Tutorial 9: Computing Normalization Coefficients for Covariance Models

Prior Error Covariance Modeling

Recall: $\mathbf{B_x} = \mathbf{K_b \Sigma C \Sigma^T K_b^T}$

C is a correlation matrix for the unbalanced increments, and is modeled as the solution of a diffusion equation:

$$\partial \mathbf{x} / \partial t - \kappa \nabla^2 \mathbf{x} = 0 \quad \rightarrow \quad \mathbf{C'x}$$

But, C´ is arbitrary at this stage, and must to be normalized to ensure that the range is ±1 as required for a correlation function. Therefore we define:

$$\mathbf{C} = \mathbf{\Lambda C' \Lambda}^{\mathrm{T}}$$

where Λ is a diagonal matrix with elements $\left(c'_{ii} \right)^{-1/2}$

Finally: $\mathbf{B_x} = \mathbf{K_b \Sigma \Lambda C' \Lambda^T \Sigma^T K_b^T}$

Computing Λ

(define NORMALIZATION)

Following Weaver & Courtier (2001) we employ two methods for computing the elements of Λ:

(i) Exact method (Nmethod=0):

$\mathbf{C}'\mathbf{e}_i \rightarrow$ i^{th} **column of C´; save** c_{ii}

where $\mathbf{e}_i^{\mathrm{T}} = (0, 0, \ldots, 0, 1, 0, \cdots, 0)$

(1 = i^{th} element)

Requires N_{grid} **runs of diffusion operators: impractical for v. large grids.**

Computing Λ

(ii) Randomization method (Nmethod=1):

Estimate the diagonal elements c'_{ii} of C´ from:

$$\mathbf{C}' \approx \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{\xi}\mathbf{C}'\boldsymbol{\xi}^{\mathrm{T}} = \tilde{\mathbf{C}}$$

where ξ is a random vector: $\xi \to N(\mathbf{0}, \mathbf{I})$

M is the sample size. As $M \to \infty, \quad \tilde{\mathbf{C}} \to \mathbf{C}'$

(Nrandom)

Uncertainty in elements of Λ^{-1}: $(2M)^{-1/2}$

Practical requirement: $M << N_{grid}$

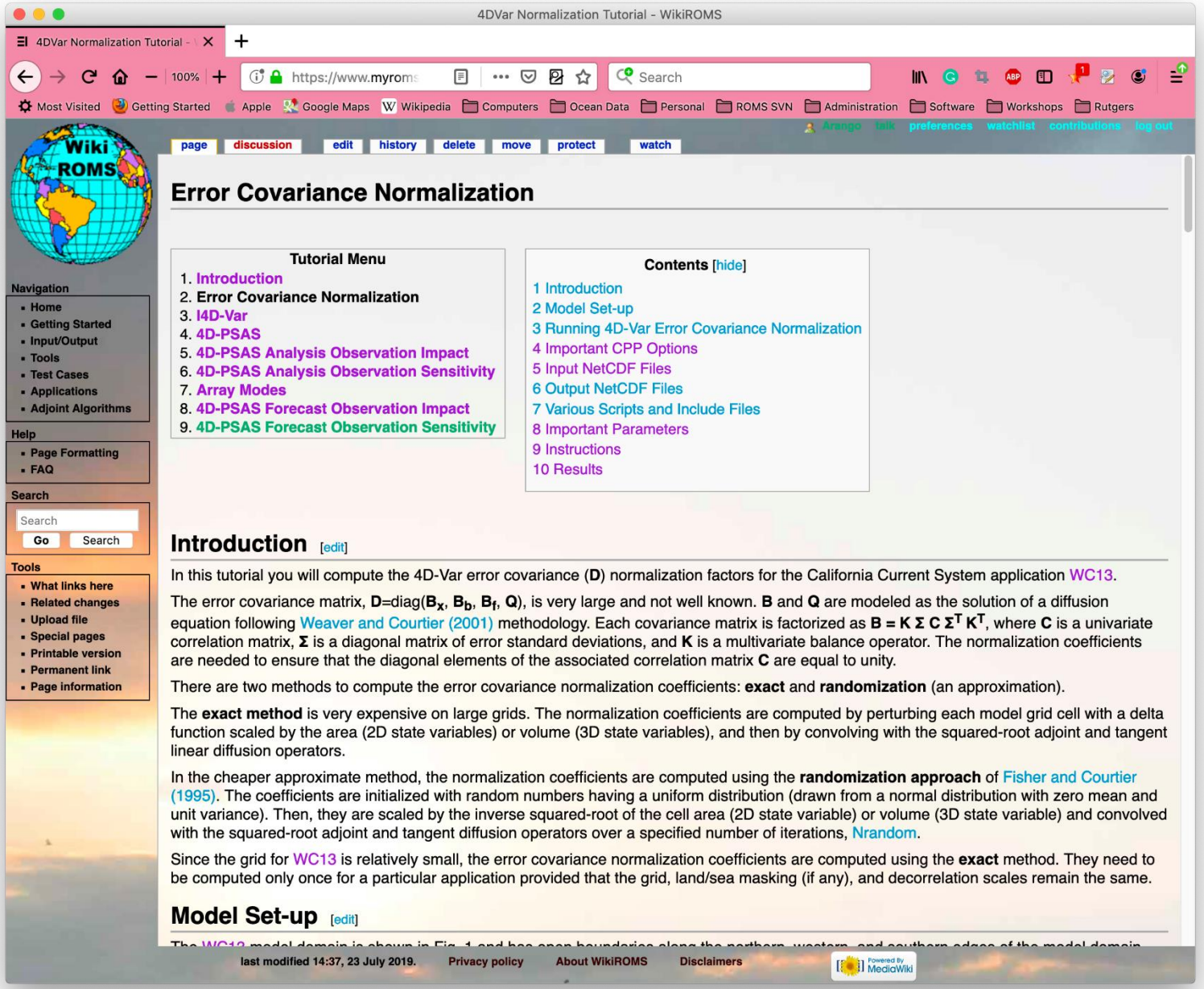
(Fisher and Courtier, 1995)

Computing Λ

Practicalities:

- **choose *M* and run normalization driver**
- **choose another seed and run again**
- **compare the estimates – are they similar**
- **compute means of the two *M* samples**
- **repeat as necessary until mean does not change significantly**

Normalization Tutorial Wiki Page



WC13 C-preprocessing Options
(Basic Configuration)

Momentum Equations Options:

```
#define UV_ADV              including advection terms
#define UV_COR              including Coriolis term
#define UV_U3HADVECTION     3rd-order Upstream Horizontal advection
#define UV_C4VADVECTION     4th-order Centered Vertical advection
#define DJ_GRADPS           splines density Jacobian PGF
#define UV_QDRAG            quadratic bottom friction
#define UV_VIS2             harmonic horizontal mixing
#define MIX_S_UV            mixing along s-levels
#define SPLINES_VVISC       parabolic Splines for Vertical Viscosity
```

Tracers Equations Options:

```
#define TS_U3HADVECTION     3rd-order Upstream horizontal advection
#define TS_C4VADVECTION     4th-order Centered  advection
#define TS_DIF2             harmonic horizontal mixing
#define MIX_GEO_TS          mixing along geo-potentials
#define SALINITY            including salinity
#define NONLIN_EOS          nonlinear equation of state
#define SPLINES_VDIFF       parabolic splines for vertical Diffusion

#define ANA_BTFLUX          analytical bottom Temp flux
#define ANA_BSFLUX          analytical bottom Salt flux
```

Surface Forcing Options:

```
#define BULK_FLUXES         surface bulk fluxes parameterization

#define DIURNAL_SRFLUX      modulate shortwave by the local diurnal cycle
#define EMINUSP             compute Salt Flux using E-P
#define LONGWAVE_OUT        compute outgoing longwave radiation
#define SOLAR_SOURCE        solar radiation source term
```

Vertical Turbulent Mixing Parameterization Options:

```
#define GLS_MIXING          Generic Length Scale Mixing (K-omega)
#ifdef GLS_MIXING
# define N2S2_HORAVG        smoothing of buoyancy/shear
# define KANTHA_CLAYSON     stability function
# RI_SPLINES                parabolic splines for Ri Number
#endif
```

Model Configuration Options:

```
#define SOLVE3D             solve 3D primitive equations
#define CURVGRID            curvilinear grid
#define MASKING             land/sea masking
#define SPHERICAL           spherical grid
#define PROFILE             time profiling

#define ANA_SPONGE          analytical viscosity/diffusion sponge
```

WC13 C-preprocessing Options
(Error Covariance Configuration)

Algorithm:

```
#define NORMALIZATION        primal form of incremental strong constraint 4D-Var
#define ANA_INTIAL           analytical initial conditions (zero fields)
```

Control Vector:

```
#define ADJUST_BOUNDARY      open boundary conditions increments
#define ADJUST_STFLUX        surface tracer flux increments
#define ADJUST_WSTRESS       surface wind stress increments
```

Error Covariance Modeling:

```
#define CORRELATION          model error covariance correlation with diffusion operators
#define FULL_GRID            consider both interior and boundary points
#define VCONVOLUTION         Vertical correlation modeling
#define IMPLICIT_VCON        Implicit vertical diffusion operator
#undef  BALANCE_OPERATOR     Multivariate balance constraint
#ifdef  BALANCE_OPERATOR
# define ZETA_ELLIPTIC       SSH elliptic equation method
#endif
```

Prior:

```
#define FORWARD_READ         read basic state linearization in TLM and ADM files
#define FORWARD_WRITE        writing basic state by the NLM
#define FORWARD_MIXING       processing basic state vertical mixing coefficients
#define NL_BULK_FLUXES       surface kinematic fluxes from nonlinear model
```

I/O :

```
#define OUT_DOUBLE           double precision data in output NLM, TLM, and ADM
```

Include File: wc13.h

```
/*
** svn $Id: wc13.h 1024 2020-05-14 03:36:12Z arango $
********************************************************************
** Copyright (c) 2002-2020 The ROMS/TOMS Group                     **
**   Licensed under a MIT/X style license                          **
**   See License_ROMS.txt                                          **
********************************************************************
**
** Options for the California Current System, 1/3 degree resolution.
**
** Application flag:   WC13
** Input script:       roms_wc13.in
**                     s4dvar.in
**
** Available Drivers options: choose only one and activate it in the
**                            build.sh script (MY_CPP_FLAGS definition)
**
**  AD_SENSITIVITY            Adjoint Sensitivity Driver
**  AFT_EIGENMODES            Adjoint Finite Time Eigenmodes
**  ARRAY_MODES               Stabilized representer matrix array modes
**  CLIPPING                  Stabilized representer matrix clipped analysis
**  CORRELATION               Background-error Correlation Check
**  GRADIENT_CHECK            TLM/ADM Gradient Check
**  FORCING_SV                Forcing Singular Vectors
**  FT_EIGENMODES             Finite Time Eigenmodes
**  I4DVAR                    Incremental, strong constraint I4D-Var
**  NLM_DRIVER                Nonlinear Basic State trajectory
**  OPT_PERTURBATION          Optimal perturbations
**  PICARD_TEST               Picard Iterations Test
**  RBL4DVAR                  Strong/Weak constraint RBL4D-Var
```

4D-Var Error Covariance Normalization Files

- **Four different error covariance normalization coefficients NetCDF files are required in ROMS 4D-Var algorithms to ensure that the diagonal elements of the associated correlation matrix (C) are equal to unity:**
  - **Model error normalization file, if weak constraint**
  - **Initial conditions normalization file**
  - **Open boundary conditions normalization file, if ADJUST_BOUNDARY**
  - **Surface forcing normalization file, if ADJUST_WSTRESS and/or ADJUST_STFLUX**
- **These normalization NetCDF files are specified in 4D-Var input script as:**

```
NRMnameM == ../Data/wc13_nrm_m.nc
NRMnameI == ../Data/wc13_nrm_i.nc
NRMnameB == ../Data/wc13_nrm_b.nc
NRMnameF == ../Data/wc13_nrm_f.nc
```

Model Error and Initial Conditions Metadata

```
Variables:
. . .

        double zeta(ocean_time, eta_rho, xi_rho) ;
                zeta:long_name = "free-surface, initial conditions error covariance normalization" ;
                zeta:units = "meter" ;
                zeta:time = "ocean_time" ;
                zeta:coordinates = "lon_rho lat_rho ocean_time" ;
        double ubar(ocean_time, eta_u, xi_u) ;
                ubar:long_name = "vertically integrated u-momentum component, initial conditions error covariance normalization" ;
                ubar:units = "meter" ;
                ubar:time = "ocean_time" ;
                ubar:coordinates = "lon_u lat_u ocean_time" ;
        double vbar(ocean_time, eta_v, xi_v) ;
                vbar:long_name = "vertically integrated v-momentum component, initial conditions error covariance normalization" ;
                vbar:units = "meter" ;
                vbar:time = "ocean_time" ;
                vbar:coordinates = "lon_v lat_v ocean_time" ;
        double u(ocean_time, s_rho, eta_u, xi_u) ;
                u:long_name = "u-momentum component, initial conditions error covariance normalization" ;
                u:units = "meter" ;
                u:time = "ocean_time" ;
                u:coordinates = "lon_u lat_u s_rho ocean_time" ;
        double v(ocean_time, s_rho, eta_v, xi_v) ;
                v:long_name = "v-momentum component, initial conditions error covariance normalization" ;
                v:units = "meter" ;
                v:time = "ocean_time" ;
                v:coordinates = "lon_v lat_v s_rho ocean_time" ;
        double temp(ocean_time, s_rho, eta_rho, xi_rho) ;
                temp:long_name = "potential temperature, initial conditions error covariance normalization" ;
                temp:units = “meter" ;
                temp:time = "ocean_time" ;
                temp:coordinates = "lon_rho lat_rho s_rho ocean_time" ;
        double salt(ocean_time, s_rho, eta_rho, xi_rho) ;
                salt:long_name = "salinity, initial conditions error covariance normalization" ;
                salt:units = “meter" ;
                salt:time = "ocean_time" ;
                salt:coordinates = "lon_rho lat_rho s_rho ocean_time" ;
```

Open Boundary Conditions Metadata

```
dimensions:

        xi_rho = 56 ;
        eta_rho = 55 ;
. . .
        IorJ = 56 ;
        boundary = 4 ;

variables:
. . .
        double zeta_obc(ocean_time, boundary, IorJ) ;
                zeta_obc:long_name = "free-surface, open boundaries conditions error covariance normalization" ;
                zeta_obc:units = "meter" ;
                zeta_obc:time = "ocean_time" ;
        double ubar_obc(ocean_time, boundary, IorJ) ;
                ubar_obc:long_name = "vertically integrated u-momentum component, open boundaries conditions error covariance normalization" ;
                ubar_obc:units = "meter second-1" ;
                ubar_obc:time = "ocean_time" ;
        double vbar_obc(ocean_time, boundary, IorJ) ;
                vbar_obc:long_name = "vertically integrated v-momentum component, open boundaries conditions error covariance normalization" ;
                vbar_obc:units = "meter second-1" ;
                vbar_obc:time = "ocean_time" ;
        double u_obc(ocean_time, s_rho, boundary, IorJ) ;
                u_obc:long_name = "u-momentum component, open boundaries conditions error covariance normalization" ;
                u_obc:units = "meter second-1" ;
                u_obc:time = "ocean_time" ;
        double v_obc(ocean_time, s_rho, boundary, IorJ) ;
                v_obc:long_name = "v-momentum component, open boundaries conditions error covariance normalization" ;
                v_obc:units = "meter second-1" ;
                v_obc:time = "ocean_time" ;
        double temp_obc(ocean_time, s_rho, boundary, IorJ) ;
                temp_obc:long_name = "potential temperature, open boundaries conditions error covariance normalization" ;
                temp_obc:units = "Celsius" ;
                temp_obc:time = "ocean_time" ;
        double salt_obc(ocean_time, s_rho, boundary, IorJ) ;
                salt_obc:long_name = "salinity, open boundaries conditions error covariance normalization" ;
                salt_obc:time = "ocean_time" ;

// global attributes:
. . .
                :boundary_index = "West=1, South=2, East=3, North=4"
```

Surface Forcing Metadata

```
dimensions:

        xi_rho = 56 ;
        xi_u = 55 ;
        xi_v = 56 ;
        eta_rho = 55 ;
        eta_u = 55 ;
        eta_v = 54 ;
        s_rho = 30 ;
        ocean_time = UNLIMITED ; // (1 currently)

variables:
. . .
        double sustr(ocean_time, eta_u, xi_u) ;
                sustr:long_name = "surface u-momentum stress, error covariance normalization" ;
                sustr:units = "newton meter-2" ;
                sustr:time = "ocean_time" ;
                sustr:coordinates = "lon_u lat_u ocean_time" ;
        double svstr(ocean_time, eta_v, xi_v) ;
                svstr:long_name = "surface v-momentum stress, error covariance normalization" ;
                svstr:units = "newton meter-2" ;
                svstr:time = "ocean_time" ;
                svstr:coordinates = "lon_v lat_v ocean_time" ;
        double shflux(ocean_time, eta_rho, xi_rho) ;
                shflux:long_name = "surface net heat flux, error covariance normalization" ;
                shflux:units = "watt meter-2" ;
                shflux:negative = "upward flux, cooling" ;
                shflux:positive = "downward flux, heating" ;
                shflux:time = "ocean_time" ;
                shflux:coordinates = "lon_rho lat_rho ocean_time" ;
        double ssflux(ocean_time, eta_rho, xi_rho) ;
                ssflux:long_name = "surface net salt flux (E-P)*SALT, error covariance normalization" ;
                ssflux:units = "meter second-1" ;
                ssflux:time = "ocean_time" ;
                ssflux:coordinates = "lon_rho lat_rho ocean_time" ;


// global attributes:

                :type = "ROMS 4D-Var surface forcing error covariance normalization" ;
                :title = "California Current System, 1/3 degree resolution (WC13)" ;
                :Conventions = "CF-1.4" ;
                :grd_file = "test/WC13/Data/wc13_grd.nc" ;
```

Standard Input File: roms_wc13.in

```
!
!  ROMS/TOMS Standard Input parameters.
!
!svn $Id: roms_wc13.in 971 2019-07-09 03:02:57Z arango $
!========================================================== Hernan G. Arango ===
!  Copyright (c) 2002-2019 The ROMS/TOMS Group                                 !
!    Licensed under a MIT/X style license                                      !
!    See License_ROMS.txt                                                      !
!===============================================================================
!                                                                              !
! Input parameters can be entered in ANY order, provided that the parameter    !
! KEYWORD (usually, upper case) is typed correctly followed by "="  or "=="    !
! symbols. Any comment lines are allowed and must begin with an exclamation    !
! mark (!) in column one.  Comments may appear to the right of a parameter     !
! specification to improve documentation.  Comments are ignored during         !
! reading.  Blank lines are also allowed and ignored. Continuation lines in    !
! a parameter specification are allowed if preceded by a backslash (\).  In    !
! some instances, more than one value is required for a parameter.  If fewer   !
! values are provided, the last value is assigned for the entire parameter     !
! array.  The multiplication symbol (*),  without blank spaces in between,     !
! is allowed for a parameter specification. For example, in two grids nested   !
! application:                                                                 !
!                                                                              !
!    AKT_BAK == 2*1.0d-6  2*5.0d-6              ! m2/s                         !
!                                                                              !
! indicates that the first two entries of array AKT_BAK,  in fortran column-   !
! major order, will have the same value of "1.0d-6" for grid 1,  whereas the   !
! next two entries will have the same value of "5.0d-6" for grid 2.            !
!                                                                              !
! In multiple levels of nesting or multiple connected domains step-ups         !
```

C4dvar.in Important Parameters

```
Nmethod  == 0                ! normalization method:  0=Exact (expensive) or 1=Approximated (randomization)
Nrandom  == 5000             ! randomization iterations
. . .
LdefNRM  == F F F F          ! Create a new normalization files (model, IC, OBC, surface forcing)
LwrtNRM  == F F F F          ! Compute and write normalization  (model, IC, OBC, surface forcing)
. . .
CnormM(isFsur) =  T          ! model error covariance, 2D variable at RHO-points
CnormM(isUbar) =  T          ! model error covariance, 2D variable at U-points
CnormM(isVbar) =  T          ! model error covariance, 2D variable at V-points
CnormM(isUvel) =  T          ! model error covariance, 3D variable at U-points
CnormM(isVvel) =  T          ! model error covariance, 3D variable at V-points
CnormM(isTvar) =  T T        ! model error covariance, NT tracers
. . .
CnormI(isFsur) =  T          ! IC error covariance, 2D variable at RHO-points
CnormI(isUbar) =  T          ! IC error covariance, 2D variable at U-points
CnormI(isVbar) =  T          ! IC error covariance, 2D variable at V-points
CnormI(isUvel) =  T          ! IC error covariance, 3D variable at U-points
CnormI(isVvel) =  T          ! IC error covariance, 3D variable at V-points
CnormI(isTvar) =  T T        ! IC error covariance, NT tracers
. . .
CnormB(isFsur) =  T          ! OBC error covariance, 2D variable at RHO-points
CnormB(isUbar) =  T          ! OBC error covariance, 2D variable at U-points
CnormB(isVbar) =  T          ! OBC error covariance, 2D variable at V-points
CnormB(isUvel) =  T          ! OBC error covariance, 3D variable at U-points
CnormB(isVvel) =  T          ! OBC error covariance, 3D variable at V-points
CnormB(isTvar) =  T T        ! OBC error covariance, NT tracers
. . .
CnormF(isUstr) =  T          ! surface forcing error covariance, U-momentum stress
CnormF(isVstr) =  T          ! surface forcing error covariance, V-momentum stress
CnormF(isTsur) =  T T        ! Surface forcing error covariance, NT tracers fluxes
. . .
NRMnameM == wc13_nrm_m.nc    ! model error (weak constraint)
NRMnameI == wc13_nrm_i.nc    ! initial conditions
NRMnameB == wc13_nrm_b.nc    ! open boundary conditions
NRMnameF == wc13_nrm_f.nc    ! surface forcing (wind stress and net heat flux)
```

C4dvar.in 4D-Var Parameters: Decorrelation Scales

```
Horizontal and vertical stability and accuracy factors (< 1):
!                    IC          Model       OBC         Sur For
          Hgamma =   0.5         0.5         0.5         0.5         ! horizontal operator
          Vgamma =   0.0005      0.0005      0.0005      0.0005      ! vertical operator

Model error correlations (m):

HdecayM(isFsur) ==   50.0d+3                                         ! free-surface    (16 convolutions)
HdecayM(isUbar) ==   50.0d+3                                         ! 2D U-momentum   (16 convolutions)
HdecayM(isVbar) ==   50.0d+3                                         ! 2D V-momentum   (16 convolutions)
HdecayM(isUvel) ==   50.0d+3                                         ! 3D U-momentum   (16 convolutions)
HdecayM(isVvel) ==   50.0d+3                                         ! 3D V-momentum   (16 convolutions)
HdecayM(isTvar) ==   50.0d+3    50.0d+3                              ! 1:NT tracers    (16 convolutions)

VdecayM(isUvel) ==   30.0d0                                          ! 3D U-momentum   (8  convolutions)
VdecayM(isVvel) ==   30.0d0                                          ! 3D V-momentum   (8  convolutions)
VdecayM(isTvar) ==   30.0d0     30.0d0                               ! 1:NT tracers    (8  convolutions)

Initial conditions error correlations (m):

HdecayI(isFsur) ==   50.0d+3                                         ! free-surface    (16 convolutions)
HdecayI(isUbar) ==   50.0d+3                                         ! 2D U-momentum   (16 convolutions)
HdecayI(isVbar) ==   50.0d+3                                         ! 2D V-momentum   (16 convolutions)
HdecayI(isUvel) ==   50.0d+3                                         ! 3D U-momentum   (16 convolutions)
HdecayI(isVvel) ==   50.0d+3                                         ! 3D V-momentum   (16 convolutions)
HdecayI(isTvar) ==   50.0d+3    50.0d+3                              ! 1:NT tracers    (16 convolutions)

VdecayI(isUvel) ==   30.0d0                                          ! 3D U-momentum   (8  convolutions)
VdecayI(isVvel) ==   30.0d0                                          ! 3D V-momentum   (8  convolutions)
VdecayI(isTvar) ==   30.0d0     30.0d0                               ! 1:NT tracers    (8  convolutions)

Surface forcing error correlations (m):

HdecayF(isUstr) == 100.0d+3                  ! surface U-momentum stress   (66 convolutions)
HdecayF(isVstr) == 100.0d+3                  ! surface V-momentum stress   (66 convolutions)
HdecayF(isTsur) == 100.0d+3  100.0d+3        ! 1:NT surface tracer flux    (66 convolutions)
```

Normalization Method

- The exact method is very expensive on large grids.
- The normalization coefficients are computed by perturbing each model grid cell with a delta function scaled by the area (2D state variables) or volume (3D state variables), and then convolving with the squared-root adjoint and tangent linear diffusion operators. The diffusion operator is self-adjointed.
- The randomization method is cheaper (Fisher and Courtier, 1985).
- The normalization coefficients are initialized with random numbers having a uniform distribution (drawn from a normal distribution with zero mean and unit variance). Then, they are scaled by the inverse squared-root of the cell area (2D state variables) or volume (3D state variables) and convolved with the squared-root adjoint and tangent linear diffusion operator over a specified number of iterations, Nrandom.
- The normalization coefficients need to be computed only once for a particular application provided that the grid, land/sea masking (if any), and decorrelation scales remain the same.

4D-Var Parameters: Decorrelation Scales

Horizontal and vertical stability and accuracy factors (< 1):

```
!                    IC          Model       OBC         Sur For

          Hgamma =   0.5         0.5         0.5         0.5          ! horizontal operator
          Vgamma =   0.0005      0.0005      0.0005      0.0005       ! vertical operator
```

Model Error correlations (m):

```
HdecayM(isFsur) ==   50.0d+3                                          ! free-surface    (16 convolutions)
HdecayM(isUbar) ==   50.0d+3                                          ! 2D U-momentum   (16 convolutions)
HdecayM(isVbar) ==   50.0d+3                                          ! 2D V-momentum   (16 convolutions)
HdecayM(isUvel) ==   50.0d+3                                          ! 3D U-momentum   (16 convolutions)
HdecayM(isVvel) ==   50.0d+3                                          ! 3D V-momentum   (16 convolutions)
HdecayM(isTvar) ==   50.0d+3     50.0d+3                              ! 1:NT tracers    (16 convolutions)

VdecayM(isUvel) ==   30.0d0                                           ! 3D U-momentum   (8  convolutions)
VdecayM(isVvel) ==   30.0d0                                           ! 3D V-momentum   (8  convolutions)
VdecayM(isTvar) ==   30.0d0      30.0d0                               ! 1:NT tracers    (8  convolutions)
```

4D-Var Parameters: Decorrelation Scales

Initial conditions correlations (m):

```
HdecayI(isFsur) ==  50.0d+3                          ! free-surface    (16 convolutions)
HdecayI(isUbar) ==  50.0d+3                          ! 2D U-momentum   (16 convolutions)
HdecayI(isVbar) ==  50.0d+3                          ! 2D V-momentum   (16 convolutions)
HdecayI(isUvel) ==  50.0d+3                          ! 3D U-momentum   (16 convolutions)
HdecayI(isVvel) ==  50.0d+3                          ! 3D V-momentum   (16 convolutions)
HdecayI(isTvar) ==  50.0d+3   50.0d+3                ! 1:NT tracers    (16 convolutions)

VdecayI(isUvel) ==  30.0d0                           ! 3D U-momentum   (8  convolutions)
VdecayI(isVvel) ==  30.0d0                           ! 3D V-momentum   (8  convolutions)
VdecayI(isTvar) ==  30.0d0    30.0d0                 ! 1:NT tracers    (8  convolutions)
```

Surface forcing correlations (m):

```
HdecayF(isUstr) == 100.0d+3                  ! surface U-momentum stress (66 convolutions)
HdecayF(isVstr) == 100.0d+3                  ! surface V-momentum stress (66 convolutions)
HdecayF(isTsur) == 100.0d+3  100.0d+3        ! 1:NT surface tracer flux  (66 convolutions)
```

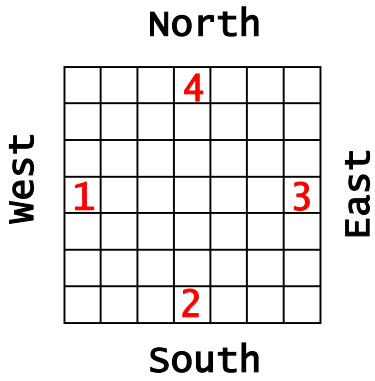
4D-Var Parameters: Decorrelation Scales

Open boundary conditions correlations (m):

```
!                    1: west   2: south   3: east   4: north

HdecayB(isFsur) == 100.0d+3 100.0d+3 100.0d+3 100.0d+3     ! free-surface
HdecayB(isUbar) == 100.0d+3 100.0d+3 100.0d+3 100.0d+3     ! 2D U-momentum
HdecayB(isVbar) == 100.0d+3 100.0d+3 100.0d+3 100.0d+3     ! 2D V-momentum
HdecayB(isUvel) == 100.0d+3 100.0d+3 100.0d+3 100.0d+3     ! 3D U-momentum
HdecayB(isVvel) == 100.0d+3 100.0d+3 100.0d+3 100.0d+3     ! 3D V-momentum
HdecayB(isTvar) == 4*100.0d+3  4*100.0d+3                  ! 1:NT tracers

VdecayB(isUvel) ==  30.0d0   30.0d0   30.0d0   30.0d0      ! 3D U-momentum
VdecayB(isVvel) ==  30.0d0   30.0d0   30.0d0   30.0d0      ! 3D V-momentum
VdecayB(isTvar) == 4*30.d0  4*30.d0                        ! 1:NT tracers
```

Boundary edges to adjust (logical switches):

```
!                    1   2   3   4

   Lobc(isFsur) ==   T   T   F   T          ! free-surface
   Lobc(isUbar) ==   T   T   F   T          ! 2D U-momentum
   Lobc(isVbar) ==   T   T   F   T          ! 2D V-momentum
   Lobc(isUvel) ==   T   T   F   T          ! 3D U-momentum
   Lobc(isVvel) ==   T   T   F   T          ! 3D V-momentum

   Lobc(isTvar) ==   T   T   F   T  \
                     T   T   F   T
```



Normalization Parameters File: c4dvar.in

```
!  4DVar assimilation input parameters.
!
!svn $Id: s4dvar.in 971 2019-07-09 03:02:57Z arango $
!================================================== Hernan G. Arango ===
!  Copyright (c) 2002-2019 The ROMS/TOMS Group                         !
!    Licensed under a MIT/X style license                              !
!    See License_ROMS.txt                                              !
!========================================================================
!                                                                       !
! Input parameters can be entered in ANY order, provided that the parameter   !
! KEYWORD (usually, upper case) is typed correctly followed by "="  or "=="   !
! symbols. Any comment lines are allowed and must begin with an exclamation   !
! mark (!) in column one.  Comments may  appear to the right of a parameter   !
! specification to improve documentation.  Comments will be ignored  during   !
! reading.  Blank lines are also allowed and ignored. Continuation lines in   !
! a parameter specification are allowed and must be preceded by a backslash   !
! (\).  In some instances, more than one value is required for a parameter.   !
! If fewer values are provided, the  last value  is assigned for the entire   !
! parameter array.  The multiplication symbol (*),  without blank spaces in   !
! between, is allowed for a parameter specification.  For example, in a two   !
! grids nested application:                                             !
!                                                                       !
!    AKT_BAK == 2*1.0d-6  2*5.0d-6              ! m2/s                  !
!                                                                       !
! indicates that the first two entries of array AKT_BAK,  in fortran column-  !
! major order, will have the same value of "1.0d-6" for grid 1,  whereas the  !
! next two entries will have the same value of "5.0d-6" for grid 2.           !
!                                                                       !
! In multiple levels of nesting and/or multiple connected domains  step-ups,  !
```

Job Script: job_normalization.csh

1. Set path definition to one directory up in the tree.

```
set Dir = `dirname ${PWD}`
```

2. Set string manipulations perl script.

```
set SUBSTITUTE =
${ROMS_ROOT}/ROMS/Bin/substitute
```

3. Set model error, initial conditions, boundary conditions and surface forcing error covariance standard deviations files.

```
set STDnameM = ../Data/wc13_std_m.nc
set STDnameI = ../Data/wc13_std_i.nc
set STDnameB = ../Data/wc13_std_b.nc
set STDnameF = ../Data/wc13_std_f.nc
```

4. Set model error, initial conditions, boundary conditions and surface forcing error covariance normalization factors files.

```
set NRMnameM = ../Data/wc13_nrm_m.nc
set NRMnameI = ../Data/wc13_nrm_i.nc
set NRMnameB = ../Data/wc13_nrm_b.nc
set NRMnameF = ../Data/wc13_nrm_f.nc
```

5. Modify 4D-Var template input script and specify above files.

```
set NORM = c4dvar.in
if (-e $NORM) then
  /bin/rm $NORM
endif
cp s4dvar.in $NORM

$SUBSTITUTE $NORM roms_std_m.nc $STDnameM
$SUBSTITUTE $NORM roms_std_i.nc $STDnameI
$SUBSTITUTE $NORM roms_std_b.nc $STDnameB
$SUBSTITUTE $NORM roms_std_f.nc $STDnameF
$SUBSTITUTE $NORM roms_nrm_m.nc $NRMnameM
$SUBSTITUTE $NORM roms_nrm_i.nc $NRMnameI
$SUBSTITUTE $NORM roms_nrm_b.nc $NRMnameB
$SUBSTITUTE $NORM roms_nrm_f.nc $NRMnameF
$SUBSTITUTE $NORM roms_obs.nc   $OBSname
$SUBSTITUTE $NORM roms_hss.nc   wc13_hss.nc
$SUBSTITUTE $NORM roms_lcz.nc   wc13_lcz.nc
$SUBSTITUTE $NORM roms_mod.nc   wc13_mod.nc
$SUBSTITUTE $NORM roms_err.nc   wc13_err.nc
```

Job Script File: job_normalization.csh

```
#!/bin/csh -f
#
# svn $Id: job_normalization.csh 977 2019-07-26 06:01:07Z arango $
#########################################################################
# Copyright (c) 2002-2019 The ROMS/TOMS Group                           #
#   Licensed under a MIT/X style license                                #
#   See License_ROMS.txt                                                #
#########################################################################
#                                                                       #
# 4D-Var error covariance normalization coefficients job script:        #
#                                                                       #
# This script NEEDS to be run before any run:                           #
#                                                                       #
#   (1) It copies a new clean nonlinear model initial conditions        #
#       file. The nonlinear model is initialized from the               #
#       background or reference state.                                  #
#   (2) Specify model, initial conditions, boundary conditions, and     #
#       surface forcing error convariance input standard deviations     #
#       files.                                                          #
#   (3) Specify model, initial conditions, boundary conditions, and     #
#       surface forcing error convariance input/output normalization    #
#       factors filenames.                                              #
#   (4) Create 4D-Var input script "c4dvar.in" from a template and      #
#       specify the error covariance standard deviation, and error      #
#       covariance normalization factors files to be used.              #
#                                                                       #
#########################################################################

# Set path definition to one directory up in the tree.
```

Compile: build_roms.csh

1. Set a local environmental variable to define the path to the directories where all this project's files are kept.

```
setenv MY_ROOT_DIR          ${HOME}/ocean/toms/repository
setenv MY_PROJECT_DIR       ${PWD}
```

2. Location of your ROMS source code.

```
setenv MY_ROMS_SRC          ${MY_ROOT_DIR}/trunk
```

3. Path of Makefile configuration (*mk) files

```
setenv COMPILERS            ${MY_ROMS_SRC}/Compilers
setenv COMPILERS            ${HOME}/Compilers/ROMS
```

4. Build script invoked CPP options.

```
 setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DNORMALIZATION"
 setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DANA_initial"
 setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DANA_SPONGE"

#setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DADJUST_BOUNDARY"
#setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DADJUST_WSTRESS"
#setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DADJUST_STFLUX"

#setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DBALANCE_OPERATOR"
#setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DZETA_ELLIPTIC"

#setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DCOLLECT_ALLREDUCE"
#setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DREDUCE_ALLGATHER"

#setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DDEBUGGING"
#setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DPOSITIVE_ZERO"
```

5. Compiler selection environment variables.

```
setenv USE_MPI              on
setenv USE_MPIF90           on
setenv which_MPI            openmp

setenv FORT                 ifort
```

6. Use custom library paths

```
#setenv USE_MY_LIBS no           # use system default library paths
 setenv USE_MY_LIBS yes          # use my customized library paths

set MY_PATHS = ${COMPILERS}/my_build_paths.csh

if ($USE_MY_LIBS == 'yes') then
  source ${MY_PATHS} ${MY_PATHS}
endif
```

Build Script: build.csh

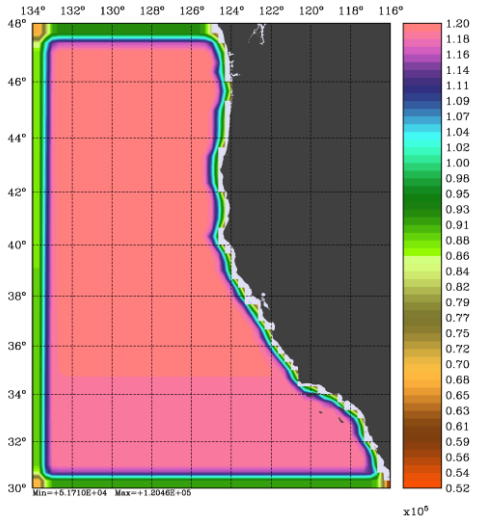
```
#!/bin/csh -f
#
# svn $Id: build_roms.csh 977 2019-07-26 06:01:07Z arango $
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# Copyright (c) 2002-2019 The ROMS/TOMS Group                           :::
#   Licensed under a MIT/X style license                                :::
#   See License_ROMS.txt                                                :::
#::::::::::::::::::::::::::::::::::::::::::::: Hernan G. Arango         :::
#                                                                       :::
# ROMS/TOMS Compiling CSH Script                                        :::
#                                                                       :::
# Script to compile an user application where the application-specific  :::
# files are kept separate from the ROMS source code.                    :::
#                                                                       :::
# Q: How/why does this script work?                                     :::
#                                                                       :::
# A: The ROMS makefile configures user-defined options with a set of    :::
#    flags such as ROMS_APPLICATION. Browse the makefile to see these.  :::
#    If an option in the makefile uses the syntax ?= in setting the     :::
#    default, this means that make will check whether an environment    :::
#    variable by that name is set in the shell that calls make. If so   :::
#    the environment variable value overrides the default (and the      :::
#    user need not maintain separate makefiles, or frequently edit      :::
#    the makefile, to run separate applications).                       :::
#                                                                       :::
# Usage:                                                                :::
#                                                                       :::
#    ./build_roms.csh [options]                                          :::
#                                                                       :::
# Options:                                                              :::
```

Build Script: my_build_paths.csh

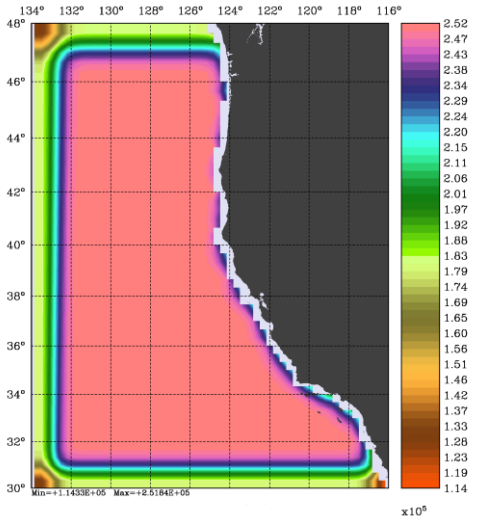
```
# svn $Id$
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# Copyright (c) 2002-2019 The ROMS/TOMS Group                           :::
#   Licensed under a MIT/X style license                                :::
#   See License_ROMS.txt                                                :::
#::::::::::::::::::::::::::::::::::::::::::::: Hernan G. Arango         :::
#                                                                       :::
# ROMS/TOMS Customized Compiling Libraries Script                       :::
#                                                                       :::
# This C-shell script sets the customized library paths needed by the   :::
# build script when the enviromental variable USE_MY_LIBS has a 'yes'   :::
# value.                                                                :::
#                                                                       :::
# For example, in build_roms.csh we have:                               :::
#                                                                       :::
#       if ($USE_MY_LIBS == 'yes') then                                 :::
#          source ${COMPILERS}/my_build_paths.csh                       :::
#       endif                                                           :::
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

set separator = `perl -e "print ':' x 100;"`

echo ""
echo "${separator}"
echo "Using customized library paths from:  $1"
echo "${separator}"
echo ""

#-----------------------------------------------------------------------
# Add MPI library to compile
```

Error Covariance Normalization Coefficients

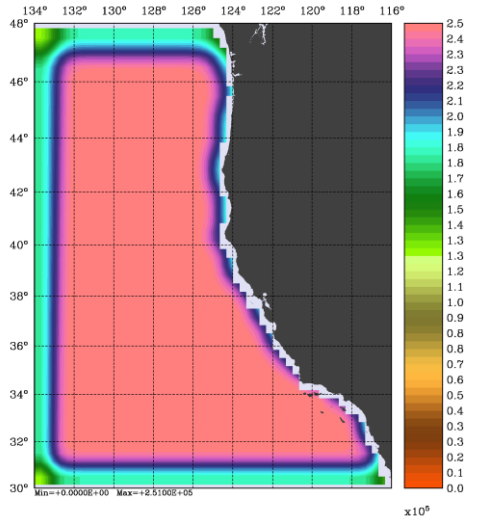
EXACT



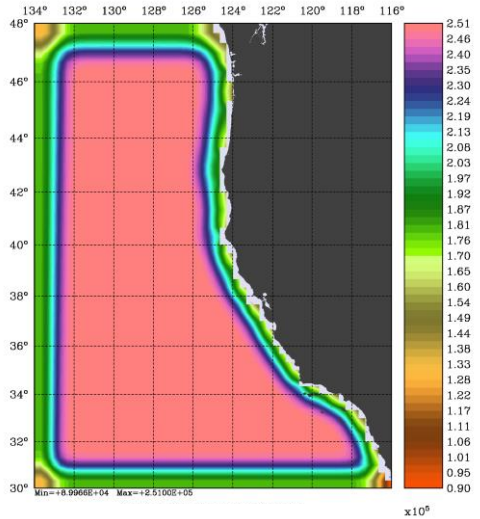
Free-surface



T_x

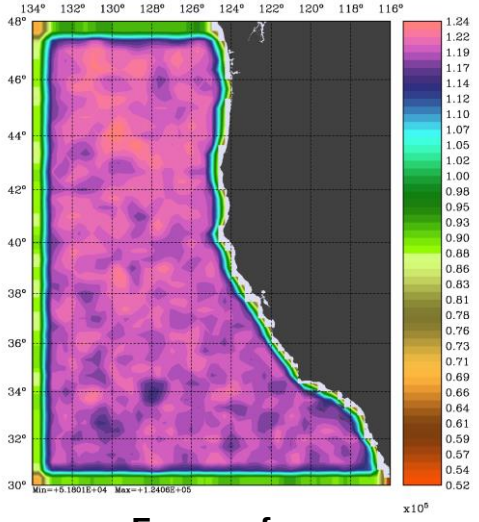


T_y

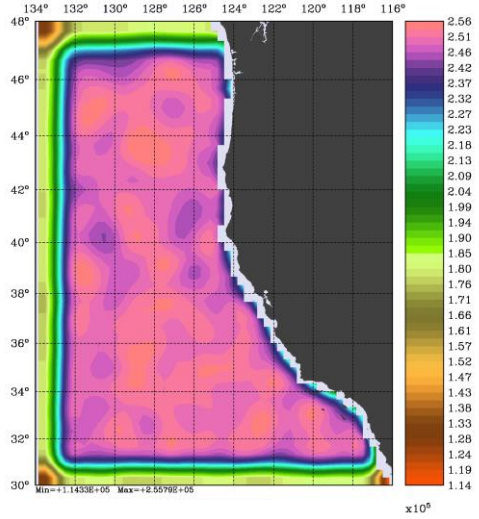


Net Heat Flux

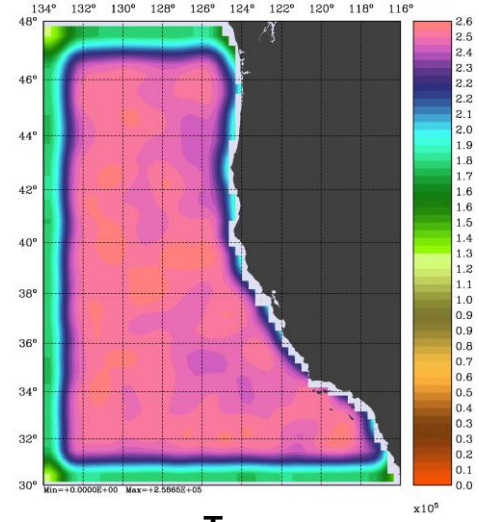
RANDOMIZATION



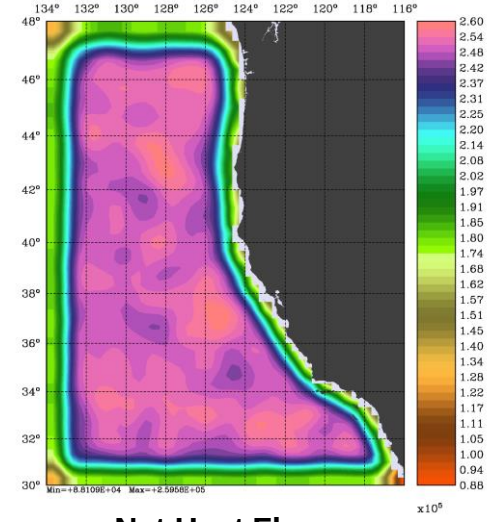
Free-surface



T_x



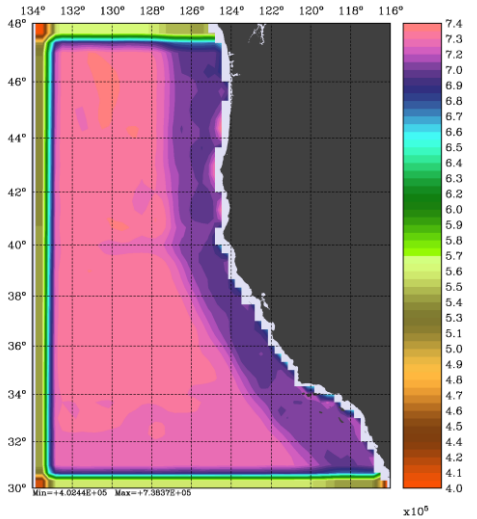
T_y



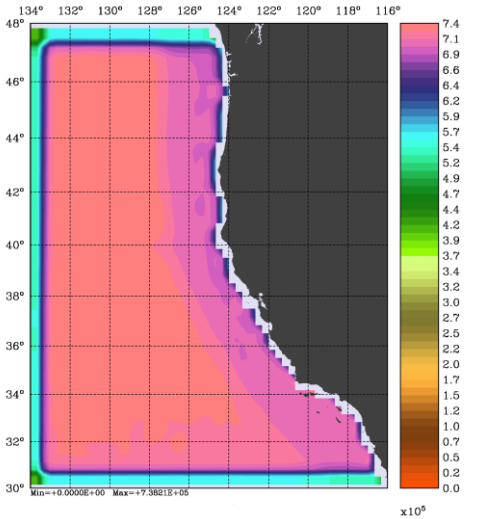
Net Heat Flux

Error Covariance Normalization Coefficients

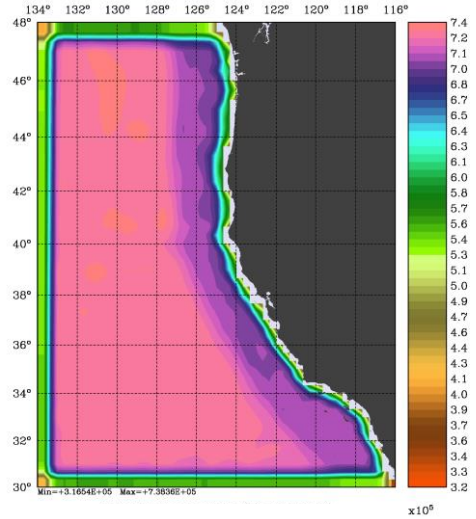
EXACT



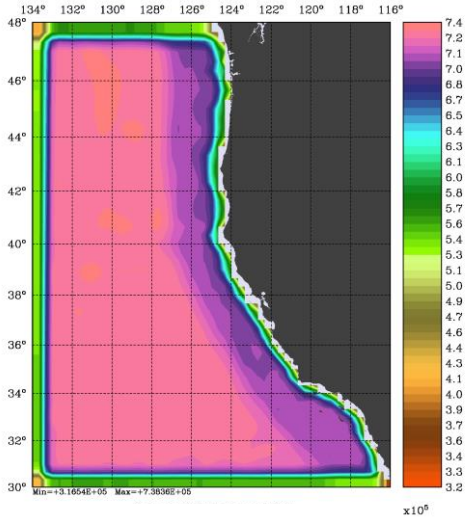
3D U-momentum



3D V-momentum

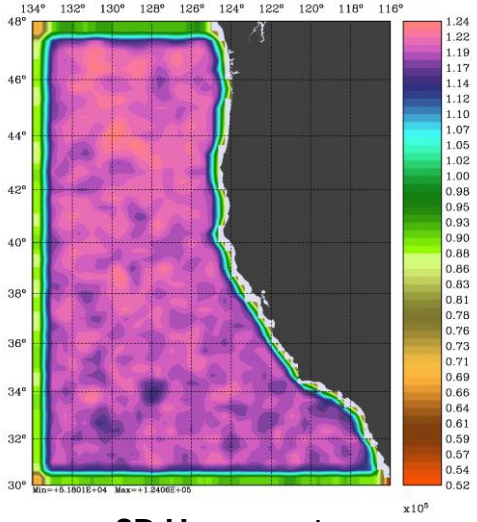


Temperature

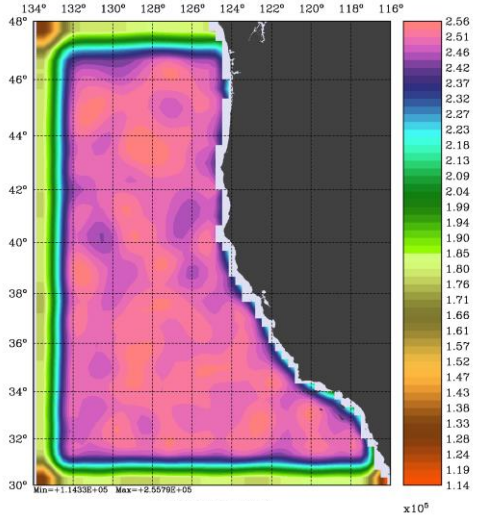


Salinity

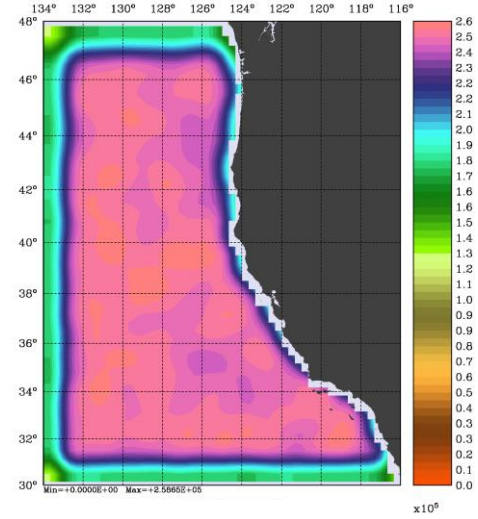
RANDOMIZATION



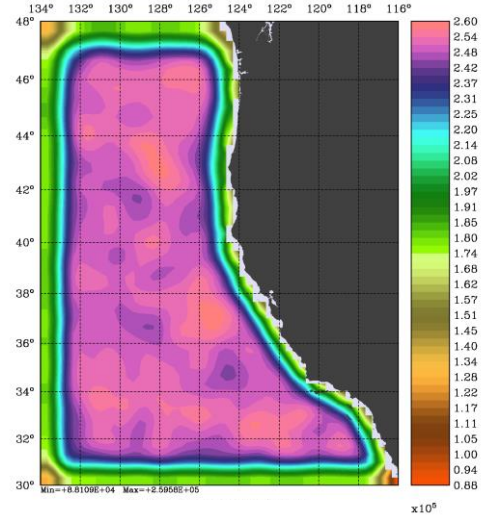
3D U-momentum



3D V-momentum



Temperature



Salinity

References

- **Fisher, M. and P. Courtier, 1995: Estimating the covariance matrices of analysis and forecast error in variational data assimilation. ECMWF Tech. Memo, 220.**
- **Weaver, A.T. and P. Courtier, 2001: Correlation modelling on the sphere using a generalized diffusion equation. *Q. J. R. Meteorol. Soc*., 127, 1815-1846.**