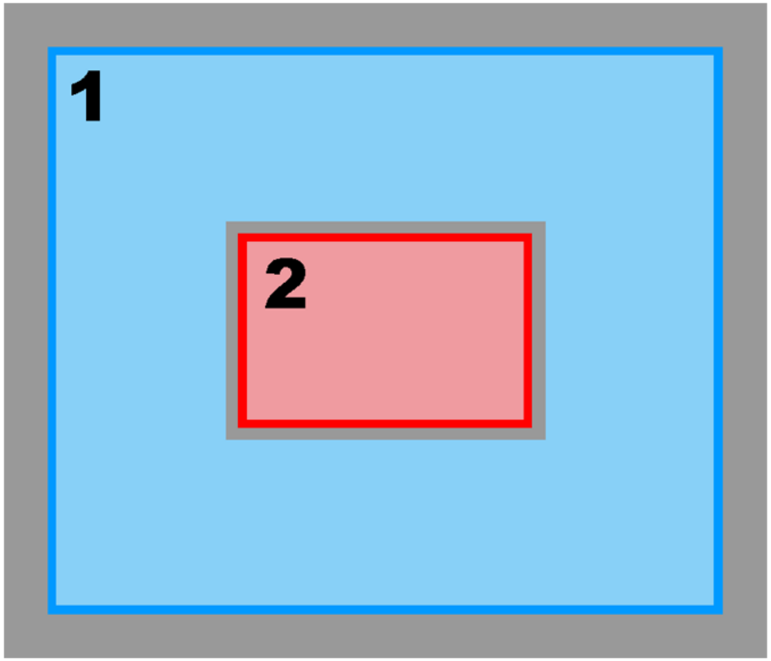


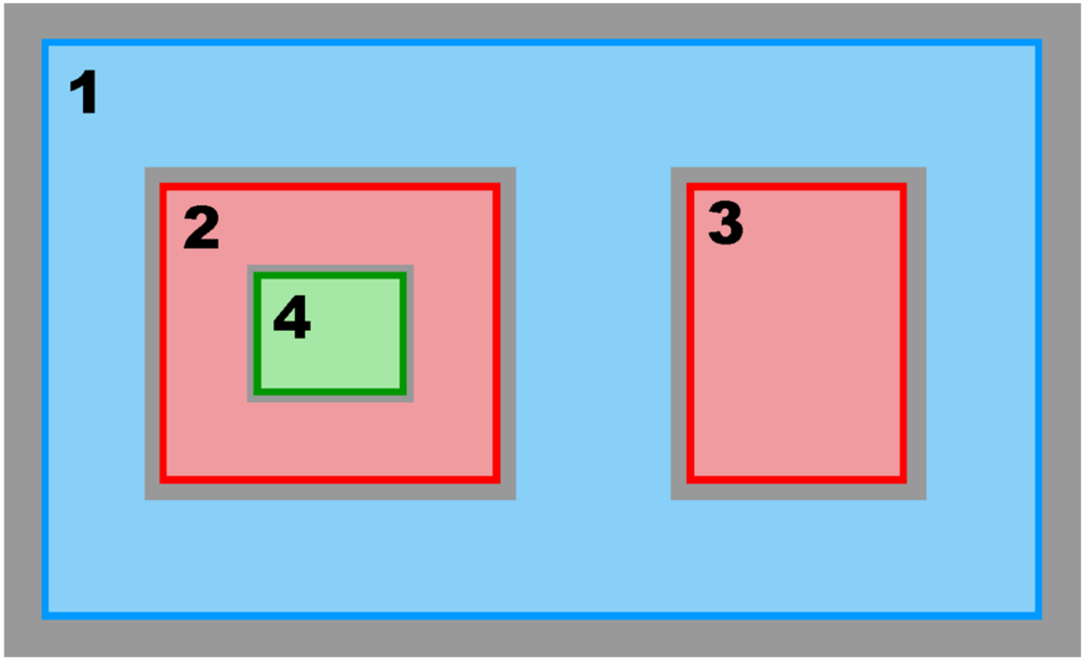
Complex Estuary Composite Sub-Class:
Ngrids = 6
NestedLayers = 1
GridsInLayer = 6
Ncontact = 10



Partial Boundary Composite Sub-Class:
Ngrids = 2
NestedLayers = 1
GridsInLayer = 2
Ncontact = 2

Refinement Grid Sub-Classes





Single Refinement Sub-Class:

Ngrids = 2

NestedLayers = 2

GridsInLayer = 1 1

Ncontact = 2

Multiple Refinement Sub-Class:

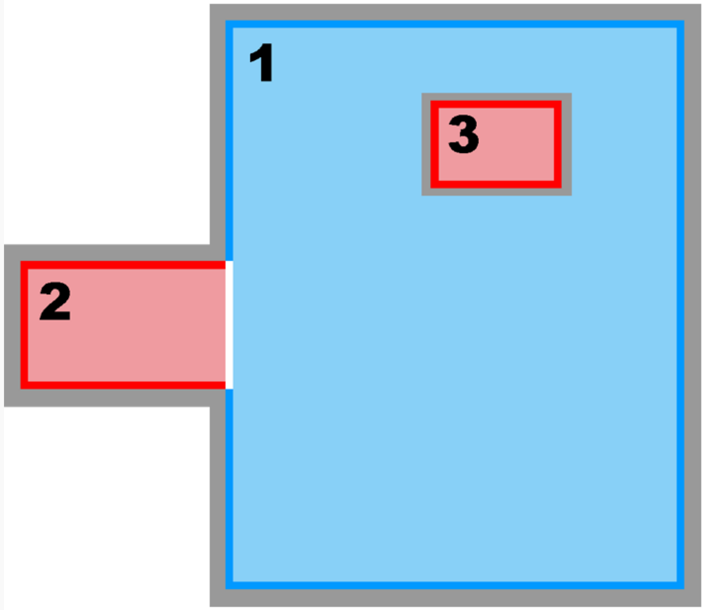
Ngrids = 4

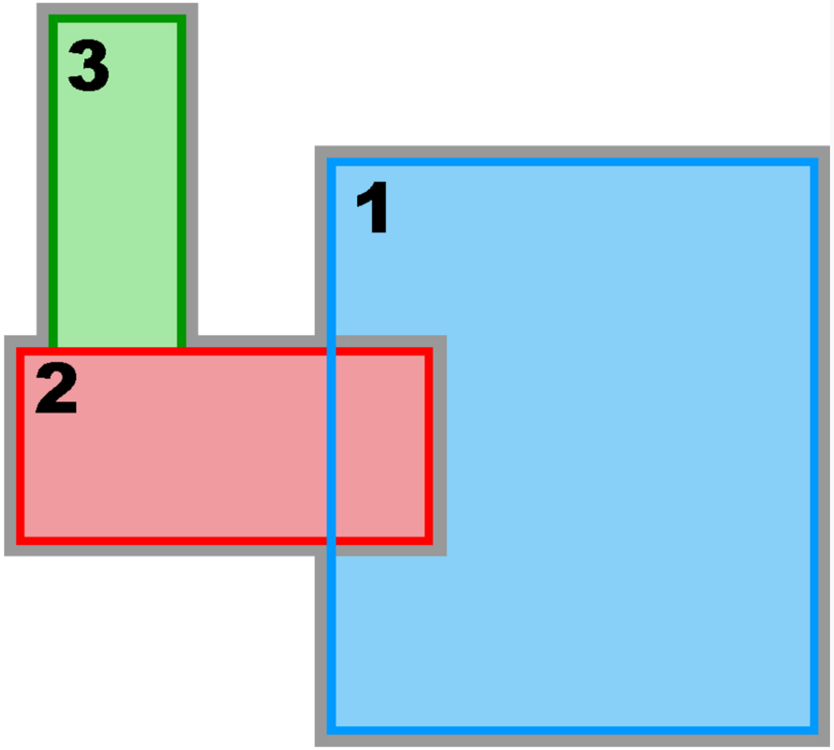
NestedLayers = 3

GridsInLayer = 1 2 1

Ncontact = 6

Composite-Refinement Grid Sub-Classes

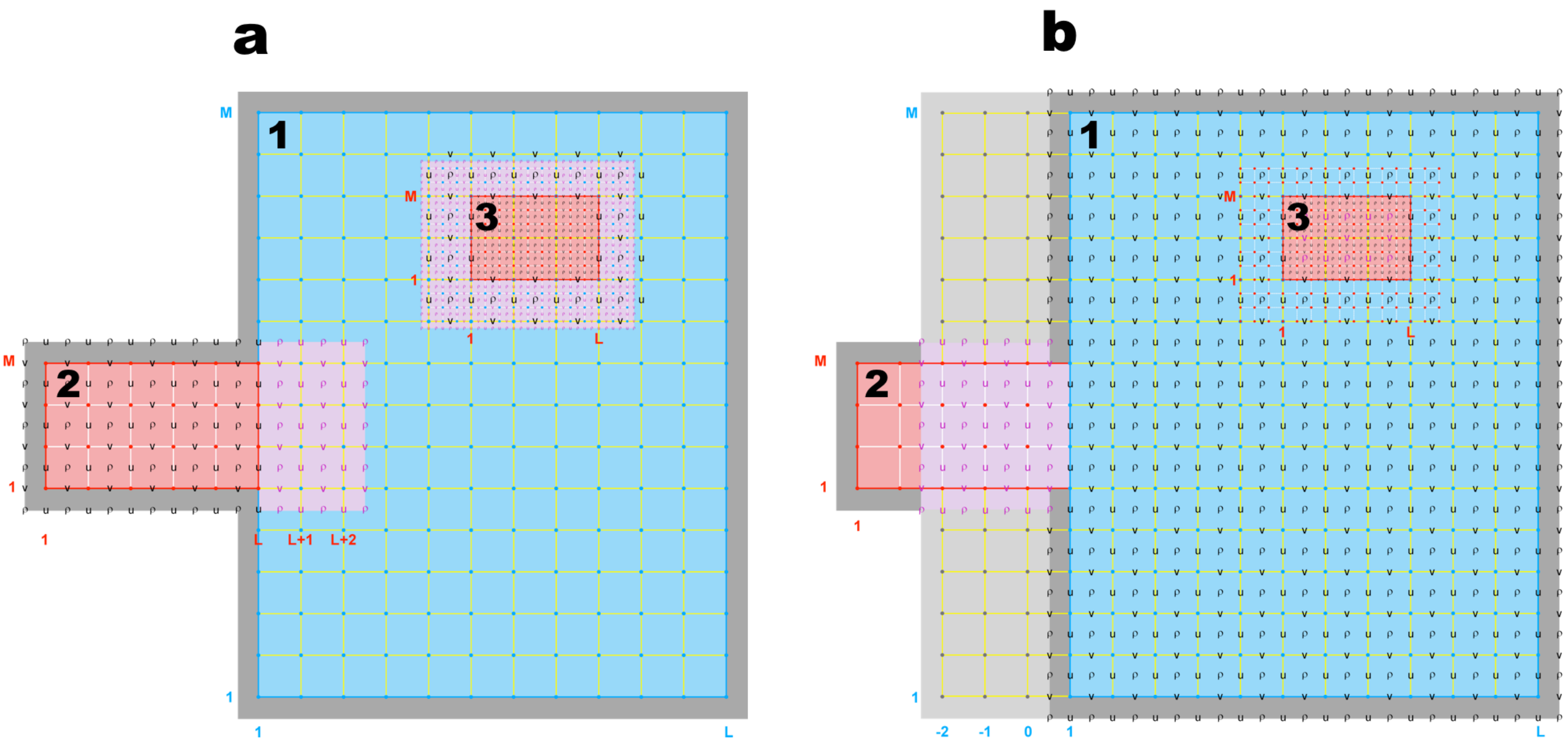




Refinement and Partial Boundary Composite Sub-Class:
Ngrids = 3
NestedLayers = 2
GridsInLayer = 2 1
Ncontact = 4

Complex Estuary Refinement-Composite Sub-Class:
Ngrids = 3
NestedLayers = 2
GridsInLayer = 1 2
Ncontact = 4

Contact Areas and Points



Refinement-Composite Sub-Class

Contact Areas and Points: Definitions

Contact Region (cr): Extended section of the nested grid that overlays an adjacent nested grid. It is the region where the exchange of data between nested grids takes place. Since ROMS nesting is two-way by default, there are Ncontact=(Ngrids-1)*2 contact regions, where Ngrids is the number of nested grids and Ncontact is the number of contact regions in a nested application. There is a duality in ROMS grid nesting: data donor in one contact region and data receiver in its conjugate contact region. Each contact region has a donor and a receiver grid.

Contact Points: Grid cells inside a contact region. Since ROMS governing equations are solved in an Arakawa C-grid, there are contact points at ρ-, Ψ-, u-, and v-points. However, the Ψ-points are only used to define the physical grid perimeters within a contact region. Since the C-grid stencil indices in ROMS are left-bottom ordered, there are always 4 ρ contact points at the left and bottom side of the contact region. On the other hand, there are 3 ρ contact points on the right and top side of the contact region.

Donor Grid (dg): Data source grid in a nesting contact region. In refinement, the donor grid is used either to interpolate data from coarse to fine grid or to average data from fine to coarse grid (two-way feedback).

Receiver Grid (rg): Data recipient grid in a nesting contact region. In refinement, the contact points of the finer receiver grid are interpolated using the coarser donor data from the grid cell containing the contact point. The interpolation can be linear or quadratic. In two-way nesting, when the coarse grid is the receiver grid the finer grid solution is averaged within the coarse cell. The coarse grid cell value is replaced with the finer grid averaged solution. This takes place in routine fine2coarse.

Nesting Layer: Nested grids time-step arrangement and order for the ROMS numerical kernel. It is directly related to the time-step size (dt) for each nested grid. The number of nested layers, NestLayers, is specified in standard input script (ocean.in) and should be equal to the different number of time-step size (dt).

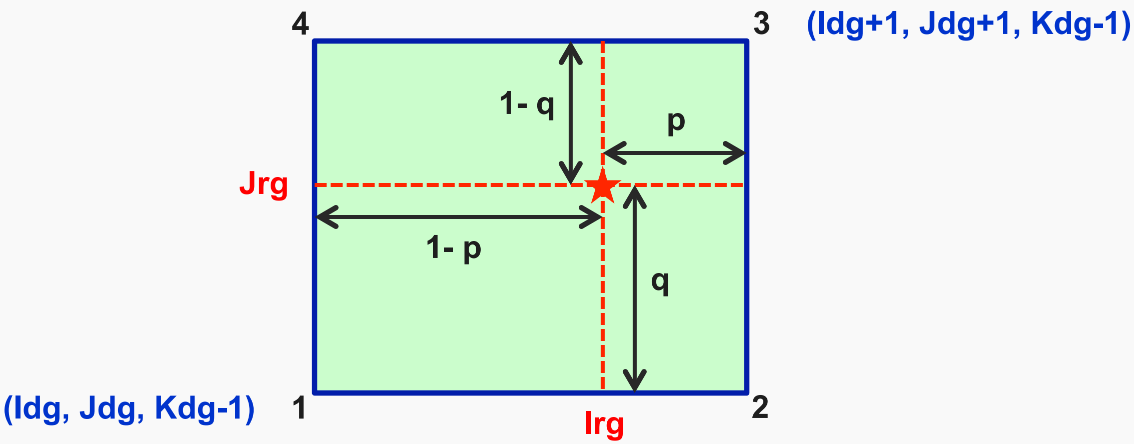
Contact Regions and Contact Points

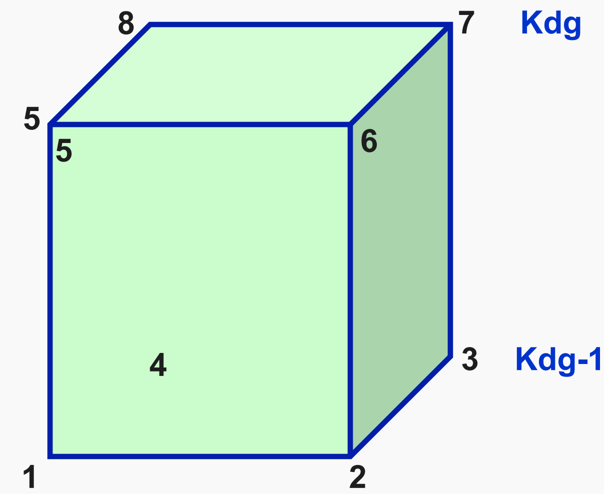
Contact Points Structure

integer :: Ncontact	total number of contact regions
TYPE T_NGC	
logical :: coincident	coincident donor and receiver points, p=q=0
logical :: interpolate	perform vertical interpolation
integer :: donor_grid	data donor grid number
integer :: receiver_grid	data receiver grid number
integer :: Npoints	
integer, pointer :: Idg (:)	donor grid, cell I-left index
integer, pointer :: Jdg (:)	donor grid, cell J-bottom index
integer, pointer :: Kdg (: , :)	donor grid, cell K-index
integer, pointer :: Irg (:)	receiver grid, I-contact point
integer, pointer :: Jrg (:)	receiver grid, J-contact point
real(r8), pointer :: Lweight (: , :)	linear horizontal weights
real(r8), pointer :: LweightUmasked (: , :)	linear horizontal unmasked weights (WET_DRY)
real(r8), pointer :: Qweight (: , :)	quadratic horizontal weights
real(r8), pointer :: QweightUmasked (: , :)	quadratic horizontal unmasked weights (WET_DRY)
real(r8), pointer :: Vweight(: , : , :)	vertical weights
END TYPE T_NGC	
TYPE (T_NGC), allocatable :: Rcontact (:)	RHO-points
TYPE (T_NGC), allocatable :: Ucontact (:)	U-points
TYPE (T_NGC), allocatable :: Vcontact (:)	V-points

Contact Points Interpolation

Suffix: dg = donor grid
rg = receiver grid





Lweight (1, :) = (1 - p) * (1 - q)
Lweight (2, :) = p * (1 - q)
Lweight (3, :) = p * q
Lweight (4, :) = (1 - p) * q

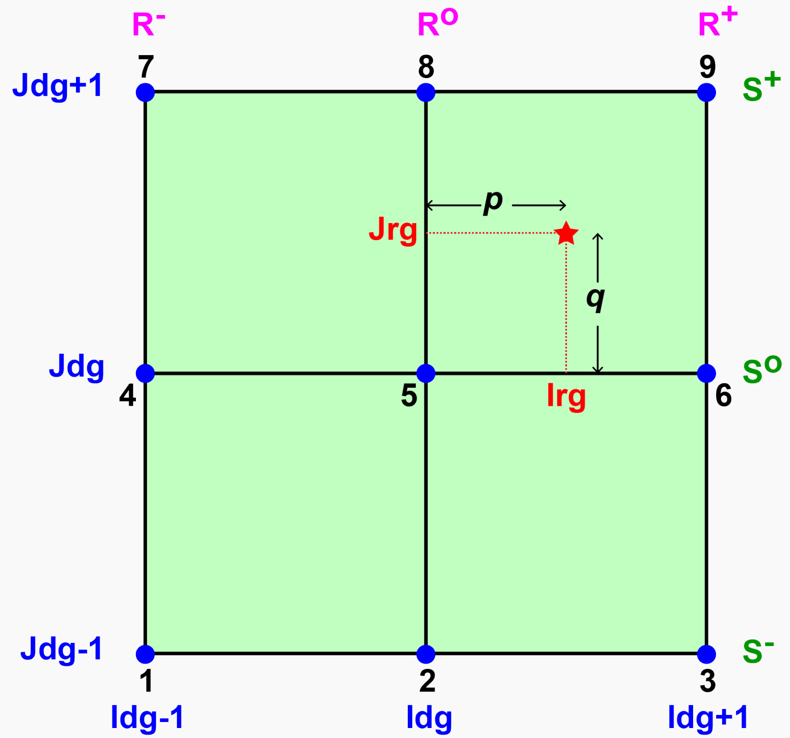
Value (Irg, Jrg) = Lweight (1,:) * F2d(Idg ,Jdg)+
Lweight (2,:) * F2d(Idg+1,Jdg)+
Lweight (3,:) * F2d(Idg+1,Jdg+1)+
Lweight (4,:) * F2d(Idg ,Jdg+1)

If coincident contact points between data donor and data receiver grids, p = q = 0.0,

Lweight (1, :) = 1.0
Lweight (3, :) = 0.0
Lweight (4, :) = 0.0
Lweight (5, :) = 0.0

Value (Irg, Jrg) = Lweight (1, :) * F2d(Idg, Jdg)

Contact Points: Quadratic Interpolation



$R^- = 0.5 * p * (p-1) + \alpha$

$R^\circ = (1-p^2) - 2\alpha$

$R^+ = 0.5 * p * (p+1) + \alpha$

$S^- = 0.5 * q * (q-1) + \alpha$

$S^\circ = (1-q^2) - 2\alpha$

$S^+ = 0.5 * q * (q+1) + \alpha$

The finer grid variable, F, is interpolated from the coarser grid variable, C, as:

```
F(Irg,Jrg) = S- * [R- * C(Idg-1,Jdg-1) + Ro * C(Idg,Jdg-1) + R+ * C(Idg+1,Jdg-1)]+
             So * [R- * C(Idg-1,Jdg  ) + Ro * C(Idg,Jdg  ) + R+ * C(Idg+1,Jdg  )]+
             S+ * [R- * C(Idg-1,Jdg+1) + Ro * C(Idg,Jdg+1) + R+ * C(Idg+1,Jdg+1)]
```

Contact Points: Quadratic Interpolation

For conservation and reversibility, we need:

$$\alpha = \frac{\left(\frac{1}{r}\right)^2 - 1}{24}$$

where r is the grid refinement factor. The reversibility condition is:

$$C(i,j) = \frac{1}{r^2}\sum_{1}^{r^2} F(:,:)$$

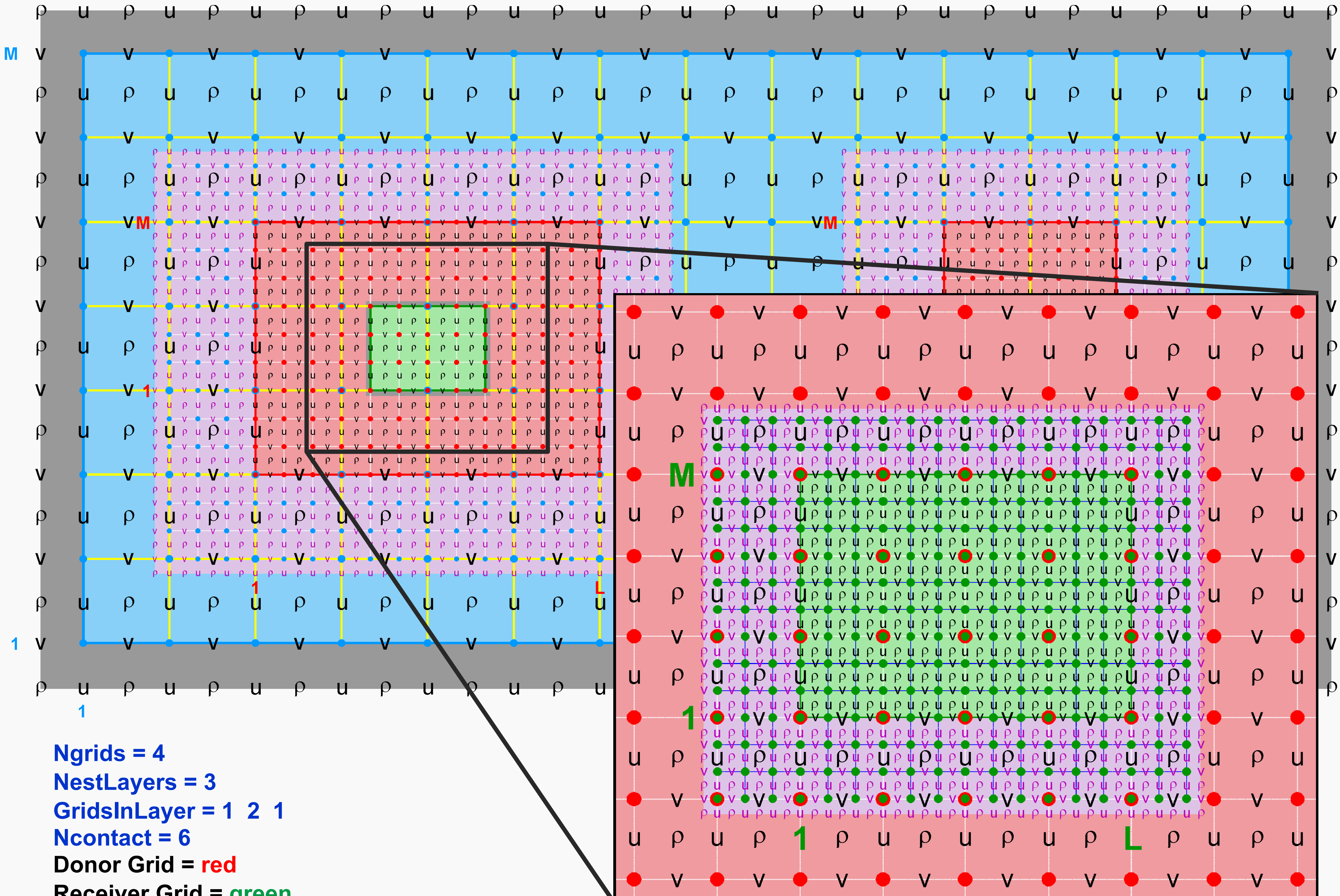
Then, the quadratic interpolation weights are:

```
Qweight (1, :) = R⁻ * S⁻
Qweight (2, :) = R° * S⁻
Qweight (3, :) = R⁺ * S⁻
Qweight (4, :) = R⁻ * S°
Qweight (5, :) = R° * S°
Qweight (6, :) = R⁺ * S°
Qweight (7, :) = R⁻ * S⁺
Qweight (8, :) = R° * S⁺
Qweight (9, :) = R⁺ * S⁺
```

Σ Qweight(1:9,:) = 1

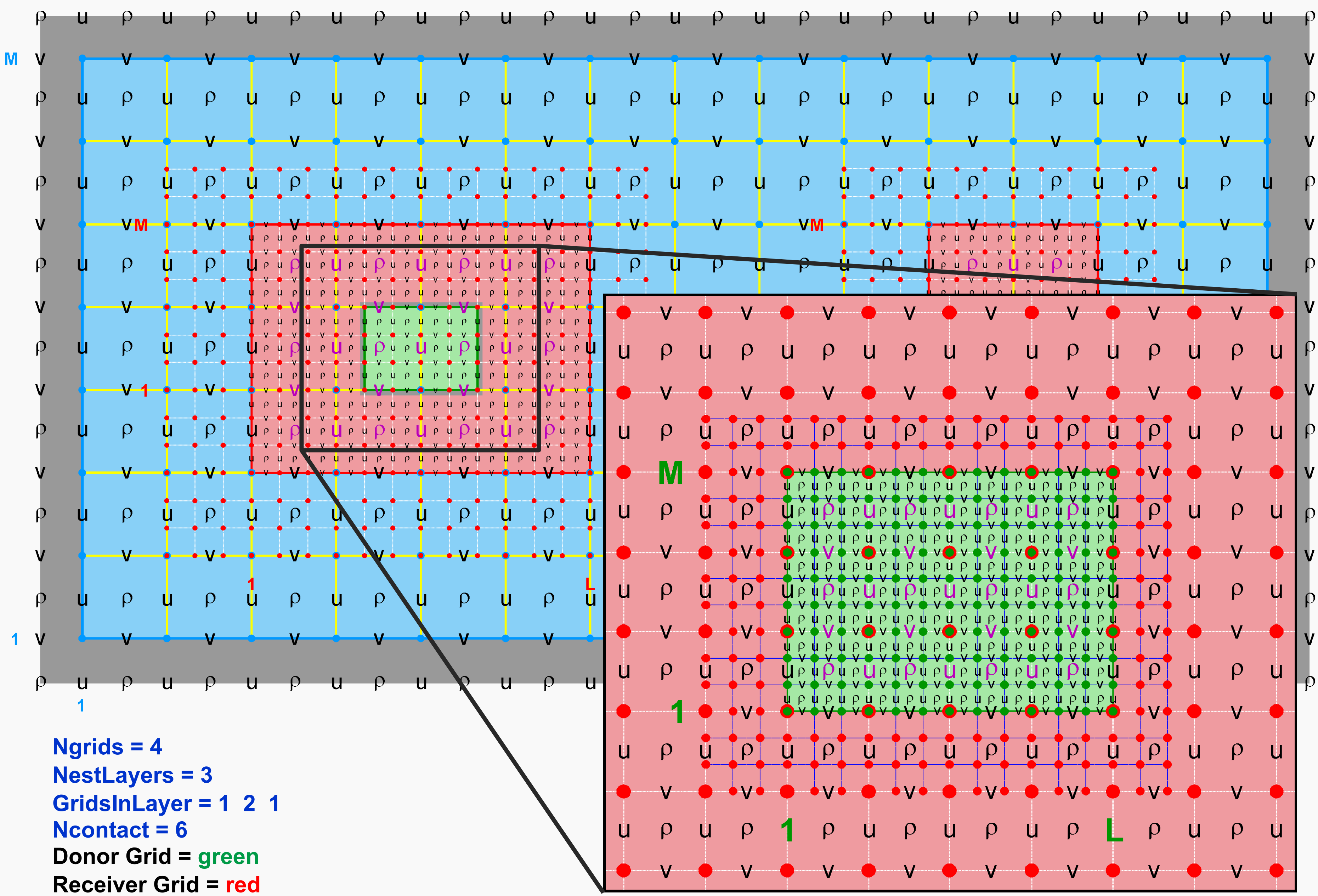
Clark and Farley, 1984

Nested Grids: Multi-Refinement Class



Ngrids = 4
NestLayers = 3
GridsInLayer = 1 2 1
Ncontact = 6
Donor Grid = red
Receiver Grid = green

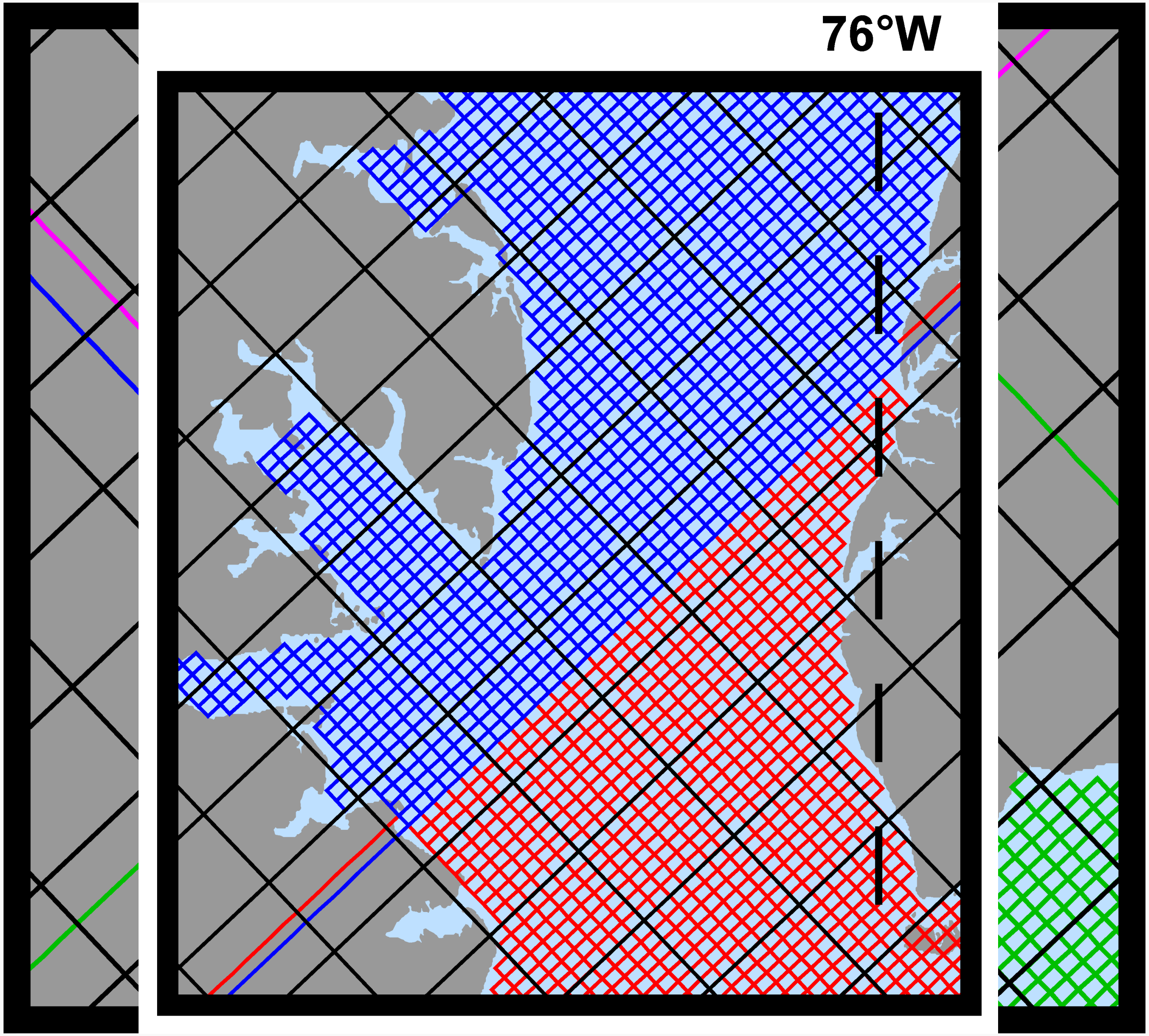
Nested Grids: Multi-Refinement Class



Ngrids = 4
NestLayers = 3
GridsInLayer = 1 2 1
Ncontact = 6
Donor Grid = green
Receiver Grid = red

Realistic Nesting Configuration: US East Coast

(Complex Estuary Refinement-Composite Sub-Class)

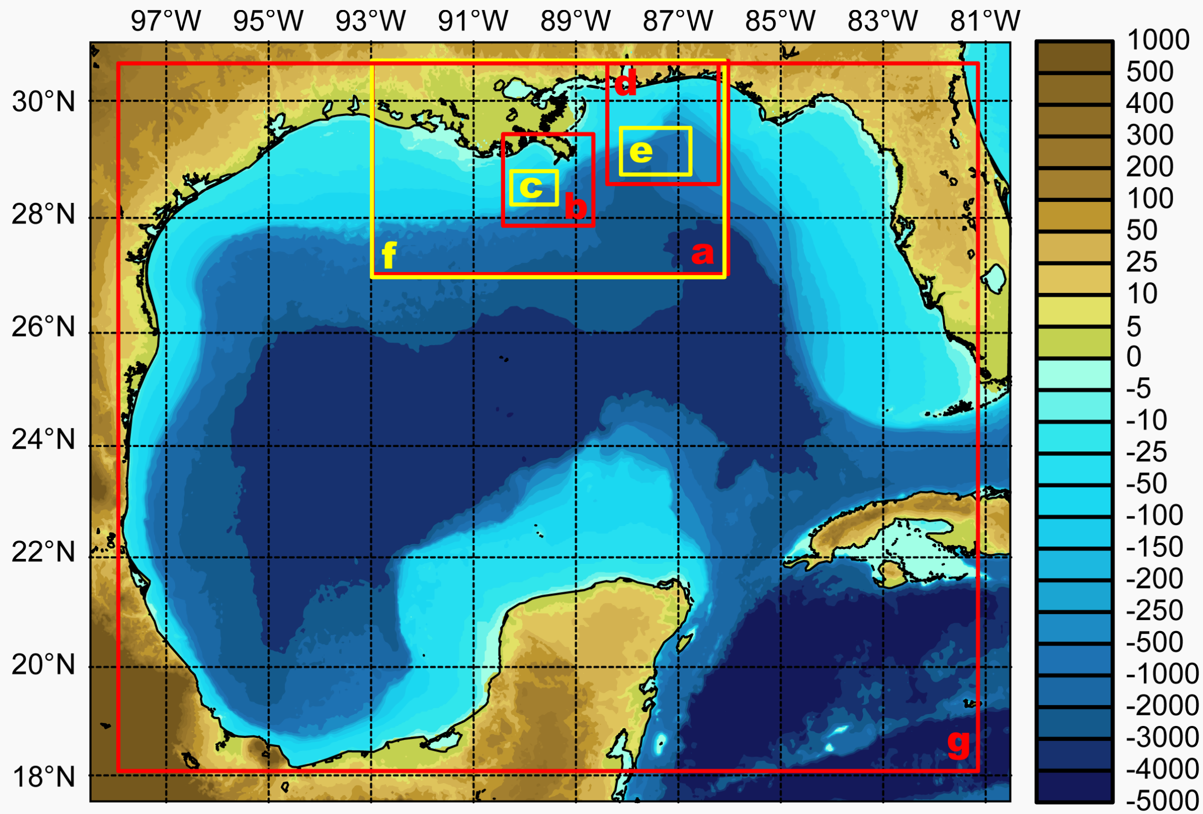


Realistic Nesting Configuration: US East Coast
(Complex Estuary Refinement-Composite Sub-Class)

The coarser grid, ESPRESSO (130 x 82), has an average resolution of dx=7.5km, dy=5.8km. The nested grids (ρ-points mesh) are color coded for convenience to show the strategy used to better resolve the Delaware and Chesapeake Estuary Systems. The red and green are refinement grids whereas blue and magenta are composite grids. The refinement ratio is 1:7. An intermediate 1:7 refinement grid is created using Matlab script coarse2fine.m that included both the Delaware and Chesapeake Estuary Systems. Then, the Matlab script grid_extract.m is used to extract the Delaware Bay refinement grid (58 x 142) and Delaware River composite grid (42 x 55). Similarly, grid_extract.m is used to extract the Chesapeake Bay outer refinement grid (135 x 142) and Chesapeake Bay inner composite grid (233 x 212).

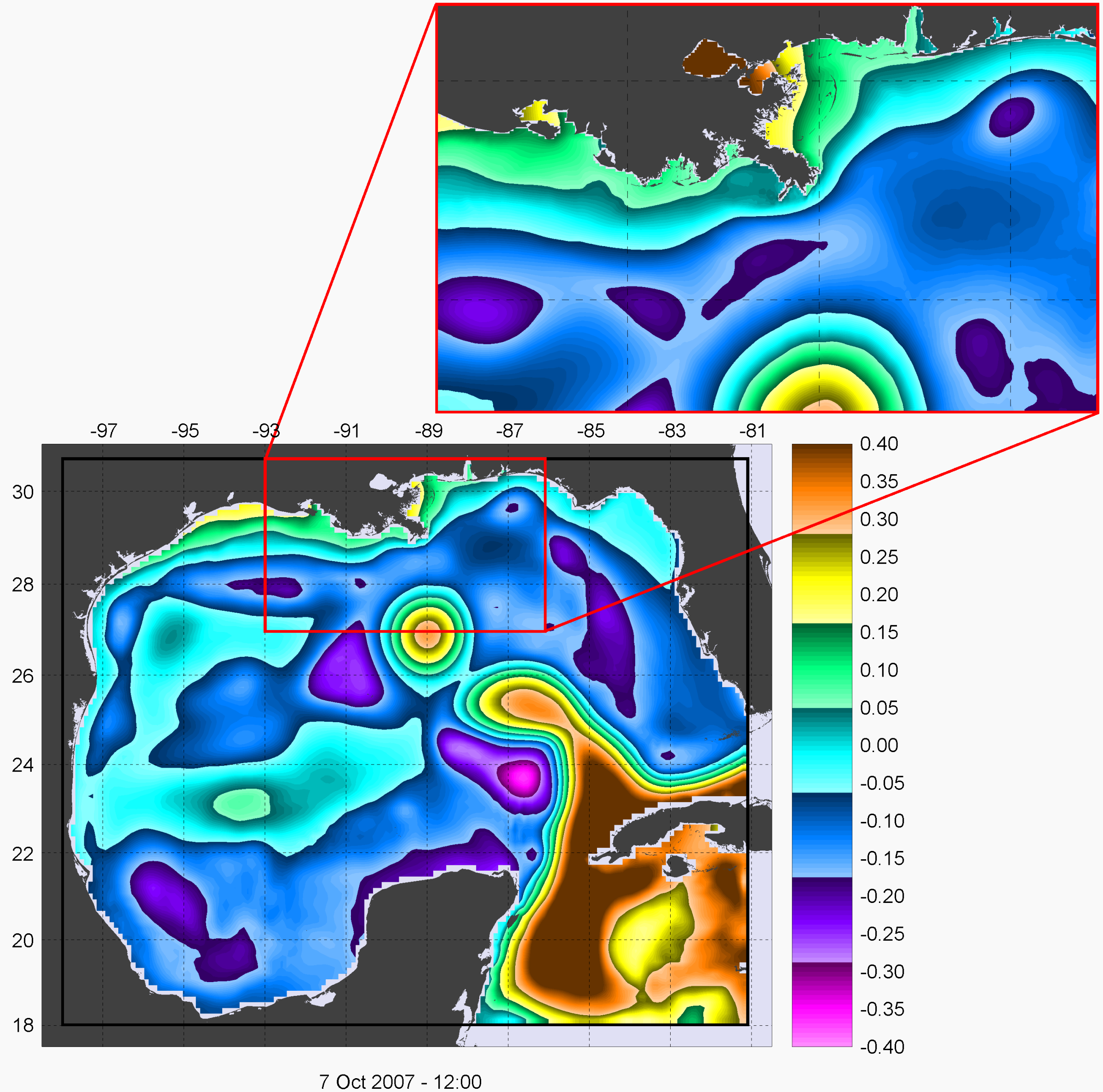
Realistic Nesting Configuration: Gulf of Mexico

(Multiple Refinement Sub-Class)



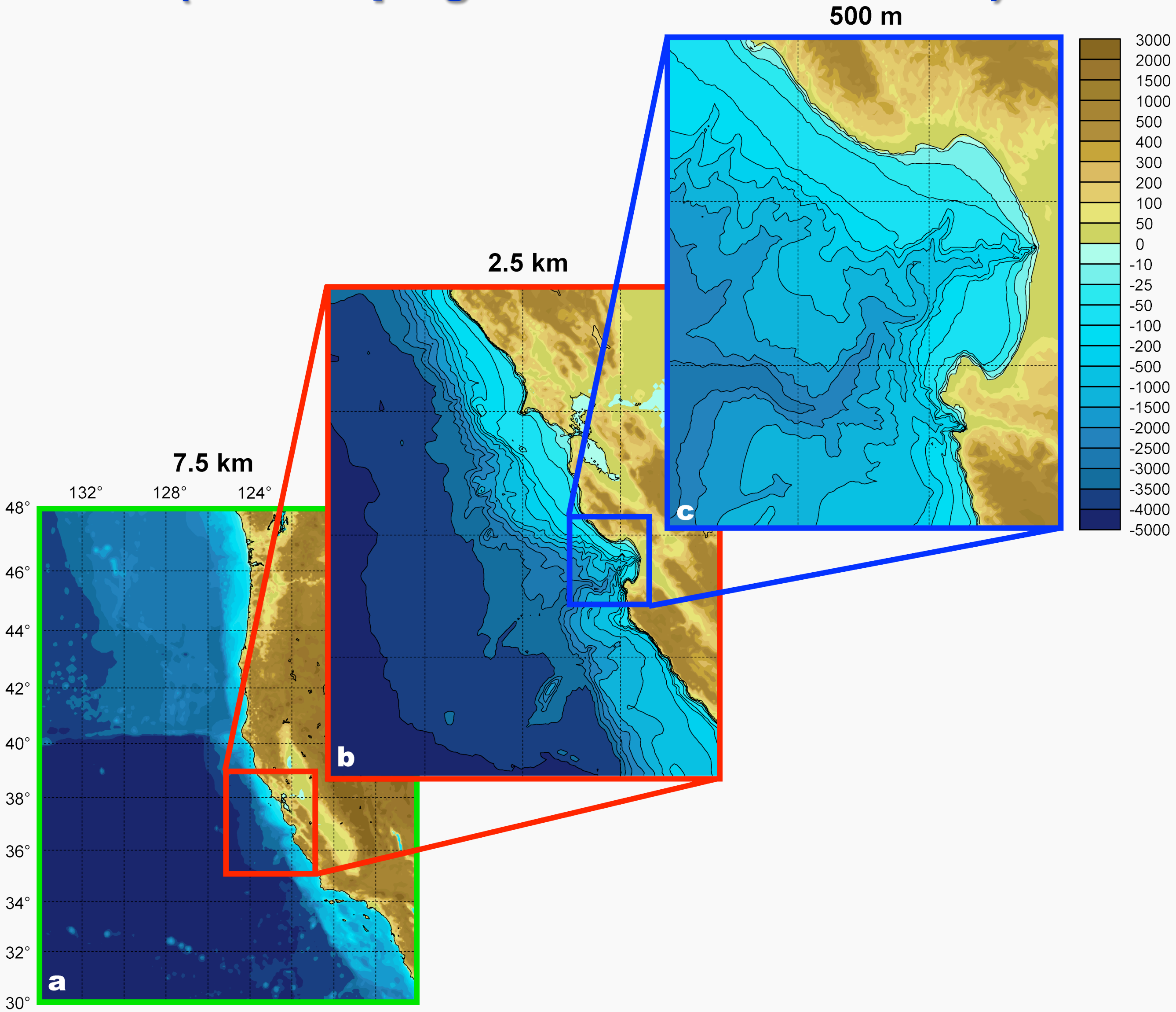
Grid Number Points	Min/Max Δx (m)	Min/Max Δy (m)	Mean Δx (m)	Mean Δy (m)	Refinement Factor
a (256x128)	2595 / 2688	3108 / 3220	2642	3165	-
b (325x275)	525 / 533	630 / 639	530	634	1:5 from **a**
c (825x500)	105 / 107	126 / 127	106	127	1:5 from **b**
d (400x365)	519 / 530	622 / 635	525	628	1:5 from **a**
e (1250x700)	105 / 106	126 / 127	105	126	1:5 from **d**
f (324x150)	2032 / 2106	2696 / 2794	2070	2746	1:5 from **g**
g (158x98)	10160 / 11236	13484 / 14905	10733	14239	-

Realistic Nesting Configuration: Gulf of Mexico
(Multiple Refinement Sub-Class)



Realistic Nesting Configuration: US West Coast

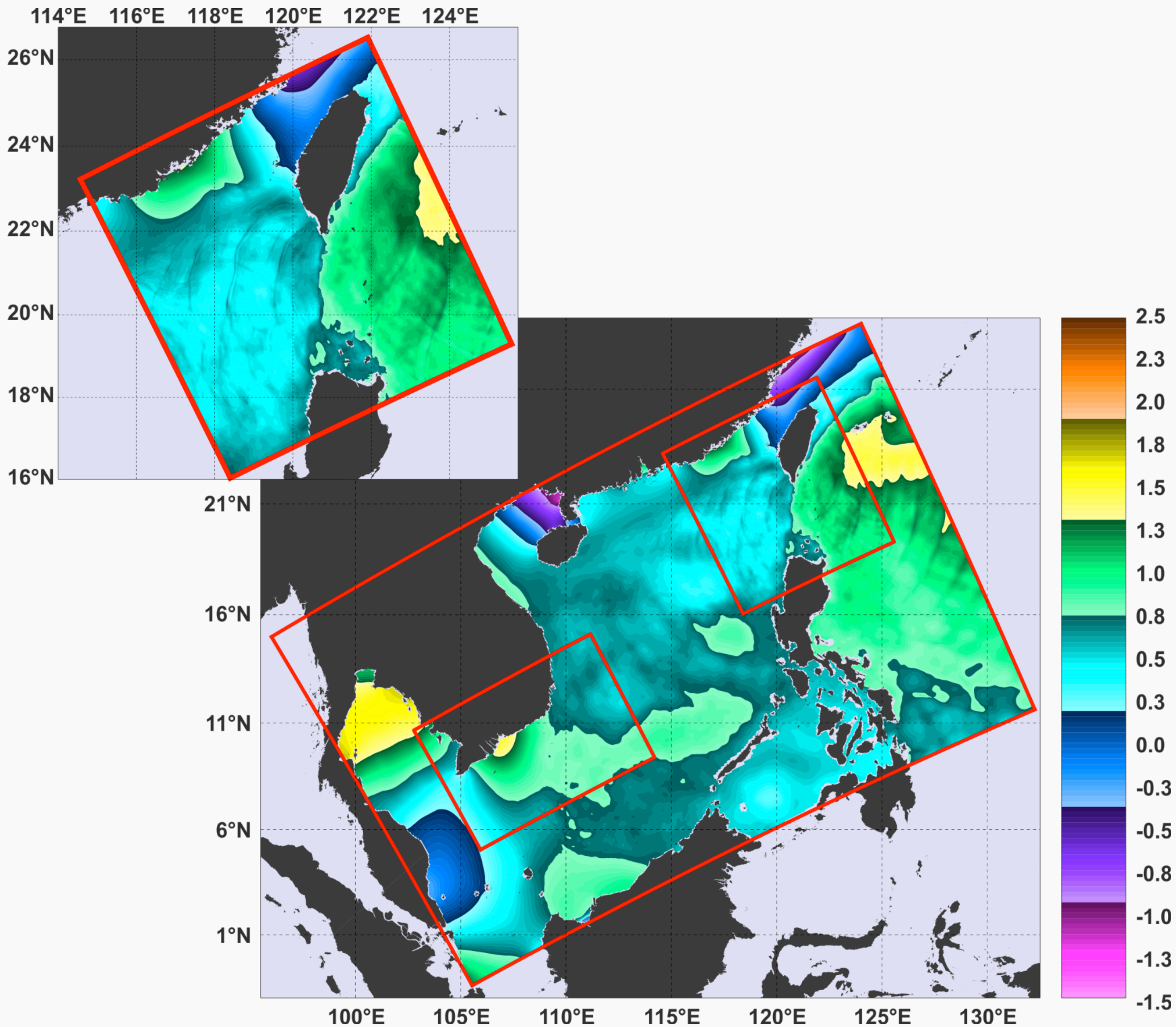
(Telescoping Refinement Sub-Class)



Modeling of Monterey Canyon

Realistic Nesting Configuration: South China Sea

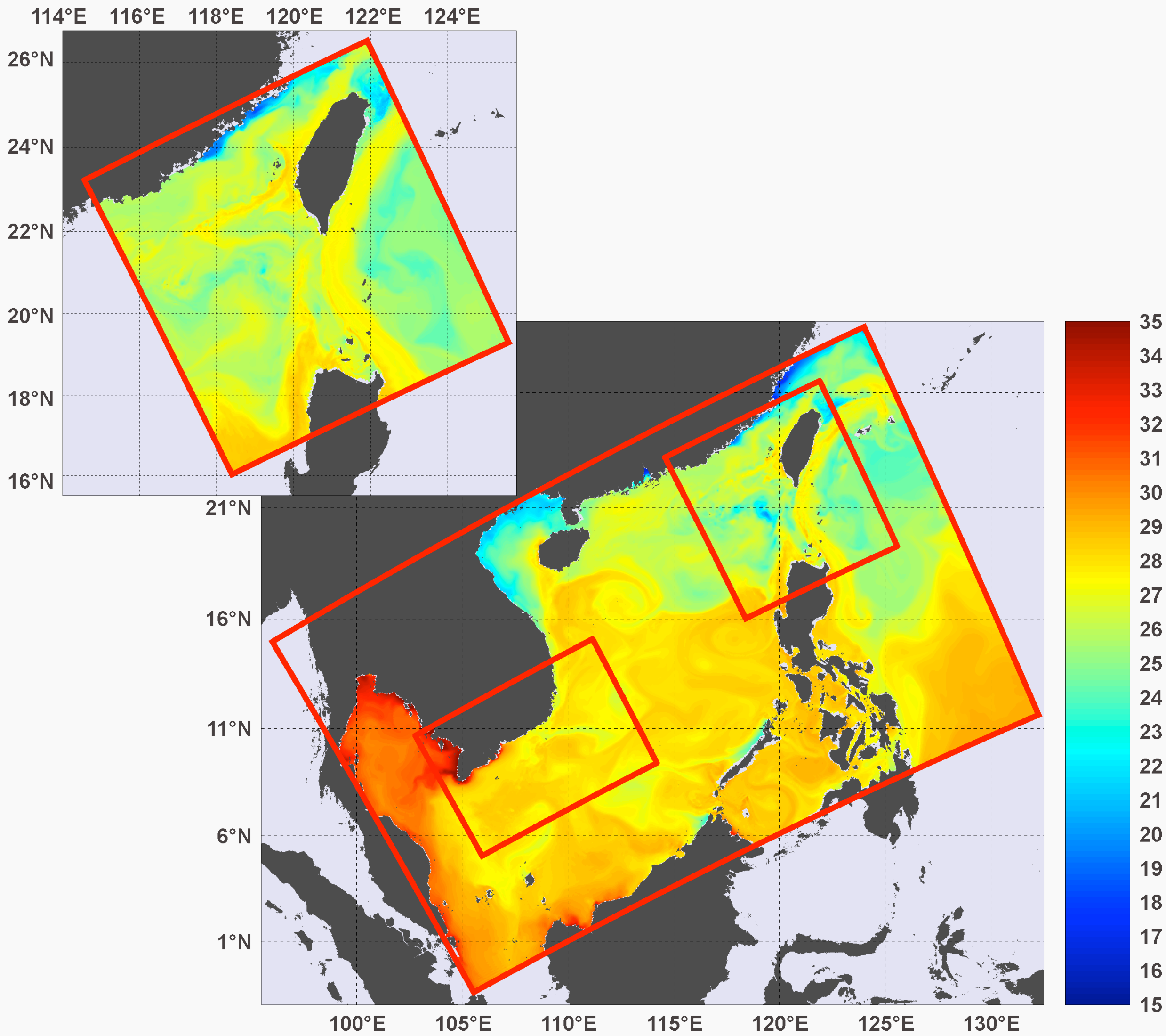
(Multiple Refinement Sub-Class)



Free-Surface

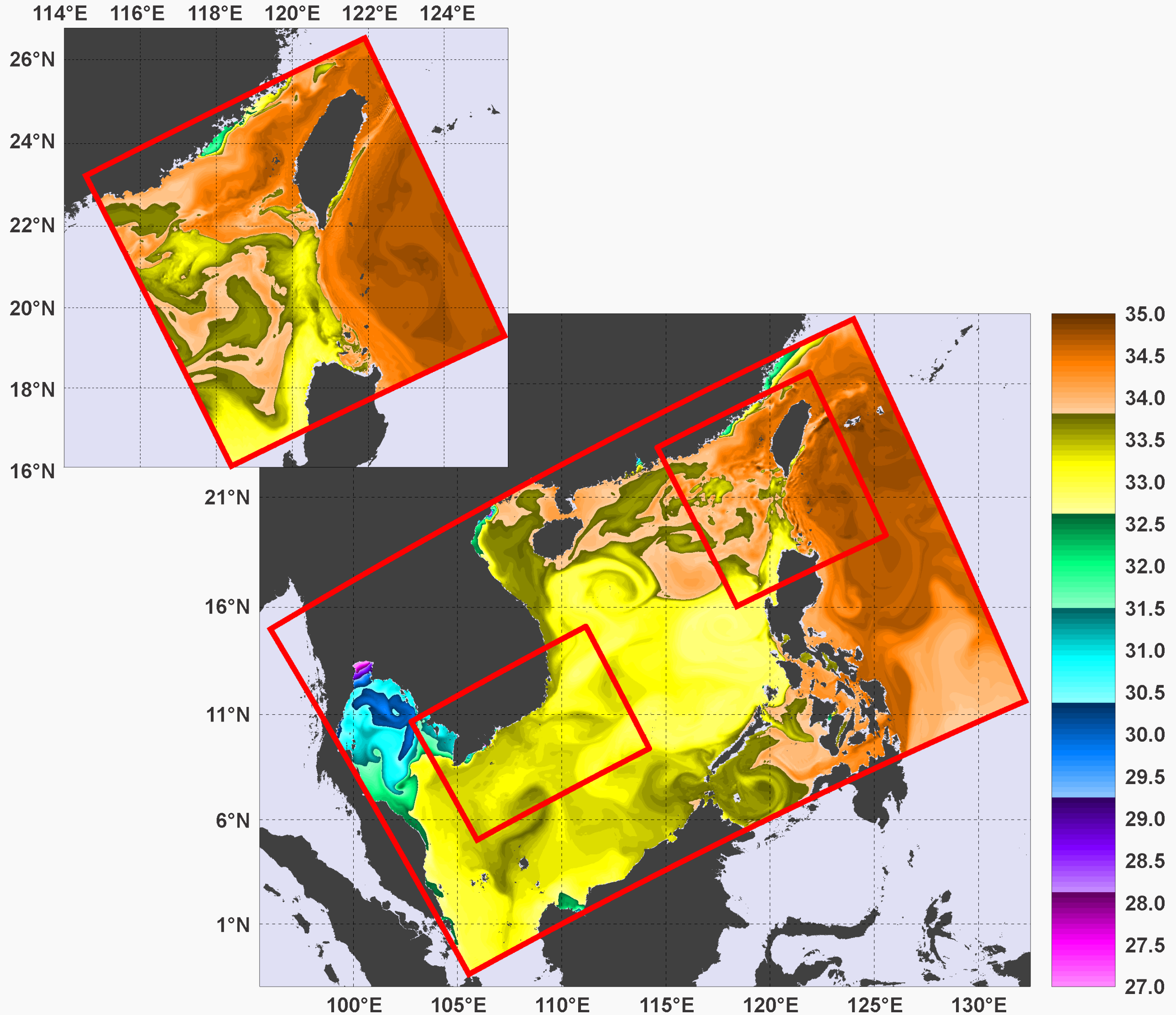
Realistic Nesting Configuration: South China Sea

(Multiple Refinement Sub-Class)



Temperature

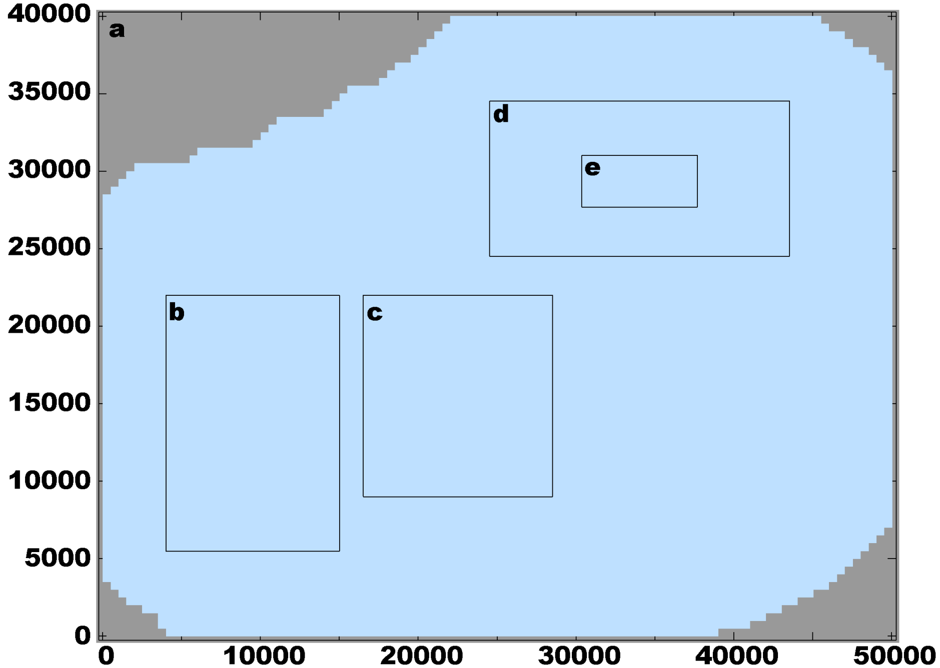
Realistic Nesting Configuration: South China Sea
(Multiple Refinement Sub-Class)



Salinity

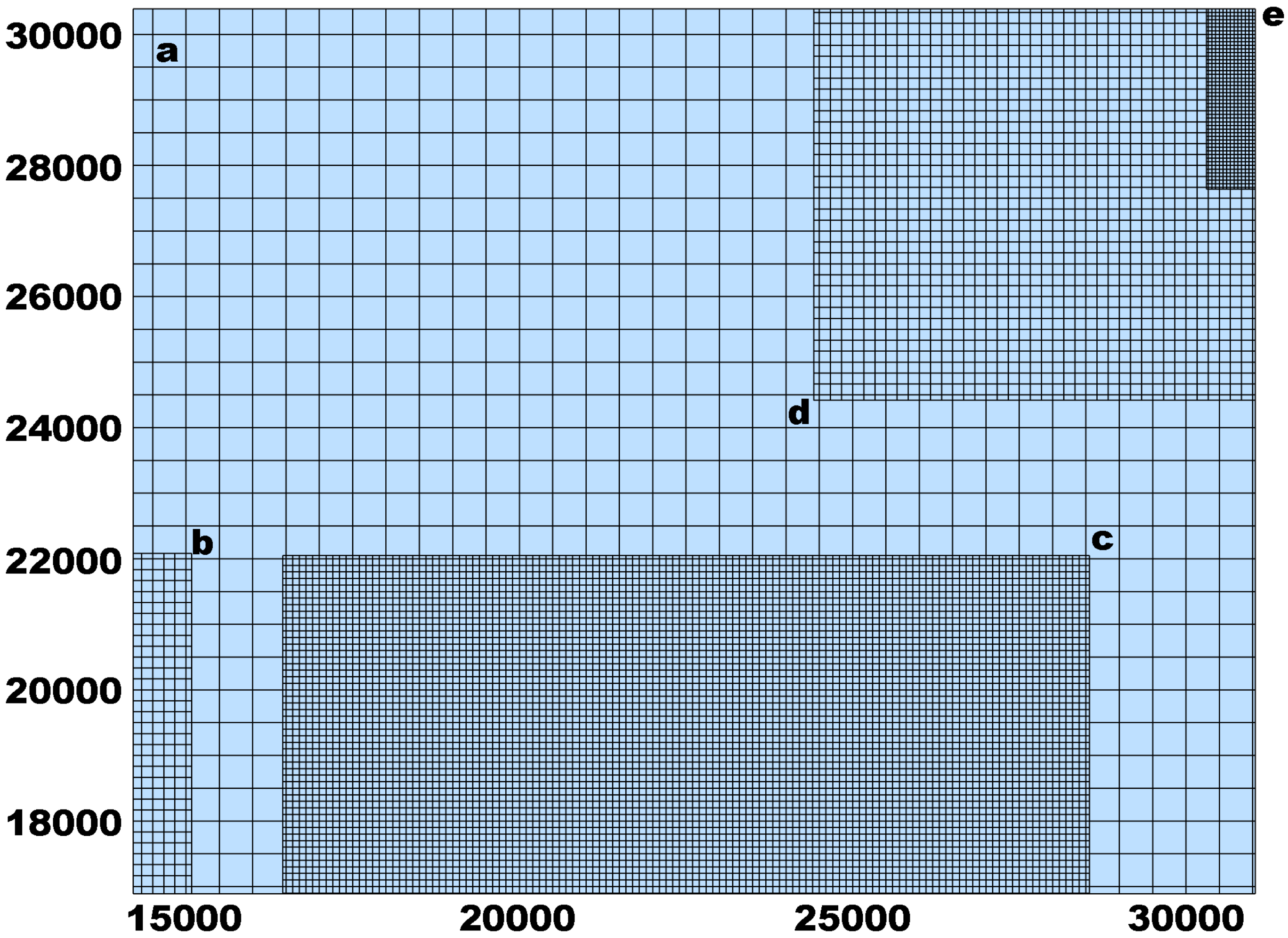
Lake Jersey Test Case

Lake Jersey



Lake Jersey Grid Information								
Grid	**Mesh Size**	**Refinement Factor**	**Parent Imin**	**Parent Imax**	**Parent Jmin**	**Parent Jmax**	**Δx Δy**	**Grid NetCDF File**
a	100x80x8	-	-	-	-	-	500.0 m	lake_jersey_grd_a.nc
b	66x99x8	1:3 from **a**	9	31	12	45	166.6 m	lake_jersey_grd_b.nc
c	120x130x8	1:5 from **a**	34	58	19	45	100.0 m	lake_jersey_grd_c.nc
d	114x60x8	1:3 from **a**	50	88	50	70	166.6 m	lake_jersey_grd_d.nc
e	132x60x8	1:3 from **d**	36	80	20	40	55.5 m	lake_jersey_grd_e.nc

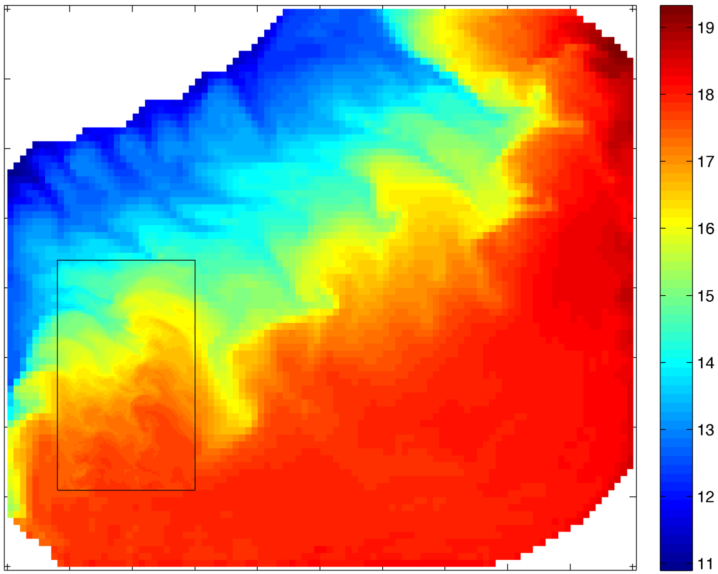
Lake Jersey



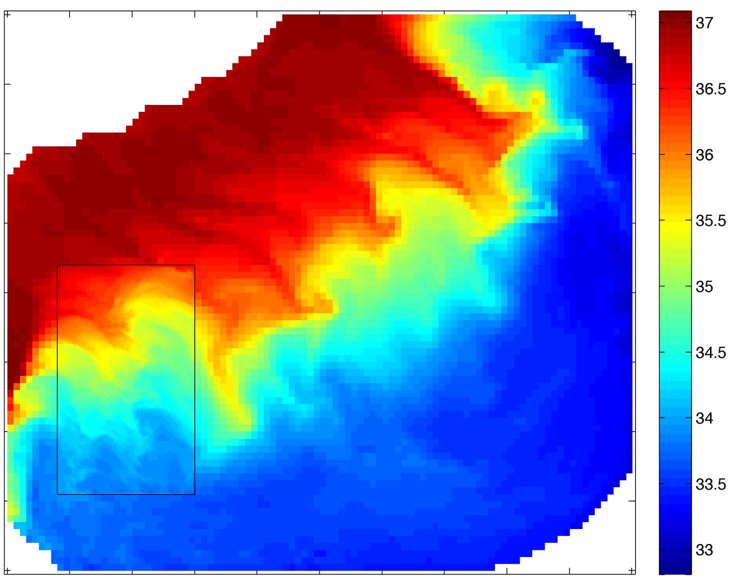
a
b
c
d
e
30000
28000
26000
24000
22000
20000
18000
15000
20000
25000
30000

Lake Jersey: Case AB

One-Way

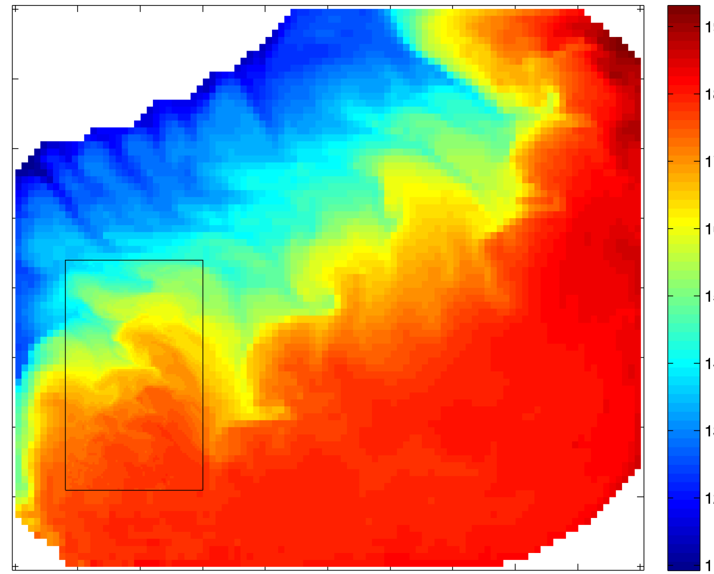


Temperature

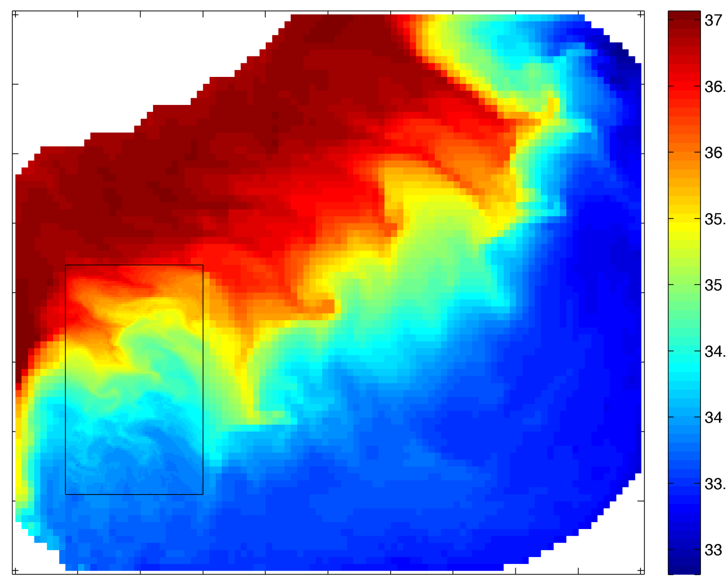


Salinity

Two-Way



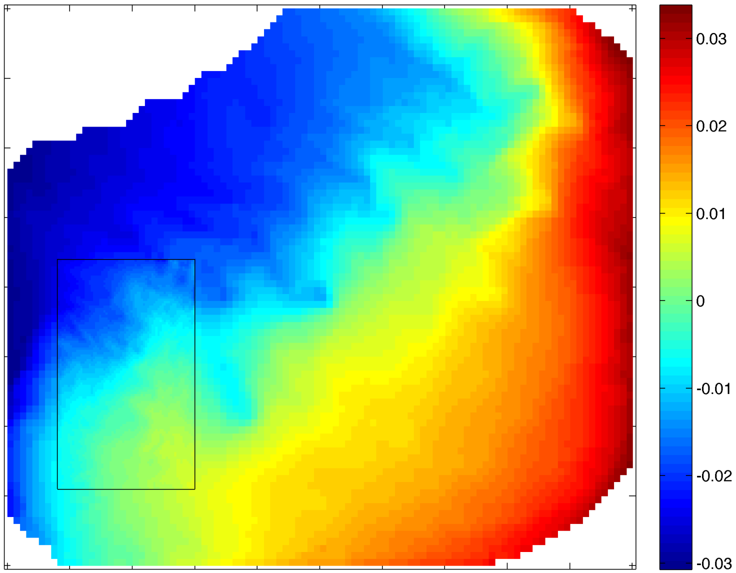
Temperature



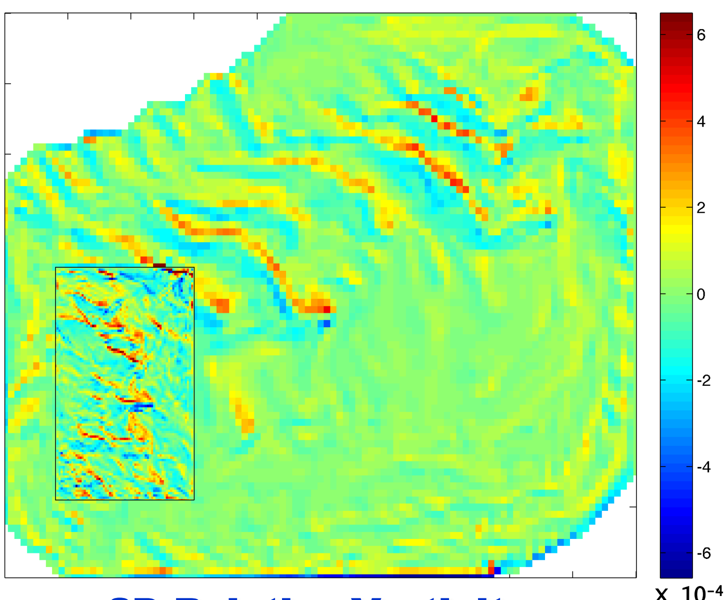
Salinity

Lake Jersey: Case AB

One-Way

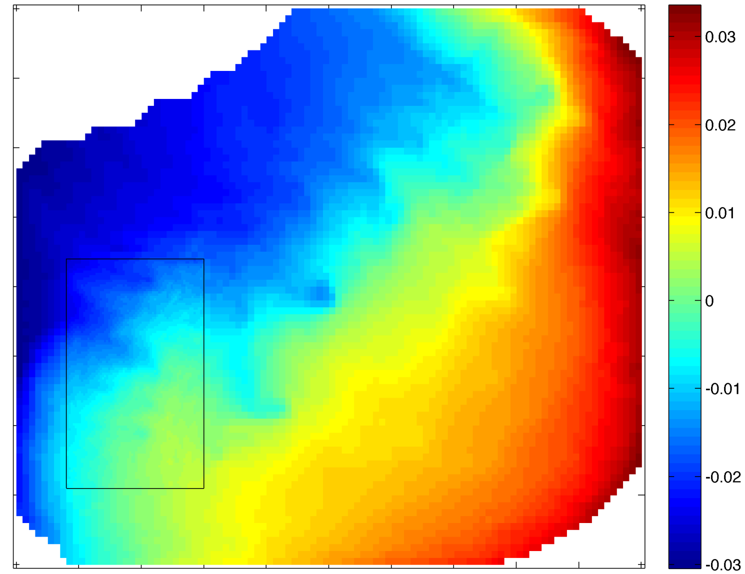


Free-Surface

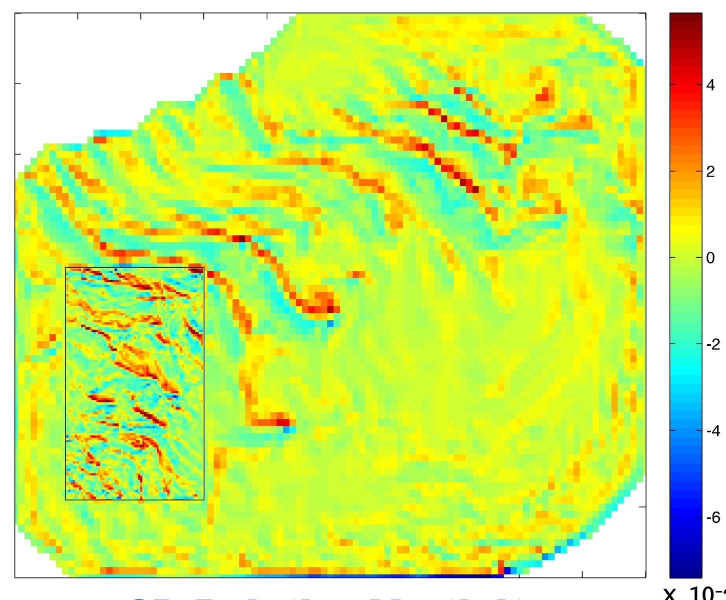


2D Relative Vorticity

Two-Way



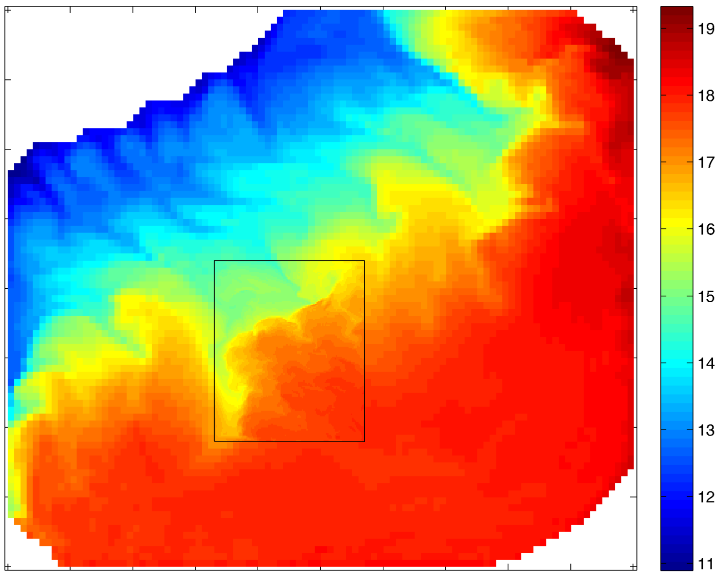
Free-Surface



2D Relative Vorticity

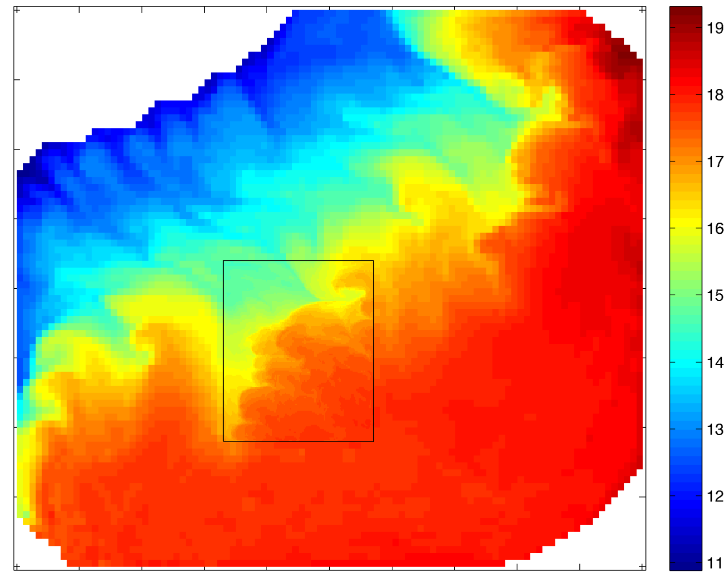
Lake Jersey: Case AC

One-Way

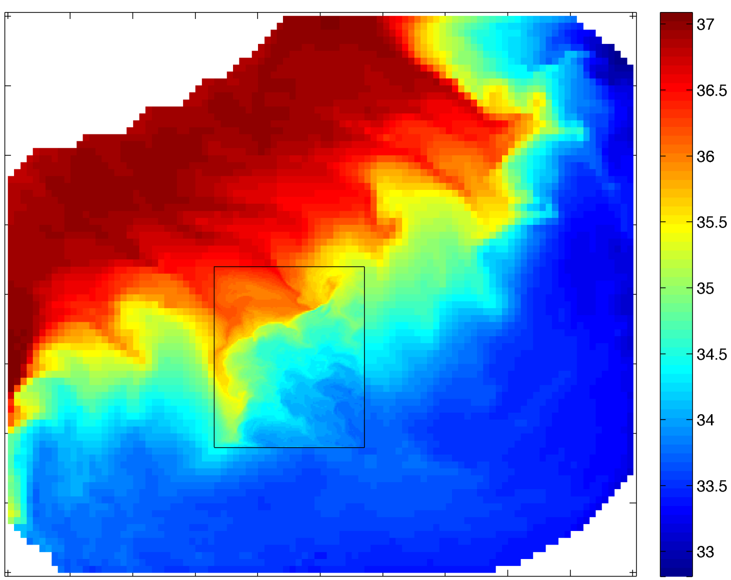


Temperature

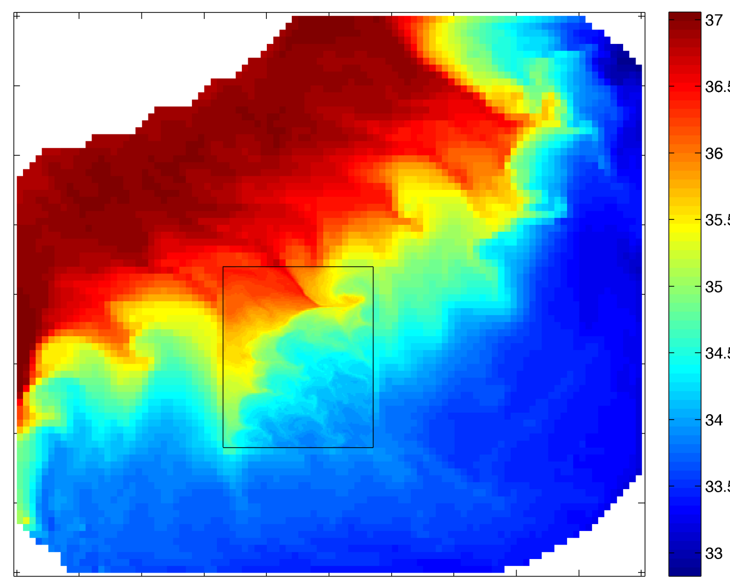
Two-Way



Temperature

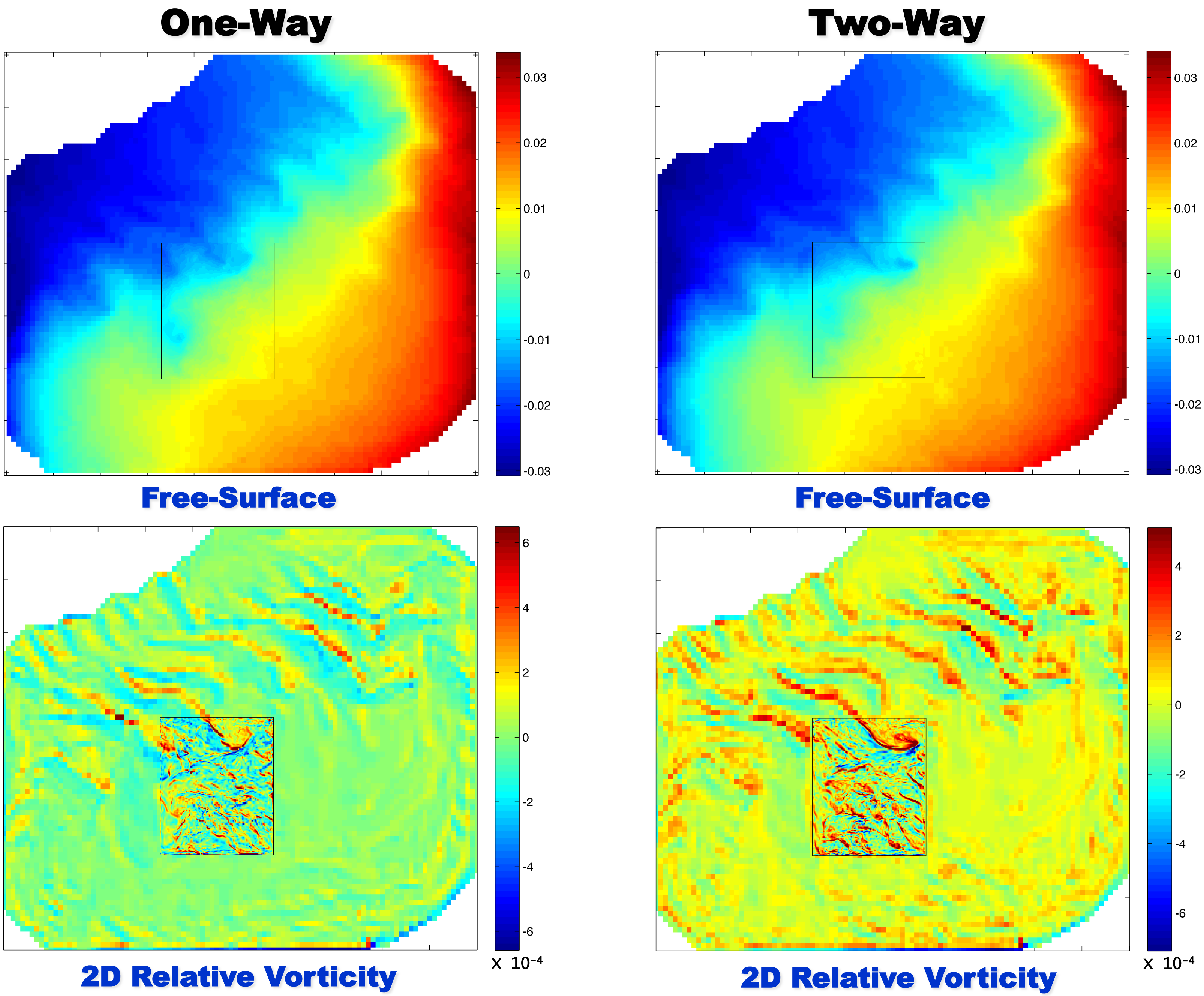


Salinity



Salinity

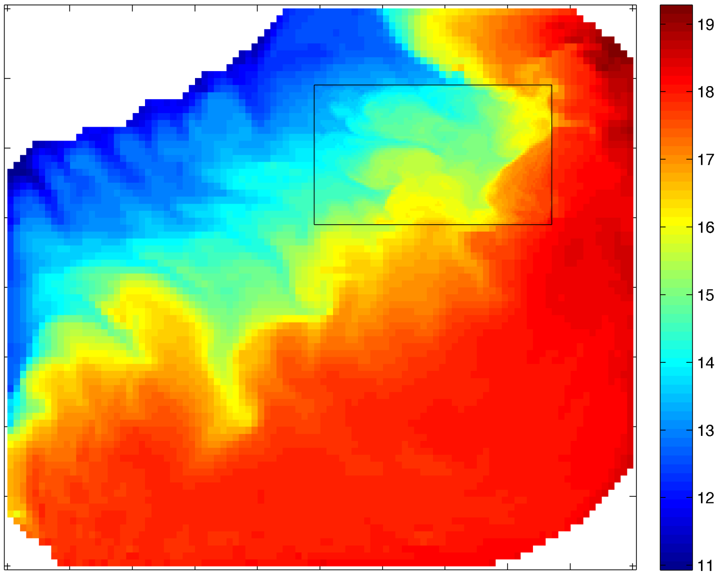
Lake Jersey: Case AC



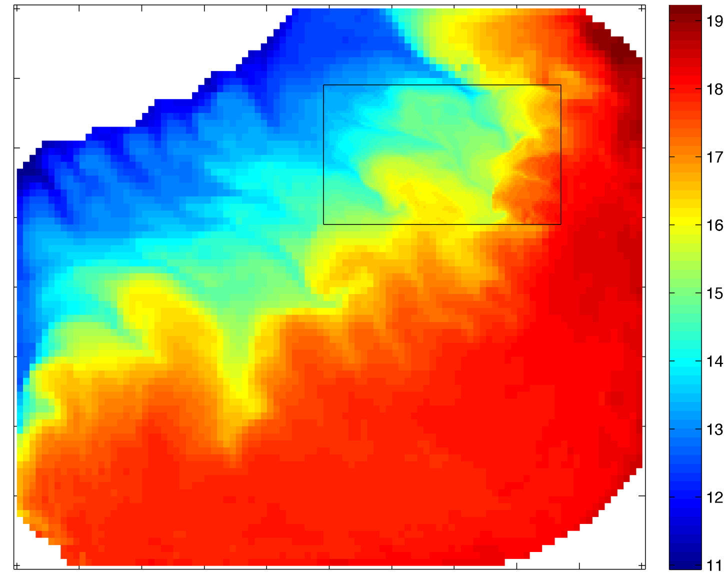
Lake Jersey: Case AD

One-Way

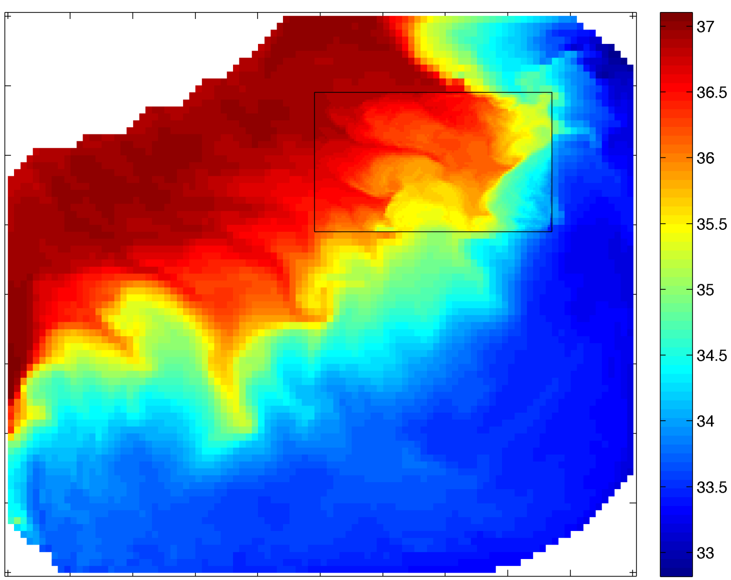
Two-Way



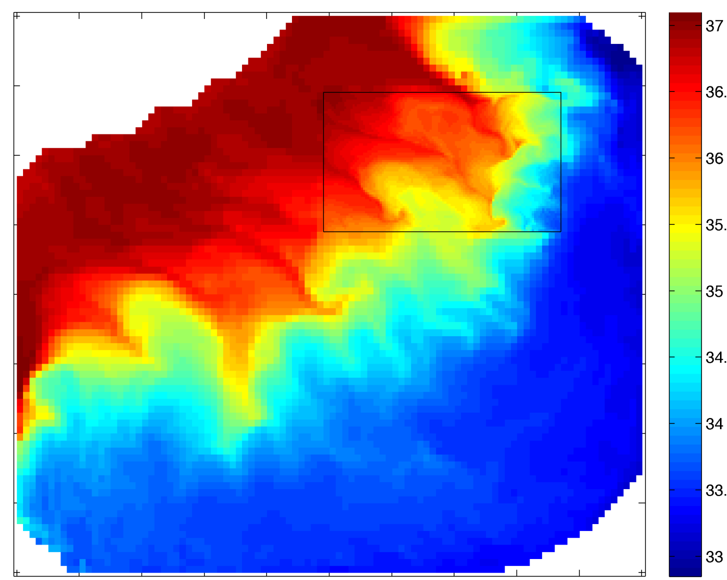
Temperature



Temperature

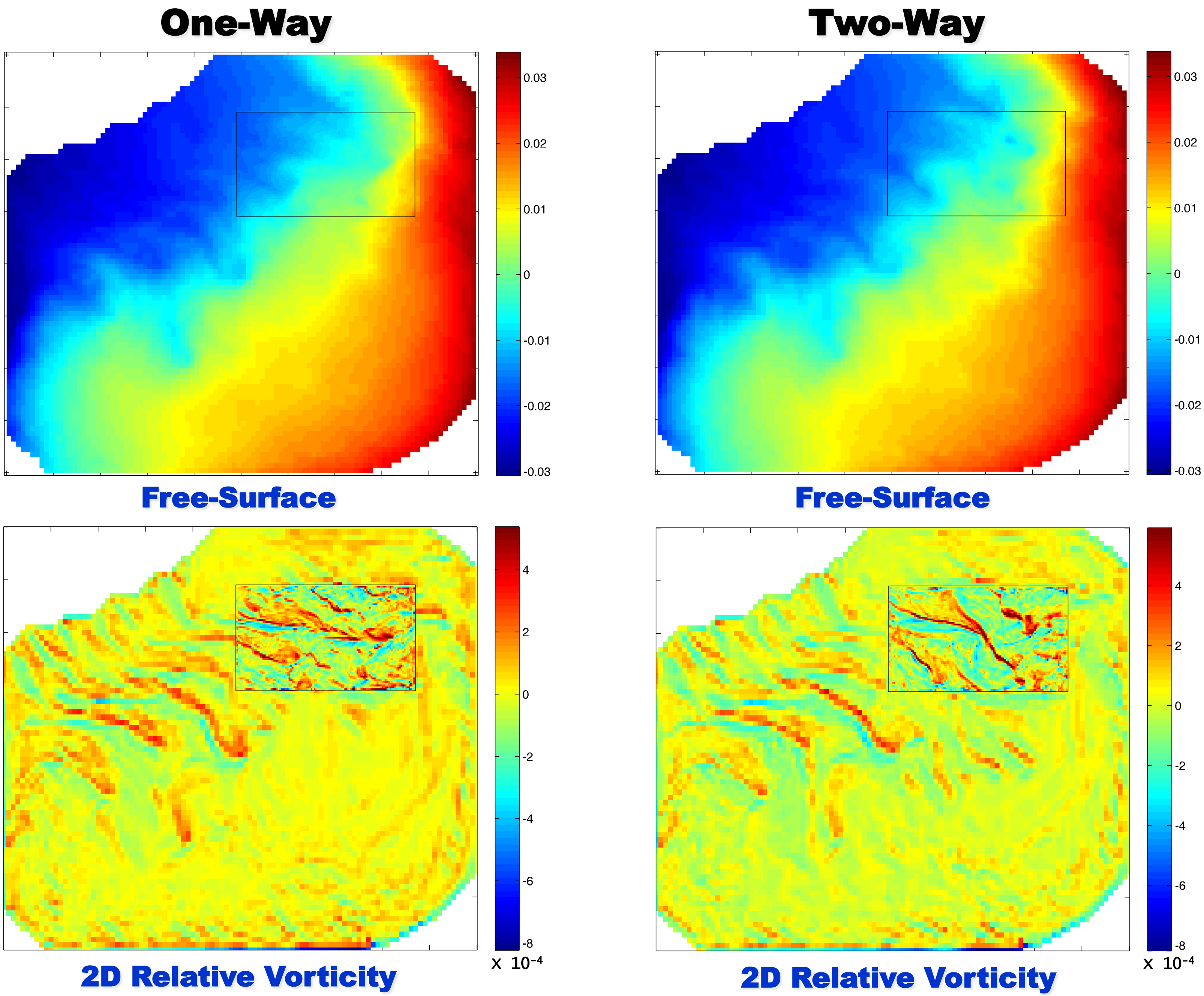


Salinity

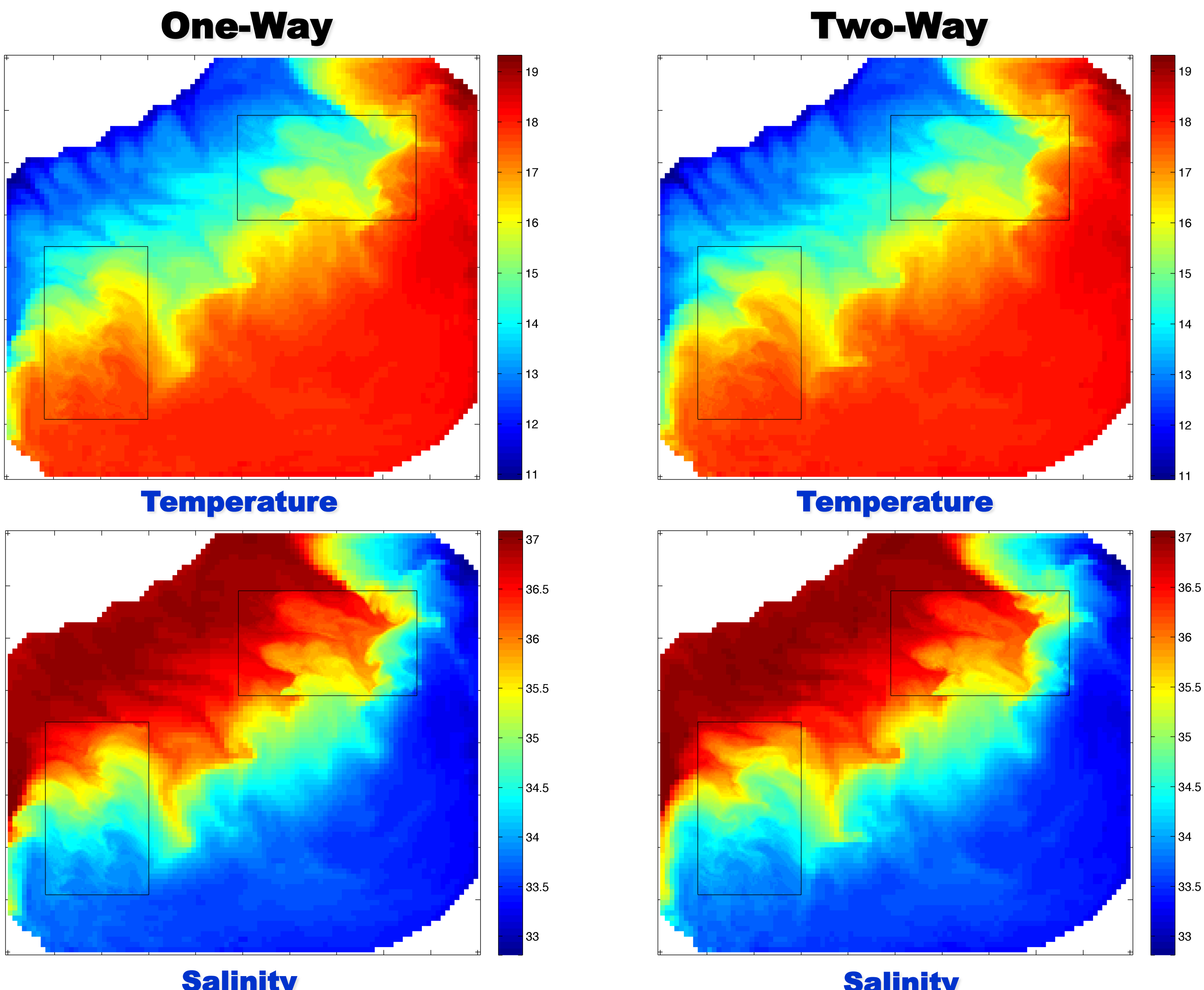


Salinity

Lake Jersey: Case AD

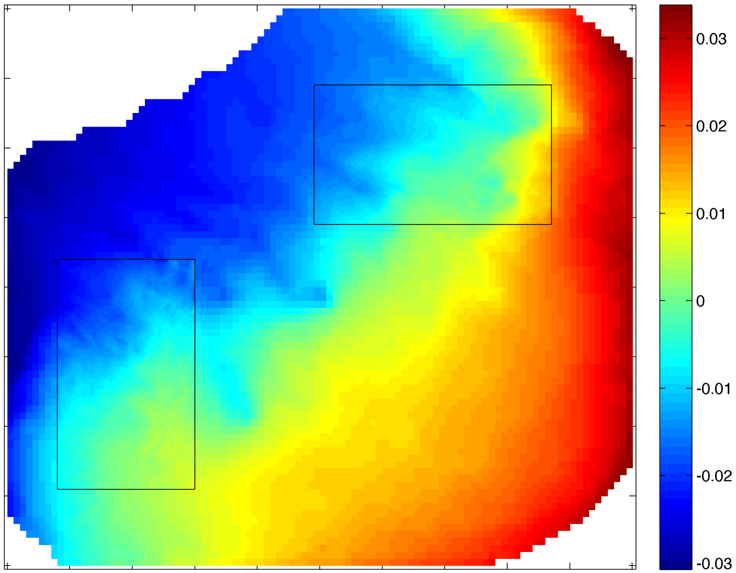


Lake Jersey: Case ABD

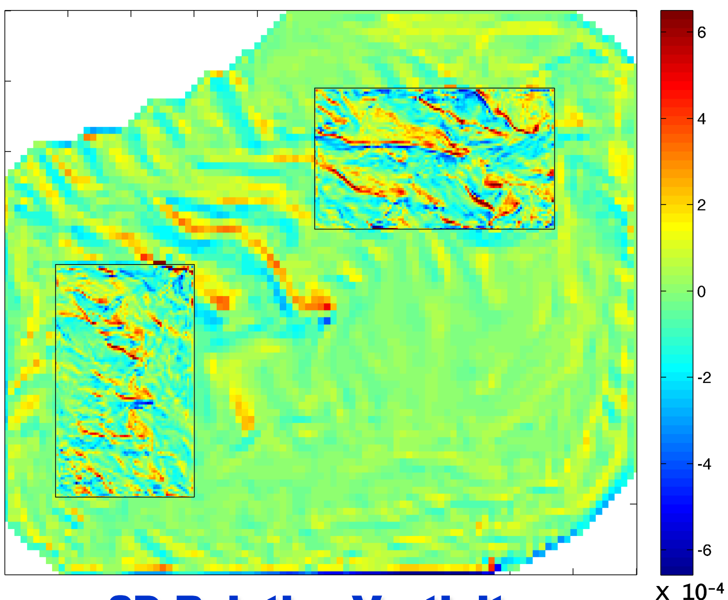


Lake Jersey: Case ABD

One-Way

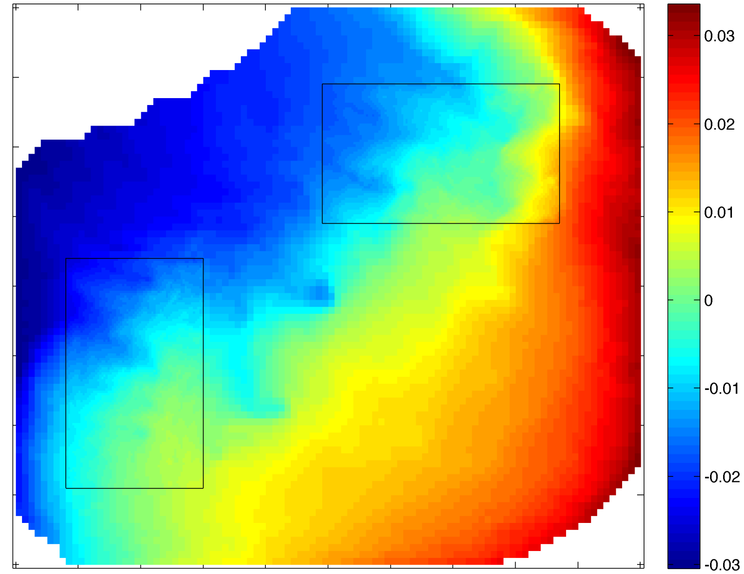


Free-Surface

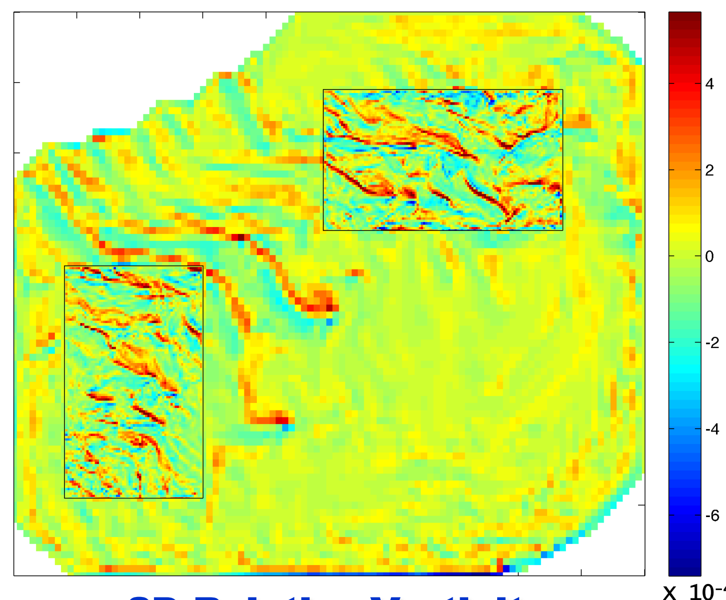


2D Relative Vorticity

Two-Way



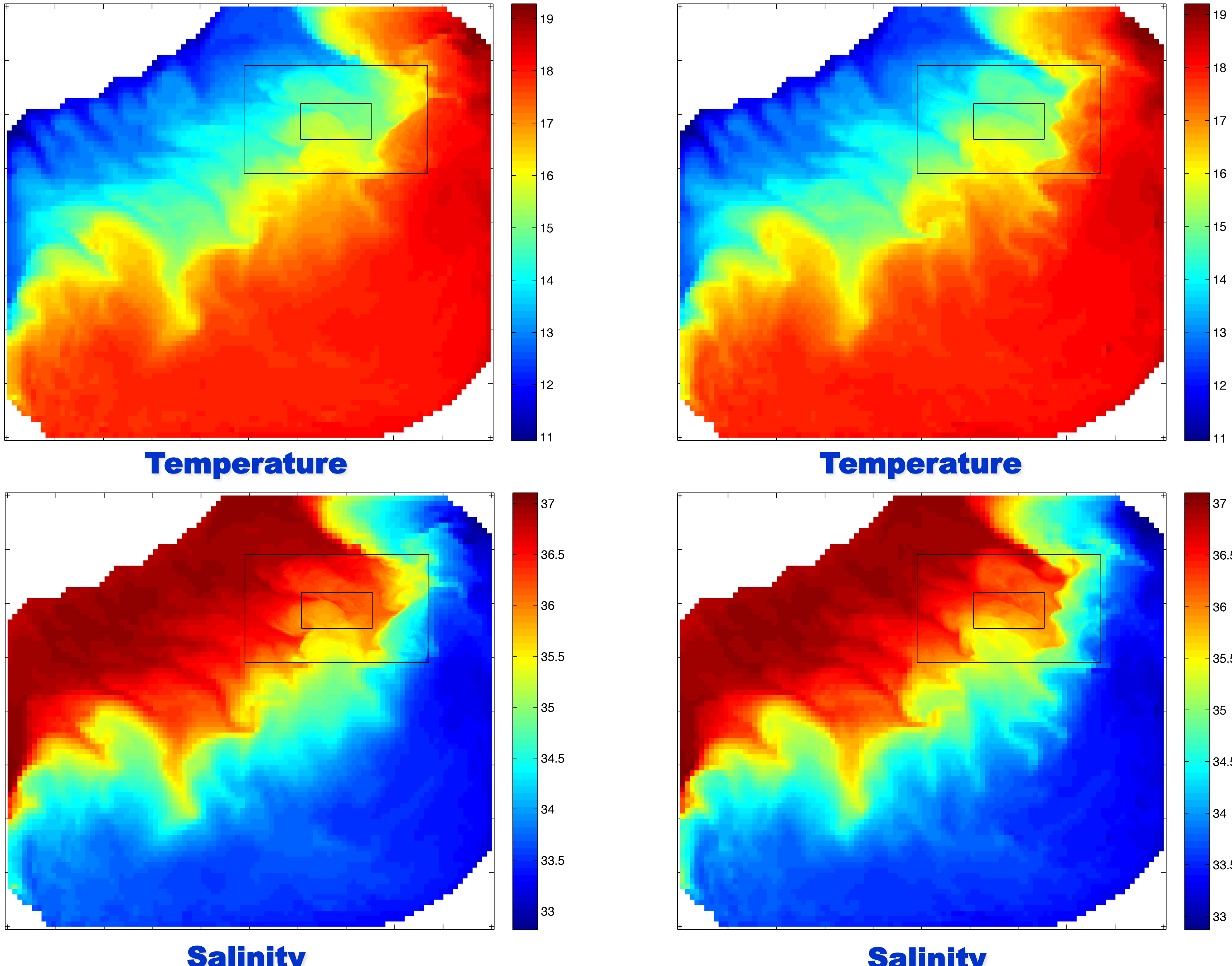
Free-Surface



2D Relative Vorticity

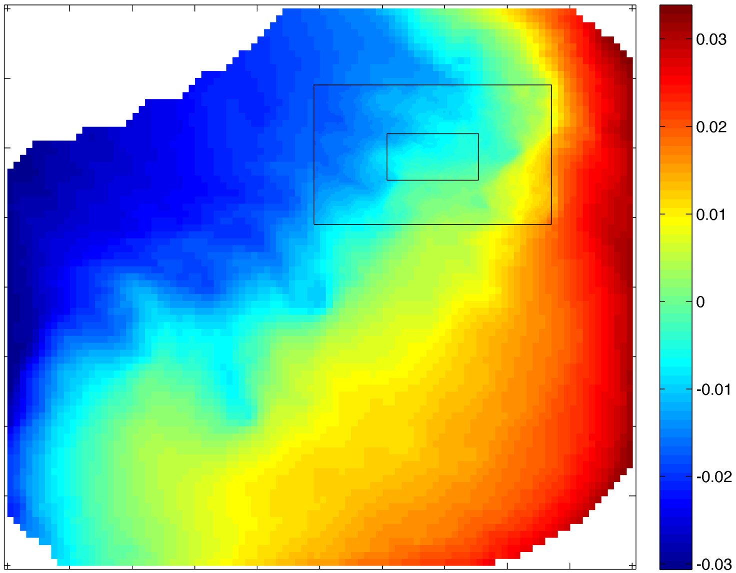
Lake Jersey: Case ADE

One-Way | **Two-Way**

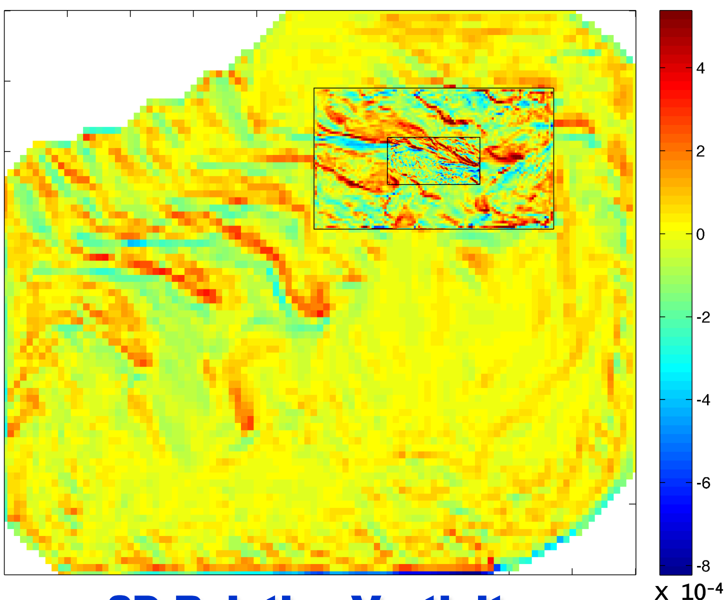


Lake Jersey: Case ADE

One-Way

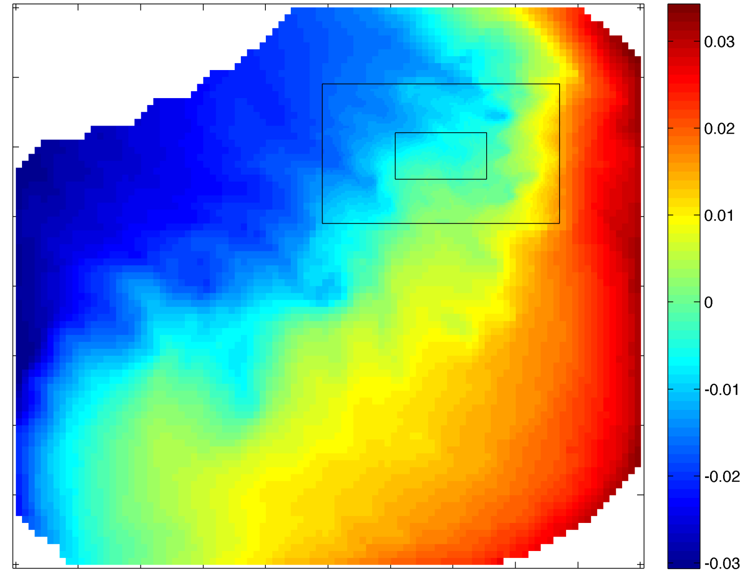


Free-Surface

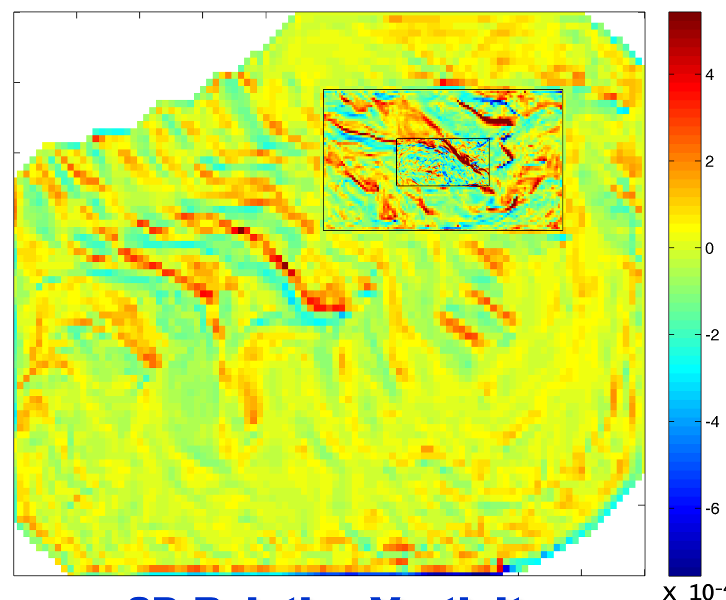


2D Relative Vorticity

Two-Way



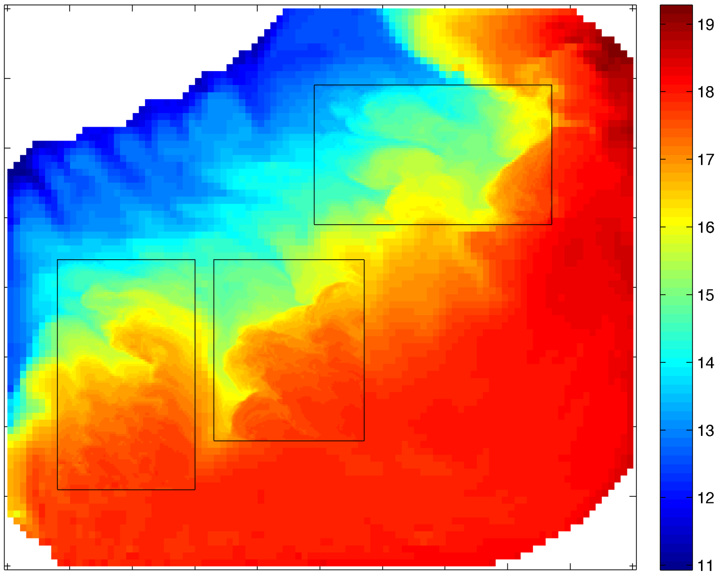
Free-Surface



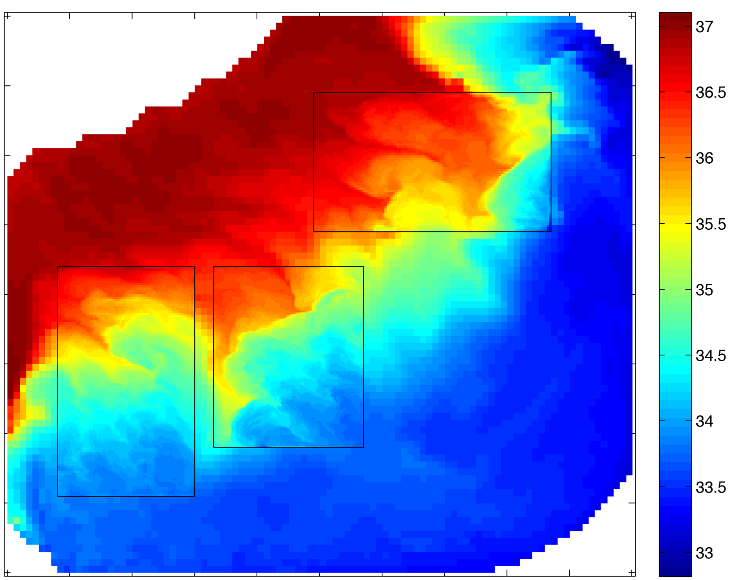
2D Relative Vorticity

Lake Jersey: Case ABDC

One-Way

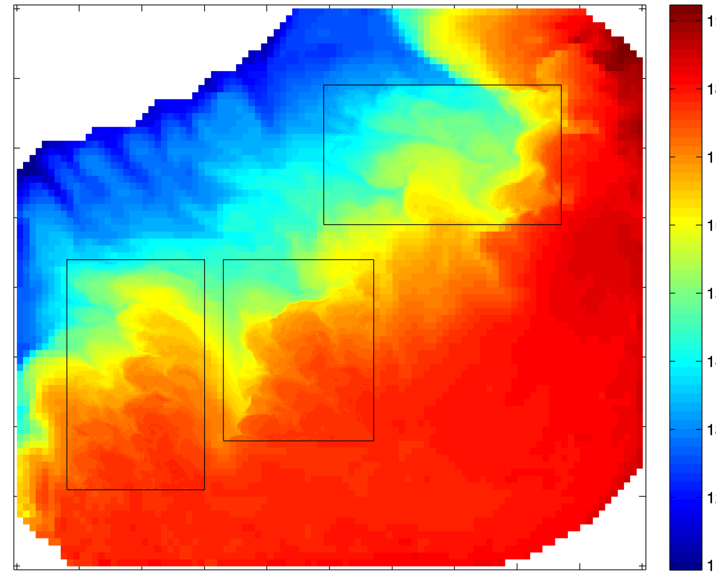


Temperature

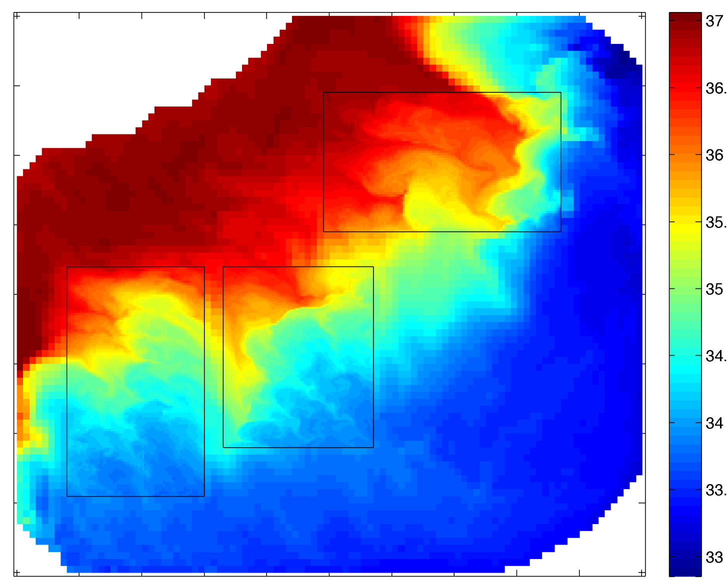


Salinity

Two-Way



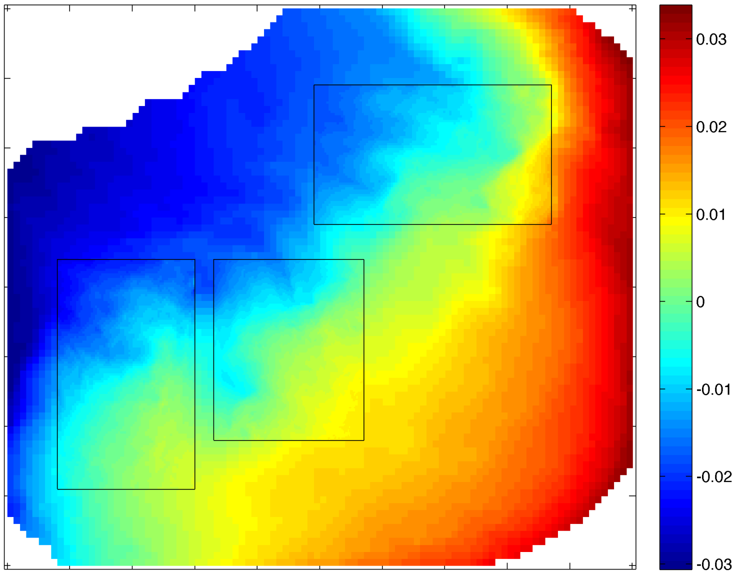
Temperature



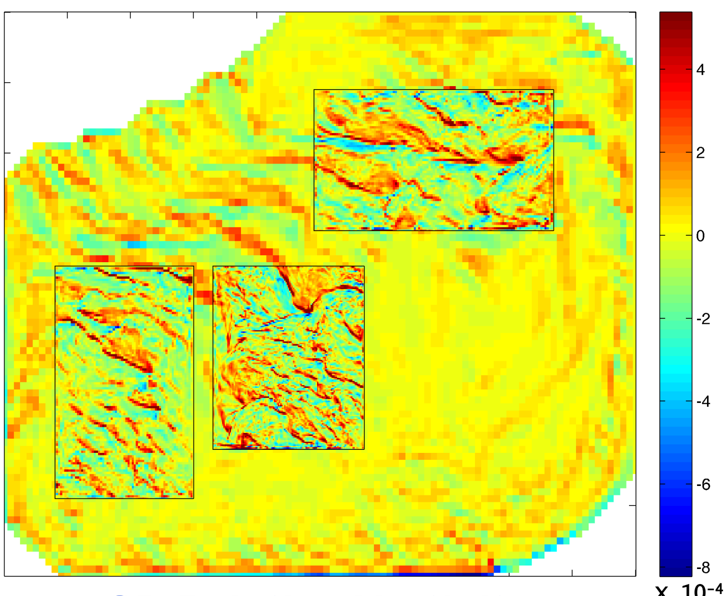
Salinity

Lake Jersey: Case ABDC

One-Way

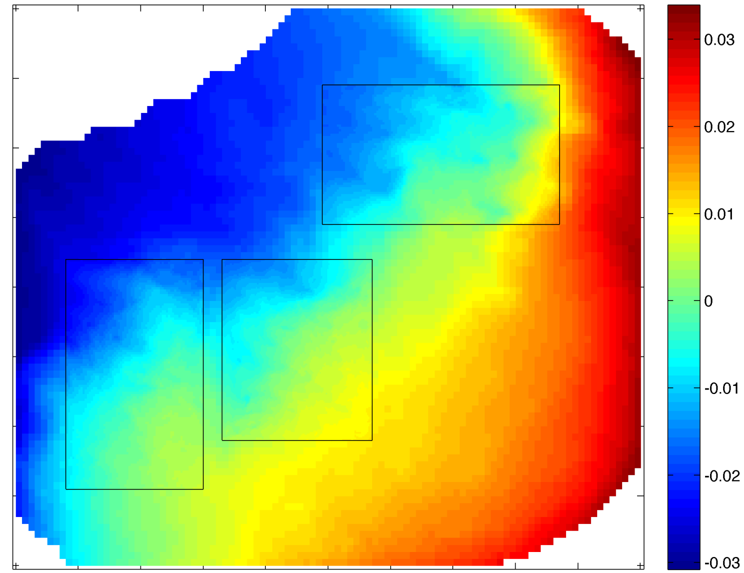


Free-Surface

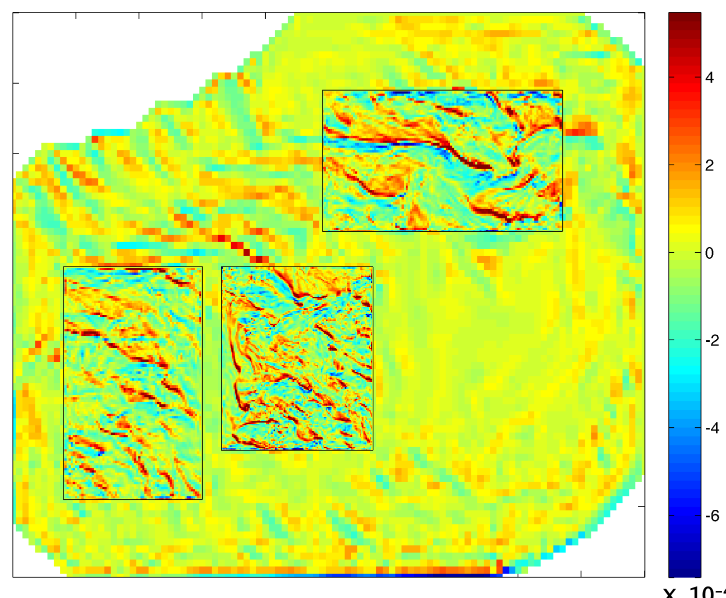


2D Relative Vorticity

Two-Way



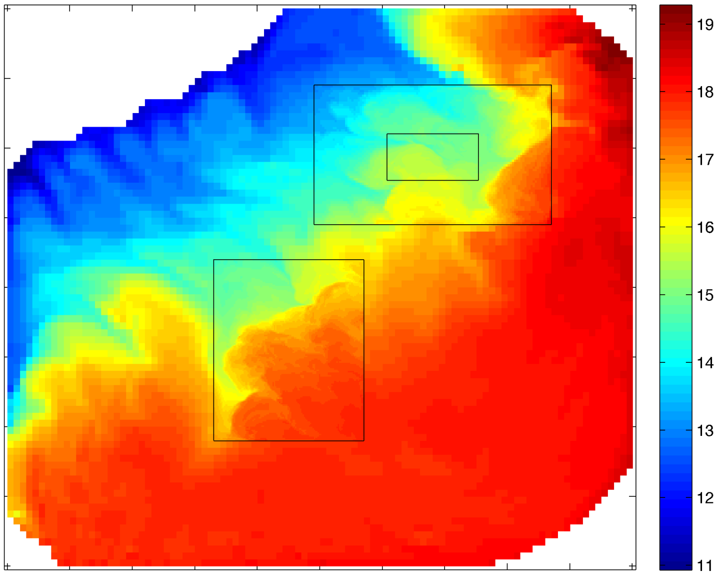
Free-Surface



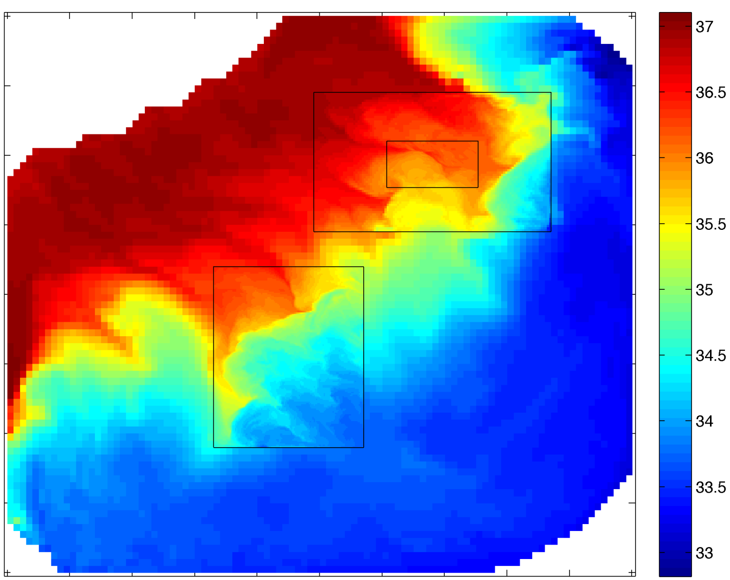
2D Relative Vorticity

Lake Jersey: Case ACDE

One-Way

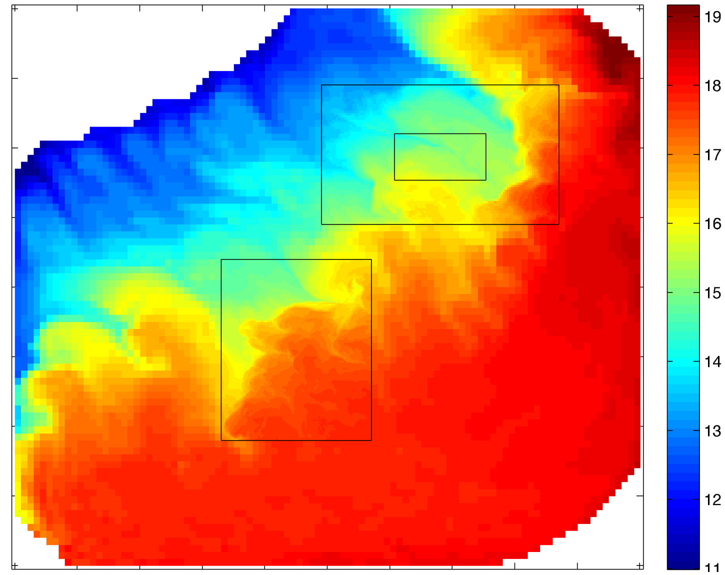


Temperature

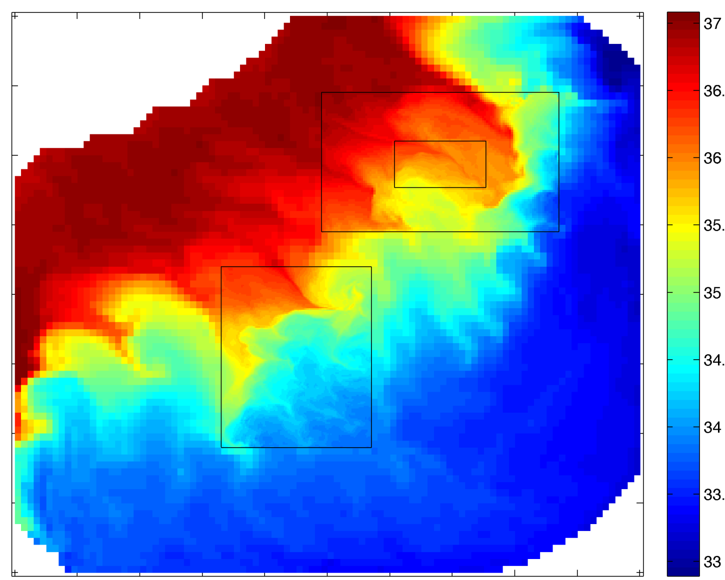


Salinity

Two-Way



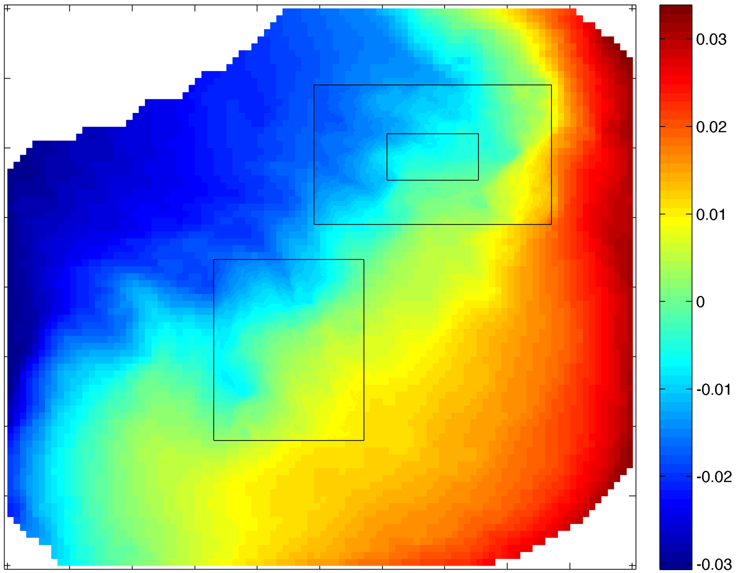
Temperature



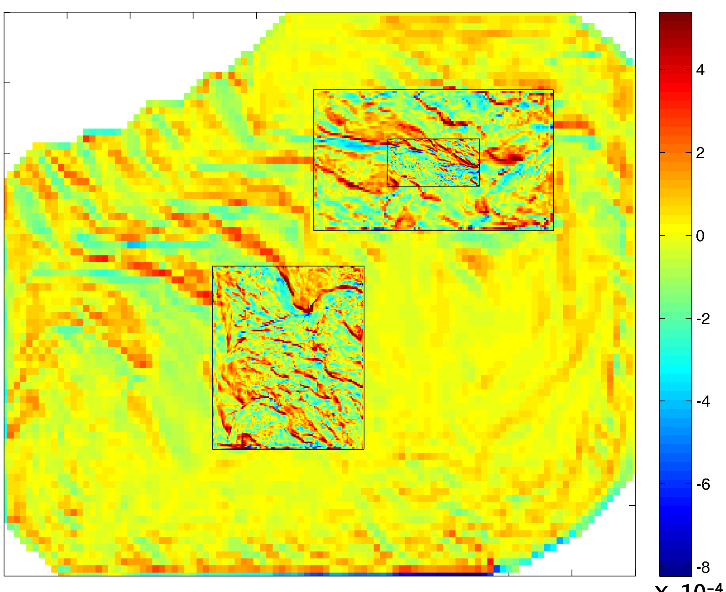
Salinity

Lake Jersey: Case ACDE

One-Way

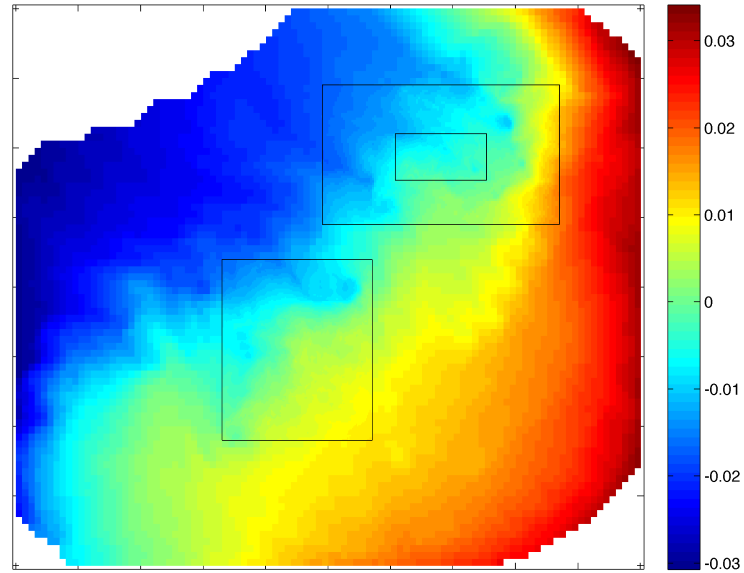


Free-Surface

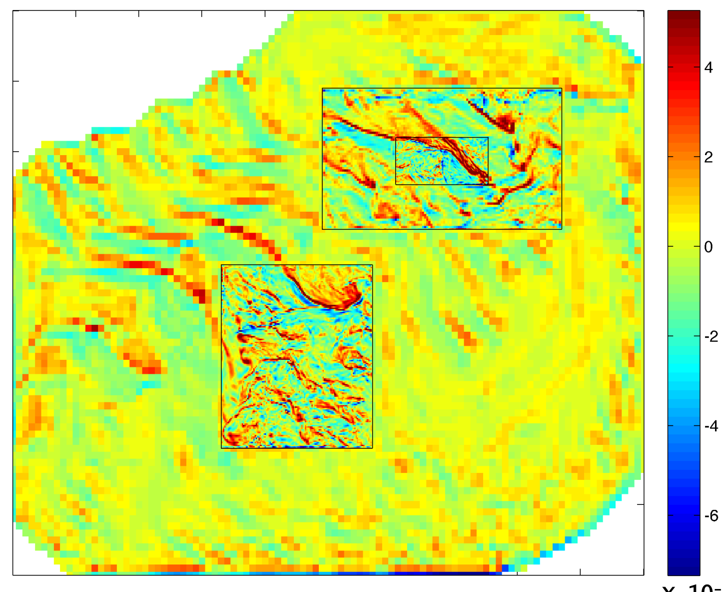


2D Relative Vorticity

Two-Way



Free-Surface

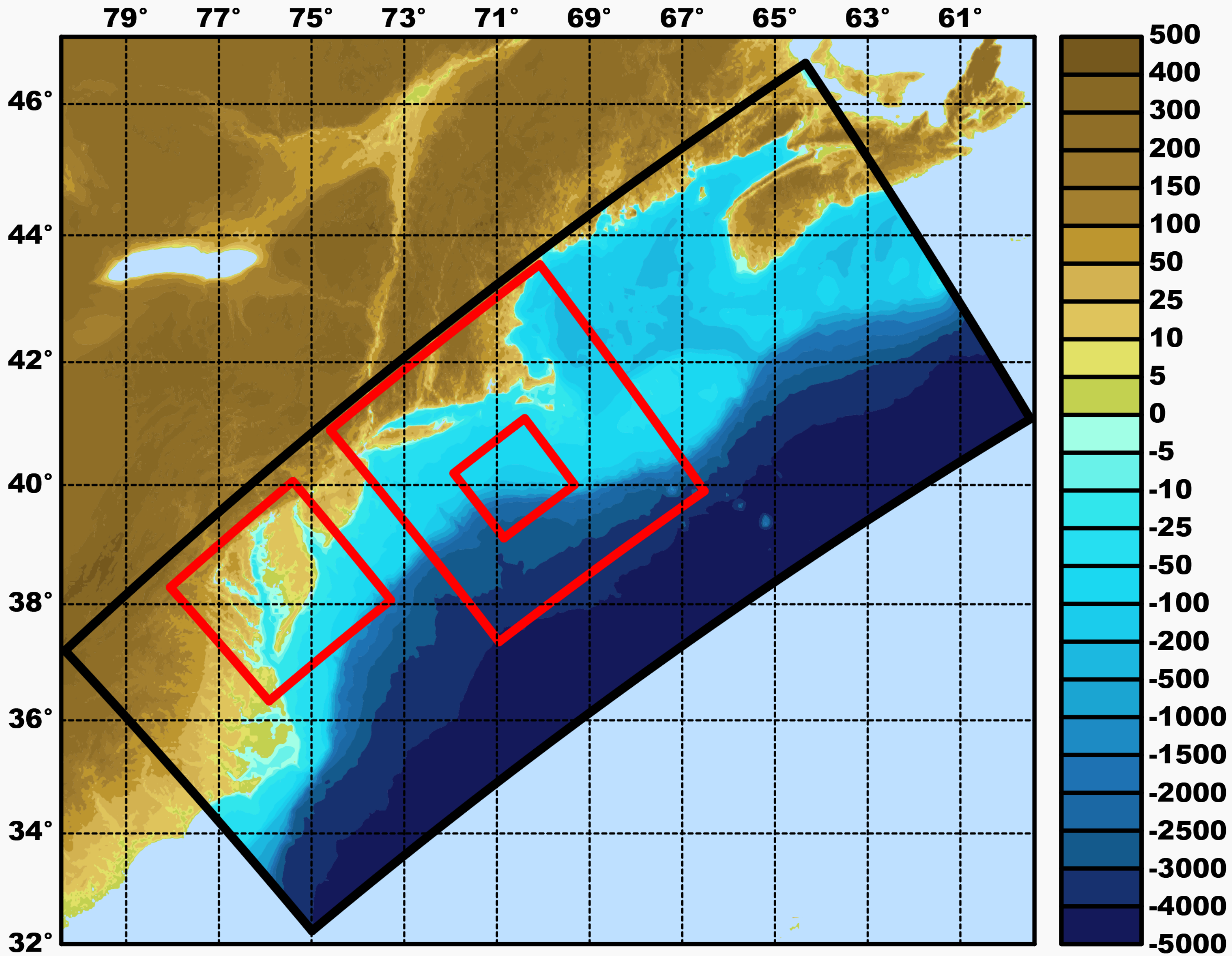


2D Relative Vorticity

Mid-Atlantic Bight Grids

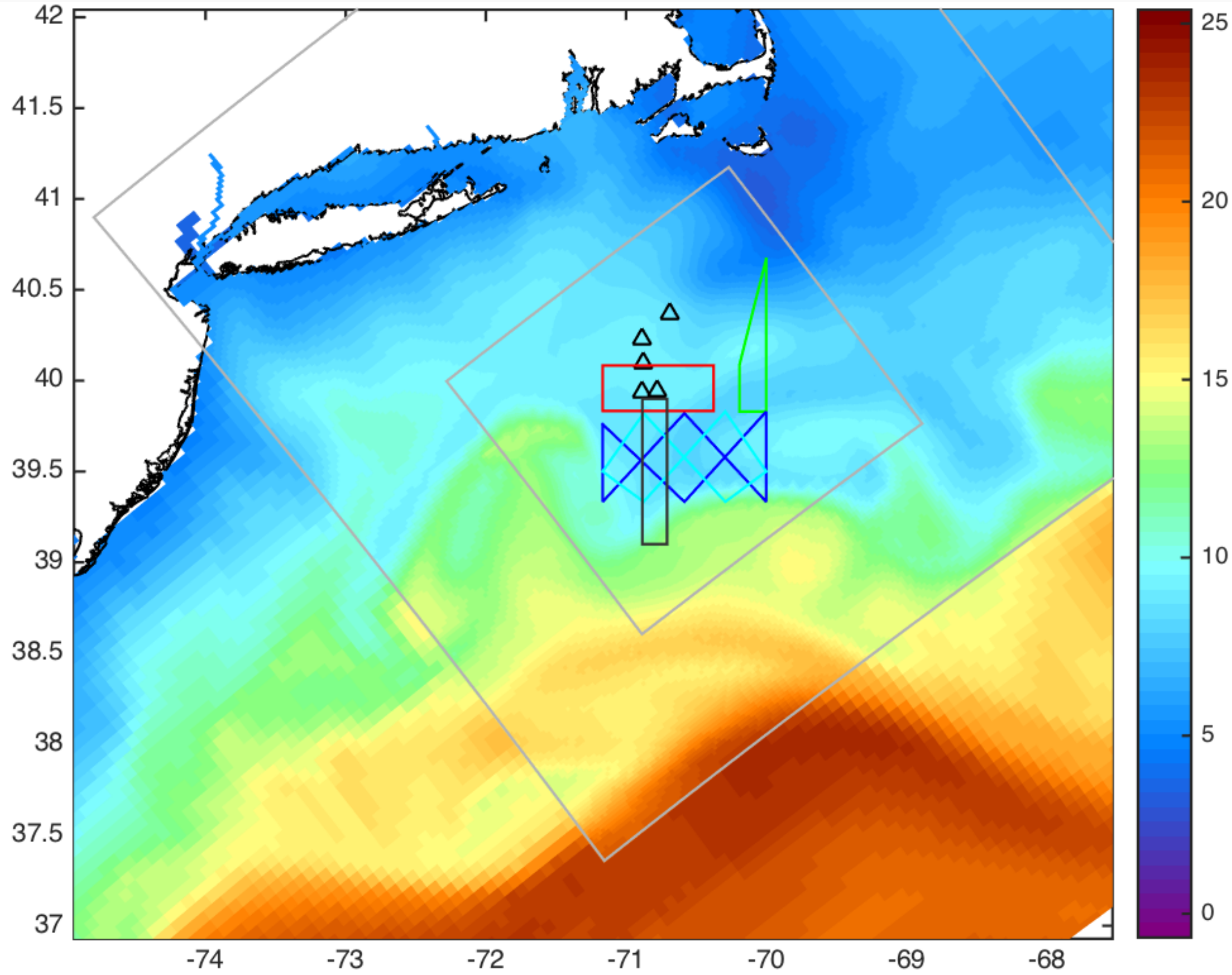
- **DOPPIO:** 242x106 (~7 km)
- **Chesapeake-Delaware grid (1:5 ratio):** 215x205 (~1.8 km)
- **Chesapeake-Delaware grid (1:7 ratio):** 303x289 (~1.2 km)
- **PIONEER (Hudson Canyon) grid (1:3 ratio):** 204x216 (~2.3 km)
- **ARRAY (Pioneer Data Array) grid (1:3; 1:9 ratio):** 210x195 (~1 km)

DOPPIO Refinement Grids



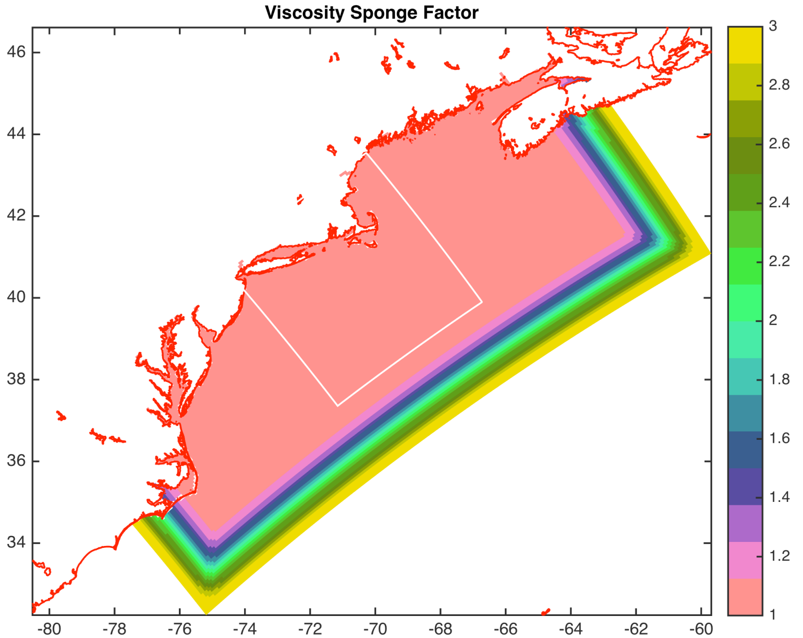
Bathymetry (m)

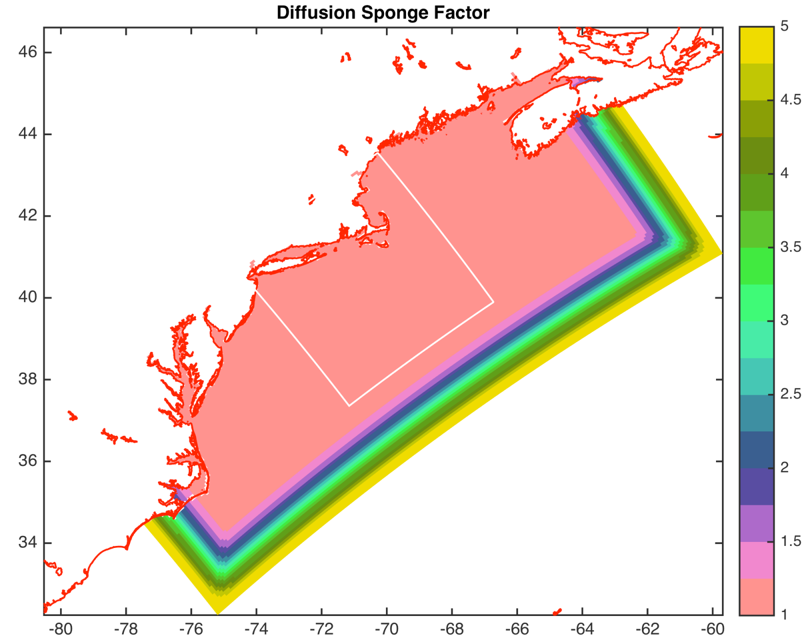
PIONEER Data Array Observations



Initial Surface Temperature (01-Jan-2014)

DOPPIO Sponge Areas





Same Reynolds Number for all Grids (U Δx/ν):

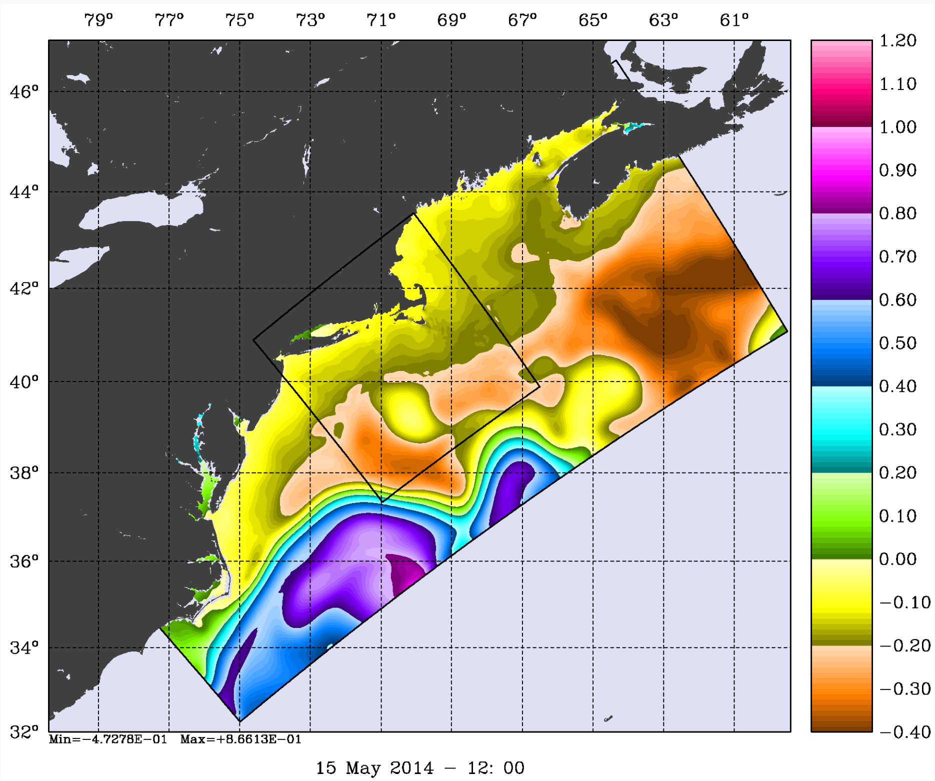
```
VISC2 == 90.0d0   30.0d0   10.0d0              (m2/s)
 TNU2 == 2*27.0d0   2*9.0d0     2*3.0d0        (m2/s)
```

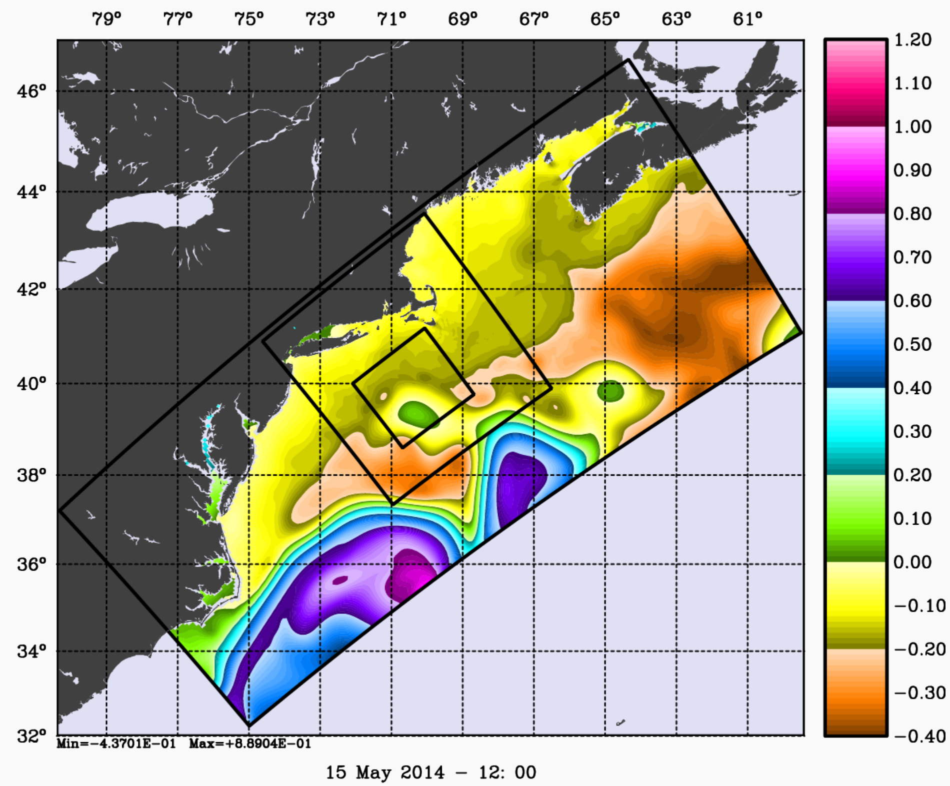
Time stepping (Δt):

```
DT == 180.0d0   90.0d0    45.0d0      (3/1.5/0.75 minute step)
```

Two-Way DOPPIO-PIONEER-ARRAY

(1:3 Refinement)

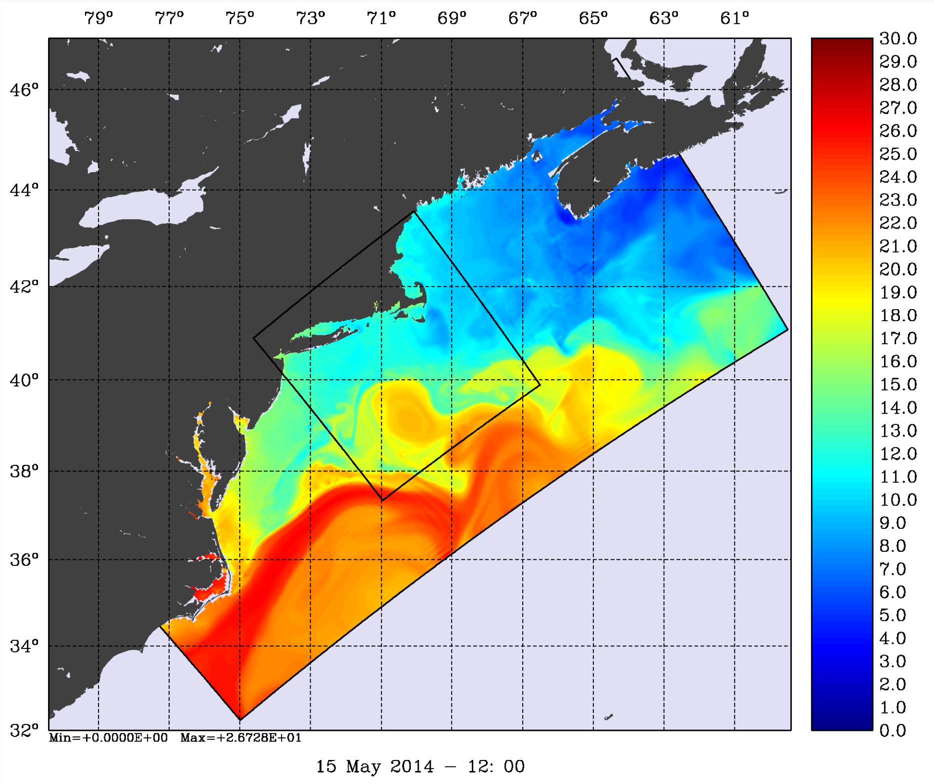


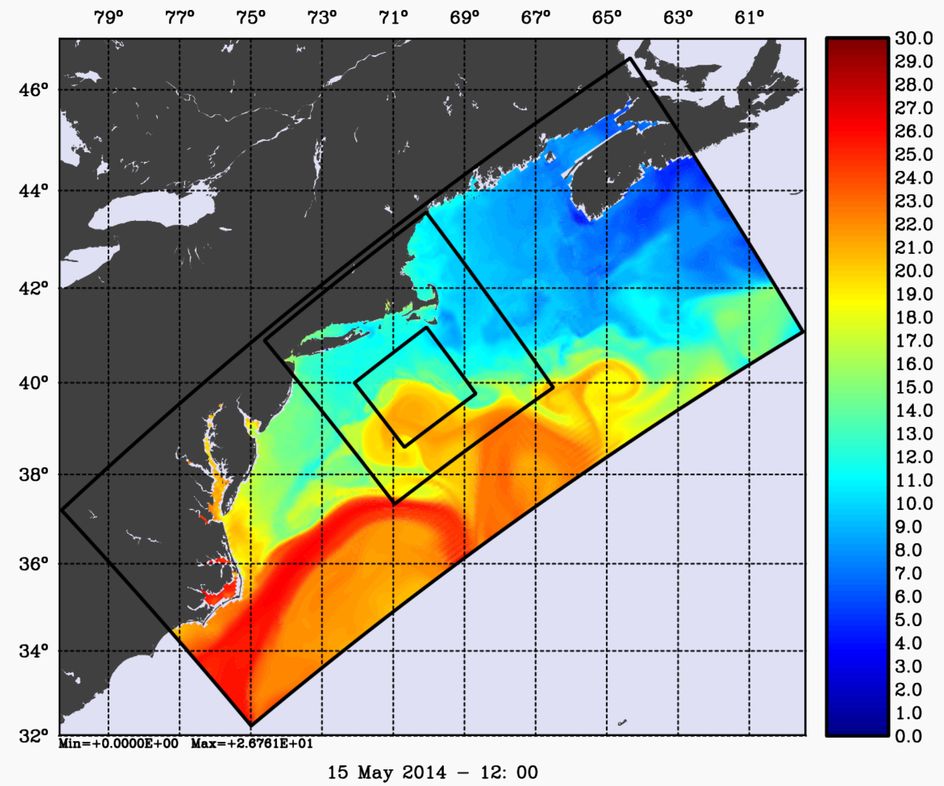


Free-Surface

Two-Way DOPPIO-PIONEER-ARRAY

(1:3 Refinement)

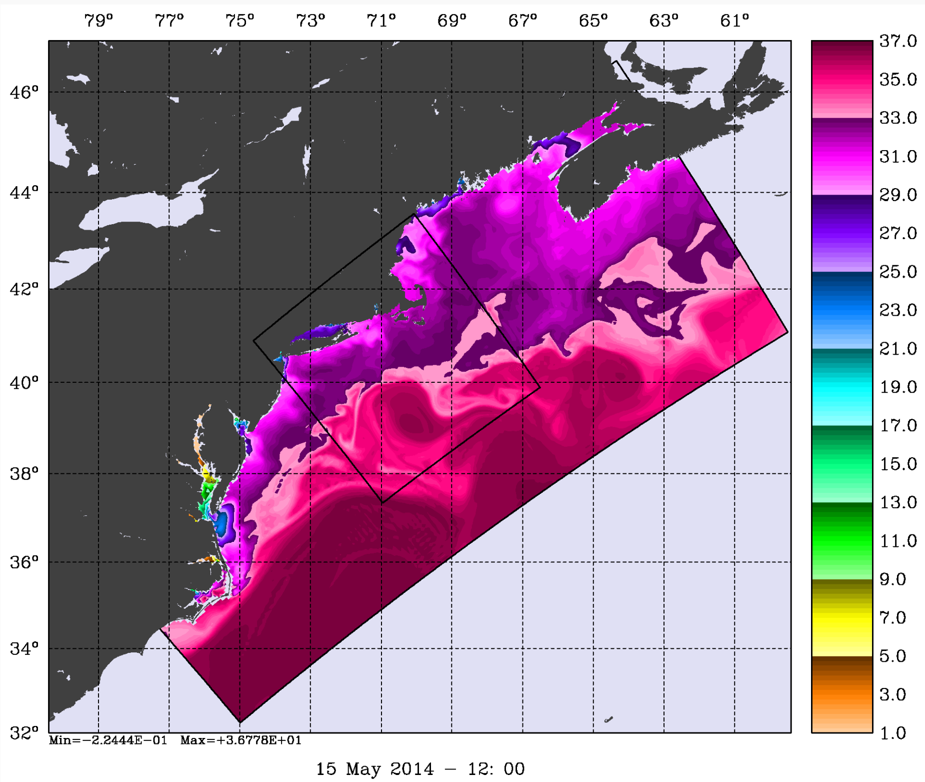


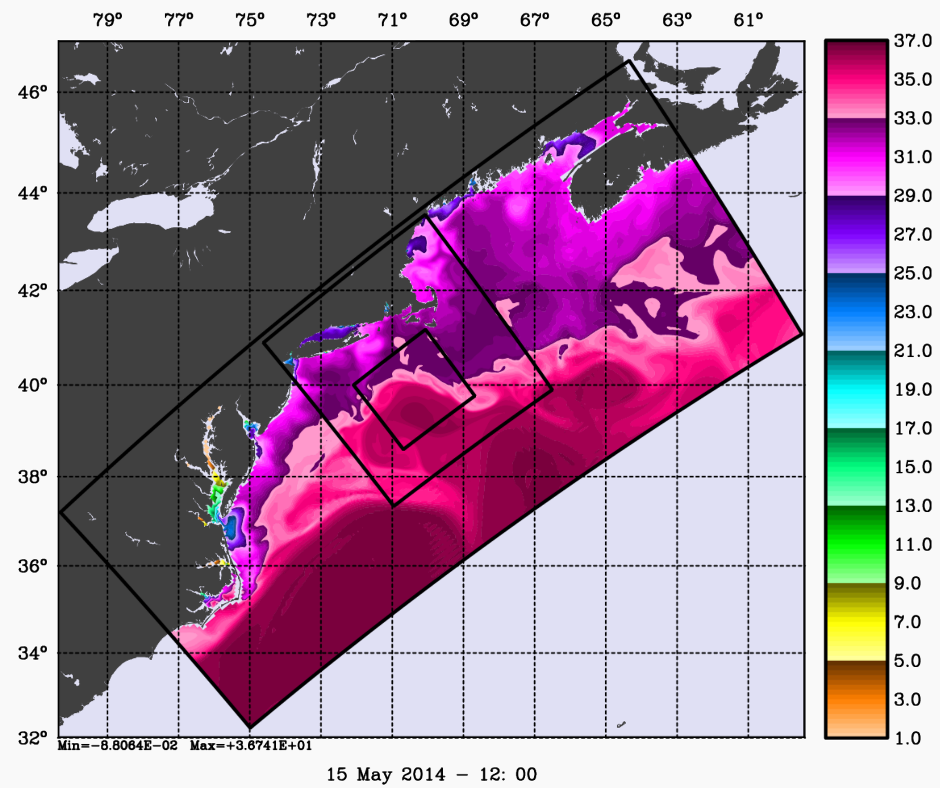


Surface Temperature (Celsius)

Two-Way DOPPIO-PIONEER-ARRAY

(1:3 Refinement)

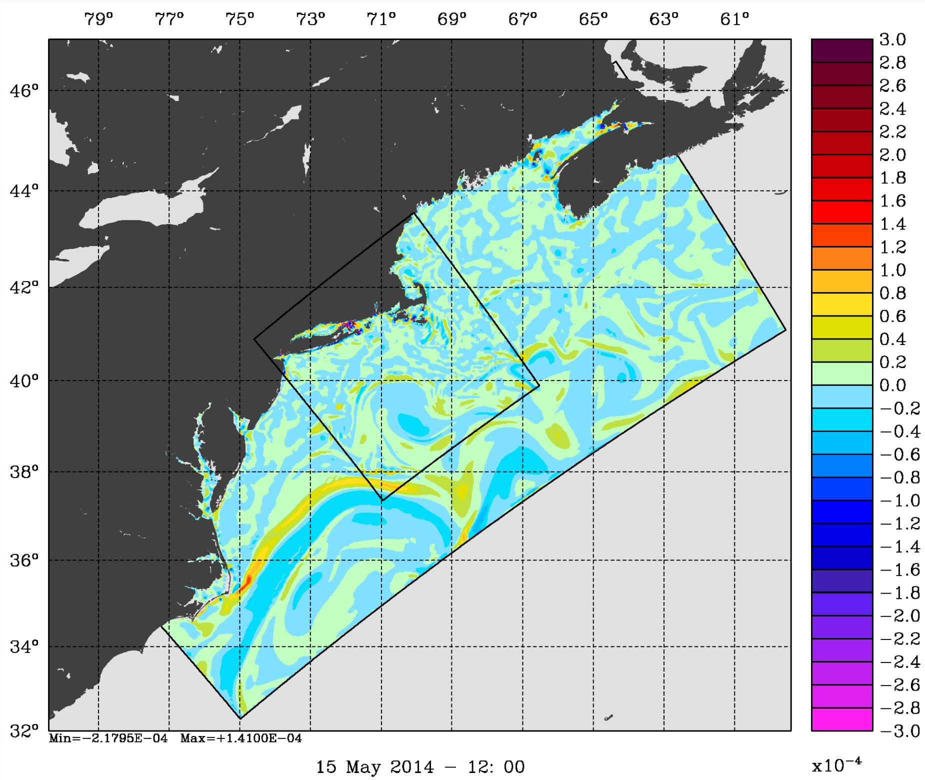


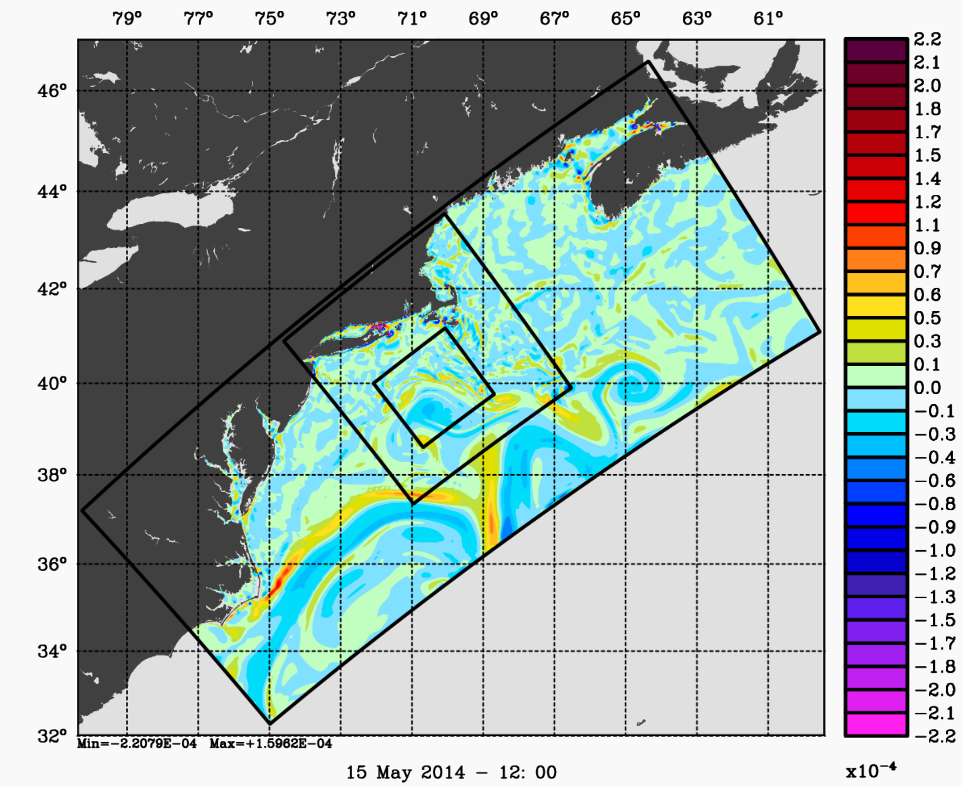


Surface Salinity

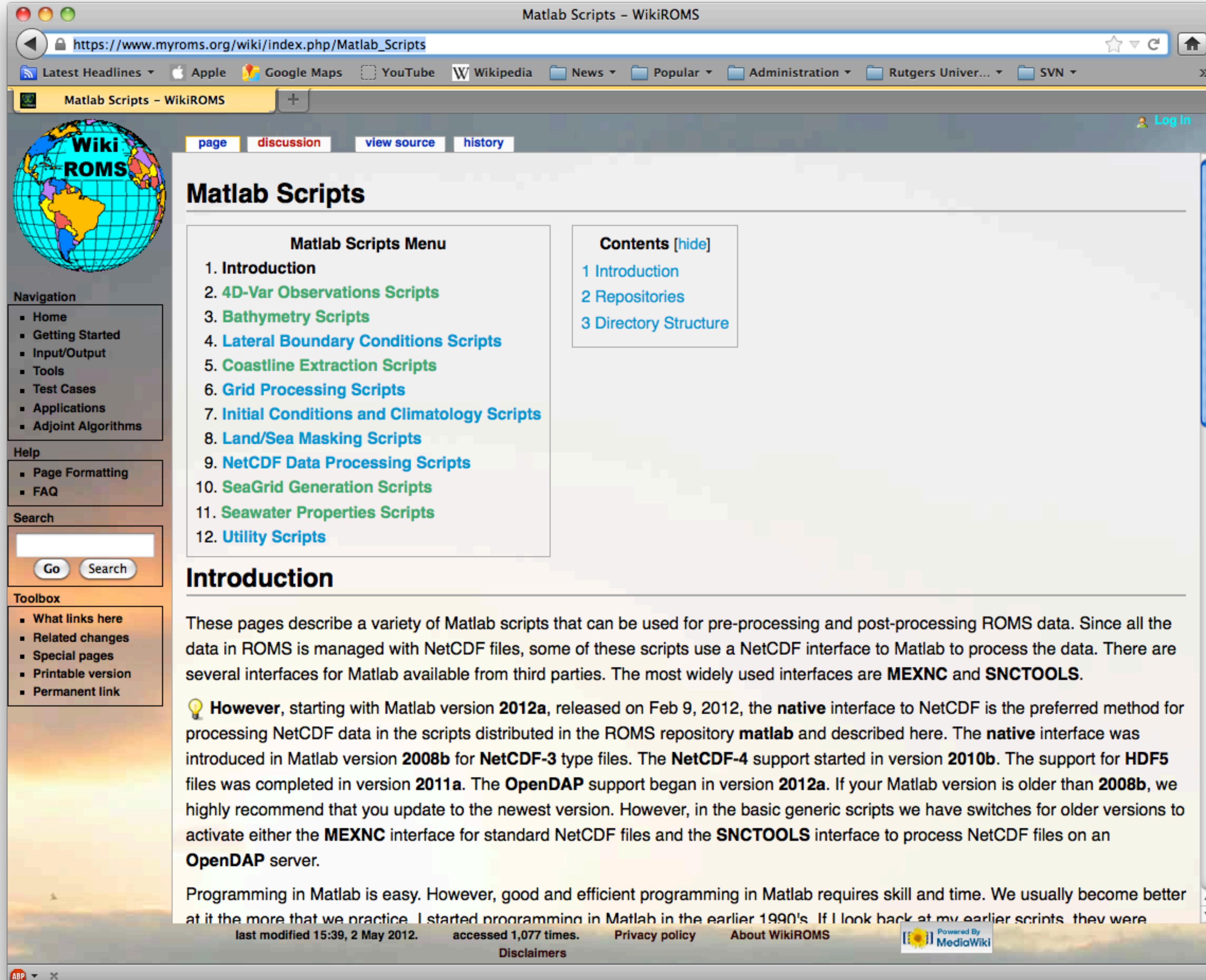
Two-Way DOPPIO-PIONEER-ARRAY

(1:3 Refinement)





Surface Relative Vorticity (1/s)


Matlab Scripts – WikiROMS

https://www.myroms.org/wiki/index.php/Matlab_Scripts

Latest Headlines | Apple | Google Maps | YouTube | Wikipedia | News | Popular | Administration | Rutgers Univer... | SVN

Matlab Scripts – WikiROMS

Log in

page | discussion | view source | history

Matlab Scripts

Matlab Scripts Menu

1. **Introduction**
2. 4D-Var Observations Scripts
3. Bathymetry Scripts
4. Lateral Boundary Conditions Scripts
5. Coastline Extraction Scripts
6. Grid Processing Scripts
7. Initial Conditions and Climatology Scripts
8. Land/Sea Masking Scripts
9. NetCDF Data Processing Scripts
10. SeaGrid Generation Scripts
11. Seawater Properties Scripts
12. Utility Scripts

Contents [hide]

Navigation
- Home
- Getting Started
- Input/Output
- Tools
- Test Cases
- Applications
- Adjoint Algorithms

Help
- Page Formatting
- FAQ

Search

Go | Search

Toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Introduction

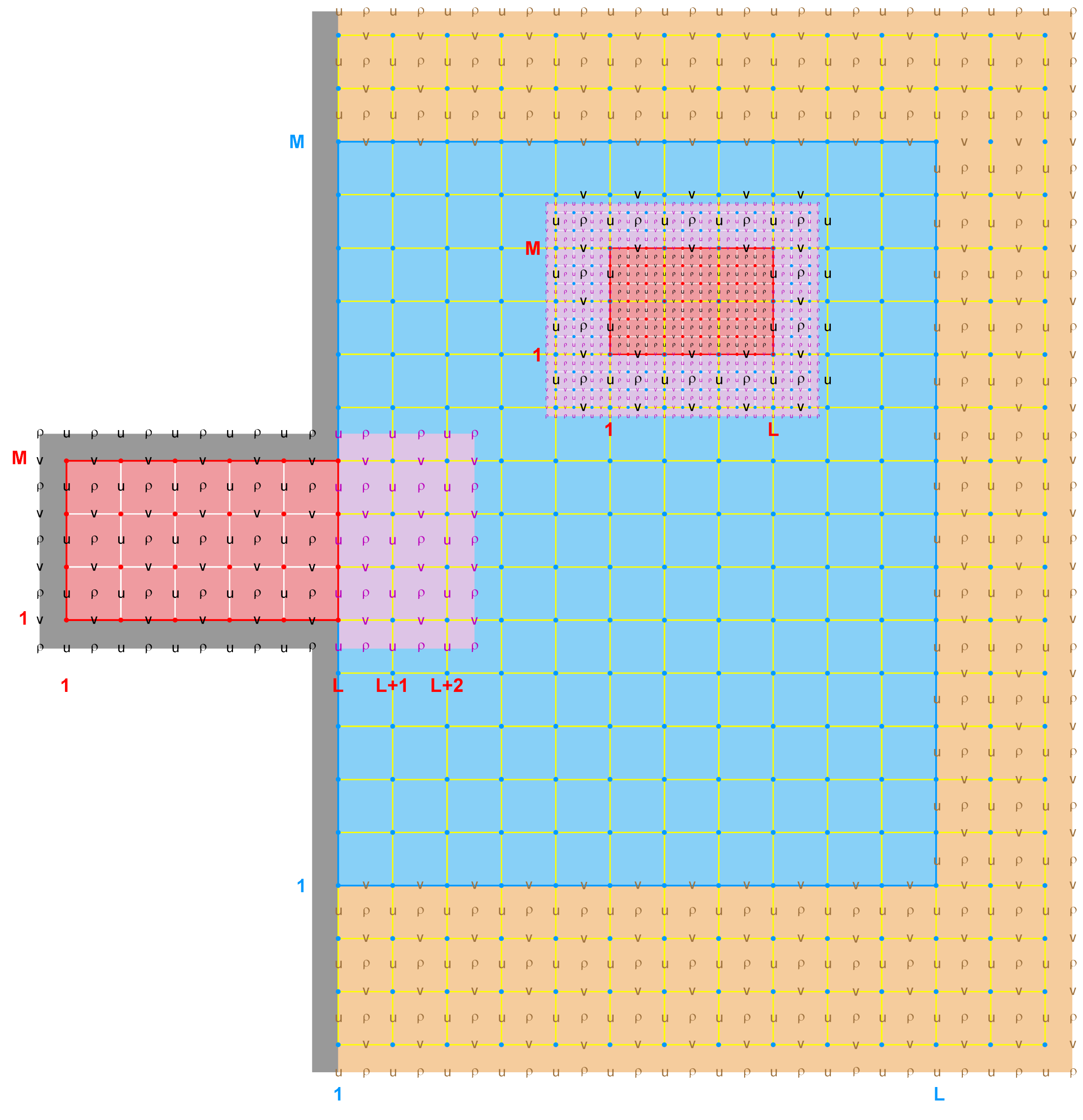
These pages describe a variety of Matlab scripts that can be used for pre-processing and post-processing ROMS data. Since all the data in ROMS is managed with NetCDF files, some of these scripts use a NetCDF interface to Matlab to process the data. There are several interfaces for Matlab available from third parties. The most widely used interfaces are **MEXNC** and **SNCTOOLS**.

However, starting with Matlab version **2012a**, released on Feb 9, 2012, the **native** interface to NetCDF is the preferred method for processing NetCDF data in the scripts distributed in the ROMS repository **matlab** and described here. The **native** interface was introduced in Matlab version **2008b** for **NetCDF-3** type files. The **NetCDF-4** support started in version **2010b**. The support for **HDF5** files was completed in version **2011a**. The **OpenDAP** support began in version **2012a**. If your Matlab version is older than **2008b**, we highly recommend that you update to the newest version. However, in the basic generic scripts we have switches for older versions to activate either the **MEXNC** interface for standard NetCDF files and the **SNCTOOLS** interface to process NetCDF files on an **OpenDAP** server.

Programming in Matlab is easy. However, good and efficient programming in Matlab requires skill and time. We usually become better at it the more that we practice. I started programming in Matlab in the earlier 1990's. If I look back at my earlier scripts, they were

https://www.myroms.org/wiki/index.php/Matlab_Scripts

Heterogeneous Model Nesting
(One-Way Open boundary Conditions)



Nesting Remarks

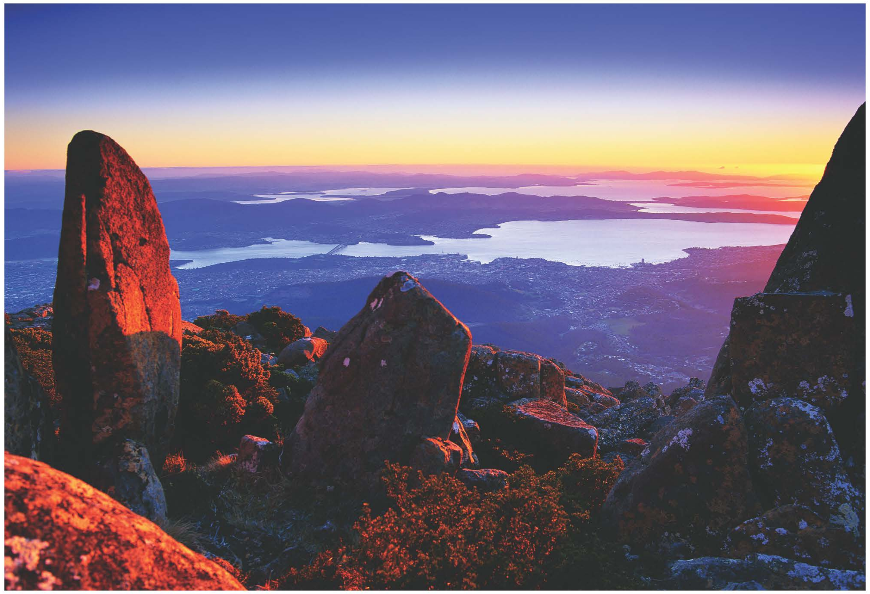
- ROMS nesting capabilities are unique and allow complex estuary and coastal configurations with unlimited number of composite and refined grids
- Coincident composite and mosaic grids produce identical solutions when compared to one large continuous grid
- Nowadays, both one-way and two-way nesting work well
- Placement of nested grids is application dependent and subject to geometrical and dynamical constrains
- The two-way exchange of information fine-to-coarse in grid refinement applications is expensive. We are exploring strategies to minimize computational cost.

Upcoming

- **Efficient two-way MPI-communications for the fine-to-coarse nesting step**
- **Heterogeneous models nesting: one-way open boundary conditions (NetCDF files or ESMF/MCT coupling)**
- **Multiple model coupling with ESMF (Earth System Modeling Framework, Version 7) including the NUOPC (National Unified Operational Prediction Capability) layer**
- **Fully coupled ROMS and COAMPS dynamics via ESMF and NUOPC**
- **Fully coupled ROMS and COAMPS data assimilation (EnKF and 4D-Var)**
- **4D-Var data assimilation within nested grids**
- **Overhaul of ROMS plotting package (NCAR GKS library) to include nesting grids**
- **Release of ROMS 4.0**

2016 ROMS Asia-Pacific Workshop

Hobart, Tasmania
Australia
October 17 - 21, 2016



Organized by: Hernan G. Arango, John L. Wilkin, Andrew M. Moore, David Gwyther, Ben Galton-Fenzi, and Andreas Klocker

http://www.myroms.org/tasmania_workshop

Reference

Warner, J.C., W.R. Geyer, and H.G. Arango, 2010: Using composite grid approach in complex coastal domain to estimate estuarine residence time, *Computer and Geosciences*, **36**, 921-935, doi:10.1016/j.cageo.2009.11.008.