Tutorial 9:
Normalization Factors for *Prior* Error Correlations

Prior Error Covariance Modeling

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

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

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

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:

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

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

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

Computing Λ

(define NORMALIZATION)

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

(i) Exact method (Nmethod=0):

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

where $\mathbf{e}_i^{\mathrm{T}} = (0, 0, \ldots, 0, \underset{\substack{\uparrow \\ i^{th} \text{ element}}}{1}, 0, \cdots, 0)$

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 $\boldsymbol{\xi}$ **is a random vector:** $\boldsymbol{\xi} \rightarrow N(\mathbf{0},\mathbf{I})$

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

(Nrandom)

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

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

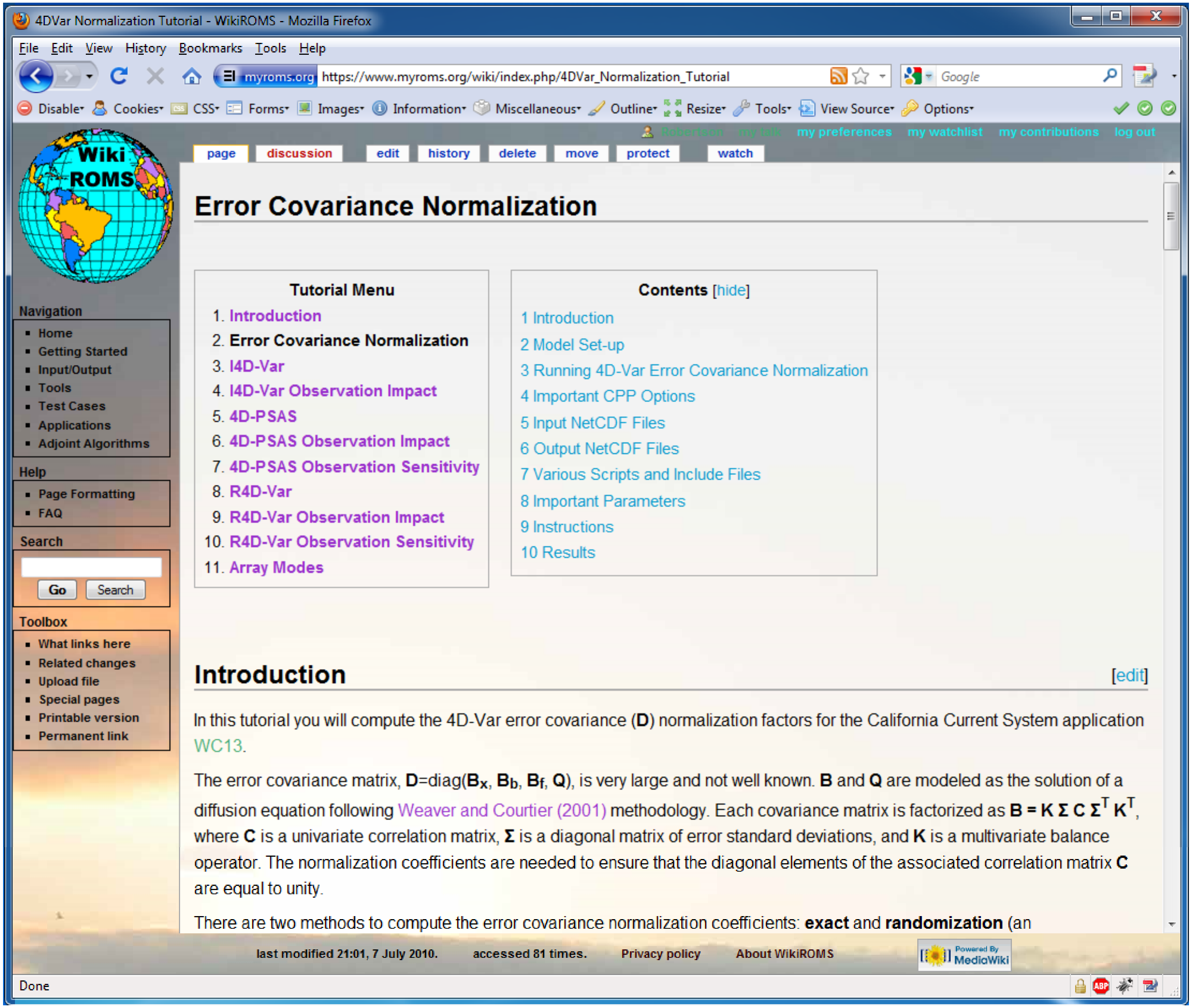
(Fisher and Courtier, 1995)

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



4DVar Normalization Tutorial - WikiROMS - Mozilla Firefox

File Edit View History Bookmarks Tools Help

https://www.myroms.org/wiki/index.php/4DVar_Normalization_Tutorial

my preferences my watchlist my contributions log out

page | discussion | edit | history | delete | move | protect | watch

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
- Upload file
- Special pages
- Printable version
- Permanent link

Error Covariance Normalization

Tutorial Menu

1. Introduction
2. Error Covariance Normalization
3. I4D-Var
4. I4D-Var Observation Impact
5. 4D-PSAS
6. 4D-PSAS Observation Impact
7. 4D-PSAS Observation Sensitivity
8. R4D-Var
9. R4D-Var Observation Impact
10. R4D-Var Observation Sensitivity
11. Array Modes

Contents [hide]

Introduction [edit]

In this tutorial you will compute the 4D-Var error covariance (**D**) normalization factors for the California Current System application WC13.

The error covariance matrix, **D**=diag($\mathbf{B_x}$, $\mathbf{B_b}$, $\mathbf{B_f}$, **Q**), is very large and not well known. **B** and **Q** are modeled as the solution of a diffusion equation following Weaver and Courtier (2001) methodology. Each covariance matrix is factorized as $\mathbf{B = K\, \Sigma\, C\, \Sigma^T K^T}$, where **C** is a univariate correlation matrix, **Σ** is a diagonal matrix of error standard deviations, and **K** is a multivariate balance operator. The normalization coefficients are needed to ensure that the diagonal elements of the associated correlation matrix **C** are equal to unity.

There are two methods to compute the error covariance normalization coefficients: **exact** and **randomization** (an

last modified 21:01, 7 July 2010. accessed 81 times. Privacy policy About WikiROMS Powered By MediaWiki

Done

WC13 C-preprocessing Options

(Basic Configuration)

Momentum Equations Options:

```
#define UV_ADV             including advection terms
#define UV_COR             including Coriolis term
#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
```

Tracers Equations Options:

```
#define TS_U3HADVECTION    3rd-order Upstream H. advection
#define TS_C4VADVECTION    4th-order Centered V. 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 ANA_BTFLUX         analytical bottom Temp flux
#define ANA_BSFLUX         analytical bottom Salt flux
```

Vertical Turbulent Mixing Parameterization:

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

Atmospheric Boundary Layer Parameterization:

```
#define BULK_FLUXES        Air/sea COARE bulk fluxes

#define DIURNAL_SRFLUX     imposing local diurnal cycle
#define SOLAR_SOURCE       solar radiation source term
#define LONGWAVE_OUT       compute outgoing long wave rad
#define EMINUSP            compute E-P
```

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 SPLINES            parabolic splines
                             reconstruction
```

Lateral Boundary Conditions:

```
#define EASTERN_WALL       closed eastern wall condition

#define WEST_FSCHAPMAN     free-surface, Chapman
#define WEST_M2FLATHER     2D momentum, Flather
#define WEST_M3CLAMPED     3D momentum, clamped
#define WEST_TCLAMPED      tracers, clamped condition

#define NORTH_FSCHAPMAN    free-surface, Chapman
#define NORTH_M2FLATHER    2D momentum, Flather
#define NORTH_M3CLAMPED    3D momentum, clamped
#define NORTH_TCLAMPED     tracers, clamped

#define SOUTH_FSCHAPMAN    free-surface, Chapman
#define SOUTH_M2FLATHER    2D momentum, Flather
#define SOUTH_M3CLAMPED    3D momentum, clamped
#define SOUTH_TCLAMPED     tracer, clamped

#define SPONGE             enhanced viscosity/diffusion
                             areas
```

WC13 C-preprocessing Options
(Error Covariance Normalization)

Algorithm:

```
#define Normalization        compute 4D-Var error covariance normalization coefficients
```

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
#define 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, RPM, and ADM
```

WC13 C-preprocessing Options
(Error Covariance Normalization)

Algorithm:

```
#define Normalization         compute 4D-Var error covariance normalization coefficients
```

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
#define 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, RPM, and ADM
```

Include File: wc13.h

```
/*
** svn $Id: wc13.h 476 2010-06-26 20:25:30Z arango $
*******************************************************************************
** Copyright (c) 2002-2010 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:       ocean_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
**  IS4DVAR                   Incremental, strong constraint 4DVAR
**  NLM_DRIVER                Nonlinear Basic State trajectory
**  OPT_PERTURBATION          Optimal perturbations
**  PICARD_TEST               Picard Iterations Test
**  R_SYMMETRY                Representer Matrix Symmetry Test
**  SANITY_CHECK              Sanity Check
**  SO_SEMI                   Stochastic Optimals: Semi-norm
```

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/TOMS 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: ocean_wc13.in

```
!
!  ROMS/TOMS Standard Input parameters.
!
!svn $Id: ocean_wc13.in 476 2010-06-26 20:25:30Z arango $
!========================================================= Hernan G. Arango ===
!  Copyright (c) 2002-2010 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,  !
! "Ngrids" entries are expected for some of these parameters.  In such case,  !
! the order of the entries for a parameter is extremely important.  It  must  !
```

c4dvar.in Important Parameters

```
Nmethod  == 0                  ! normalization method (0: exact, 1:randomization)
Nrandom  == 5000               ! randomization iterations
. . .
LdefNRM == T T T T             ! Create a new normalization files
LwrtNRM == T T T T             ! Compute and write normalization
. . .
CnormI(isFsur) =  T            ! 2D variable at RHO-points
CnormI(isUbar) =  T            ! 2D variable at U-points
CnormI(isVbar) =  T            ! 2D variable at V-points
CnormI(isUvel) =  T            ! 3D variable at U-points
CnormI(isVvel) =  T            ! 3D variable at V-points
CnormI(isTvar) =  T T          ! NT tracers
. . .
CnormB(isFsur) =  T            ! 2D variable at RHO-points
CnormB(isUbar) =  T            ! 2D variable at U-points
CnormB(isVbar) =  T            ! 2D variable at V-points
CnormB(isUvel) =  T            ! 3D variable at U-points
CnormB(isVvel) =  T            ! 3D variable at V-points
CnormB(isTvar) =  T T          ! NT tracers
. . .
CnormF(isUstr) =  T            ! surface U-momentum stress
CnormF(isVstr) =  T            ! surface V-momentum stress
CnormF(isTsur) =  T T          ! NT surface tracers flux
. . .
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)
```

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

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

4D-Var Parameters: Decorrelation Scales

Initial conditions correlations (m):

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

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

Surface forcing correlations (m):

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

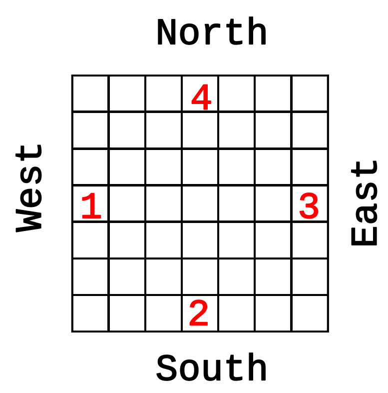
4D-Var Parameters: Decorrelation Scales

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 1256 2010-06-12 21:59:26Z arango $
!========================================================= Hernan G. Arango ===
!  Copyright (c) 2002-2010 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,  !
! "Ngrids" entries are expected for some of these parameters.  In such case,  !
! the order of the entries for a parameter is extremely important.  It  must  !
! follow the same order (1:Ngrids) as in the state variable declaration. The  !
```

Job Script: job_normalization.sh

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. Copy nonlinear model initial conditions file.

```
cp -p ${Dir}/Data/wc13_ini.nc wc13_ini.nc
```

4. 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
```

5. 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
```

6. 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 ocean_std_i.nc $STDnameI
$SUBSTITUTE $NORM ocean_std_b.nc $STDnameB
$SUBSTITUTE $NORM ocean_std_f.nc $STDnameF
$SUBSTITUTE $NORM ocean_nrm_i.nc $NRMnameI
$SUBSTITUTE $NORM ocean_nrm_b.nc $NRMnameB
$SUBSTITUTE $NORM ocean_nrm_f.nc $NRMnameF
$SUBSTITUTE $NORM ocean_obs.nc   $OBSname
$SUBSTITUTE $NORM ocean_hss.nc   wc13_hss.nc
$SUBSTITUTE $NORM ocean_lcz.nc   wc13_lcz.nc
$SUBSTITUTE $NORM ocean_mod.nc   wc13_mod.nc
$SUBSTITUTE $NORM ocean_err.nc   wc13_err.nc
```

Job Script File: job_normalization.sh

```
#!/bin/csh -f
#
# svn $Id: job_normalization.sh 474 2010-06-25 20:19:44Z arango $
#######################################################################
# Copyright (c) 2002-2010 The ROMS/TOMS Group                         #
#   Licensed under a MIT/X style license                              #
#   See License_ROMS.txt                                              #
#######################################################################
#                                                                     #
# 4D-Var error covariance normalization coeficients 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.

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

Compile: build.sh

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/arango/ocean/toms/repository
setenv MY_PROJECT_DIR       ${PWD}
```

2. Location of your ROMS source code.

```
setenv MY_ROMS_SRC          ${MY_ROOT_DIR}/branches/arango
```

3. Buld script invoked CPP options.

```
setenv MY_CPP_FLAGS "-DNORMALIZATION"
```

4. Compiler selection environment variables.

```
setenv USE_MPI             on
setenv USE_MPIF90          on
setenv FORT                pgi
```

5. Use custom library paths.

```
#setenv USE_MY_LIBS         on
```

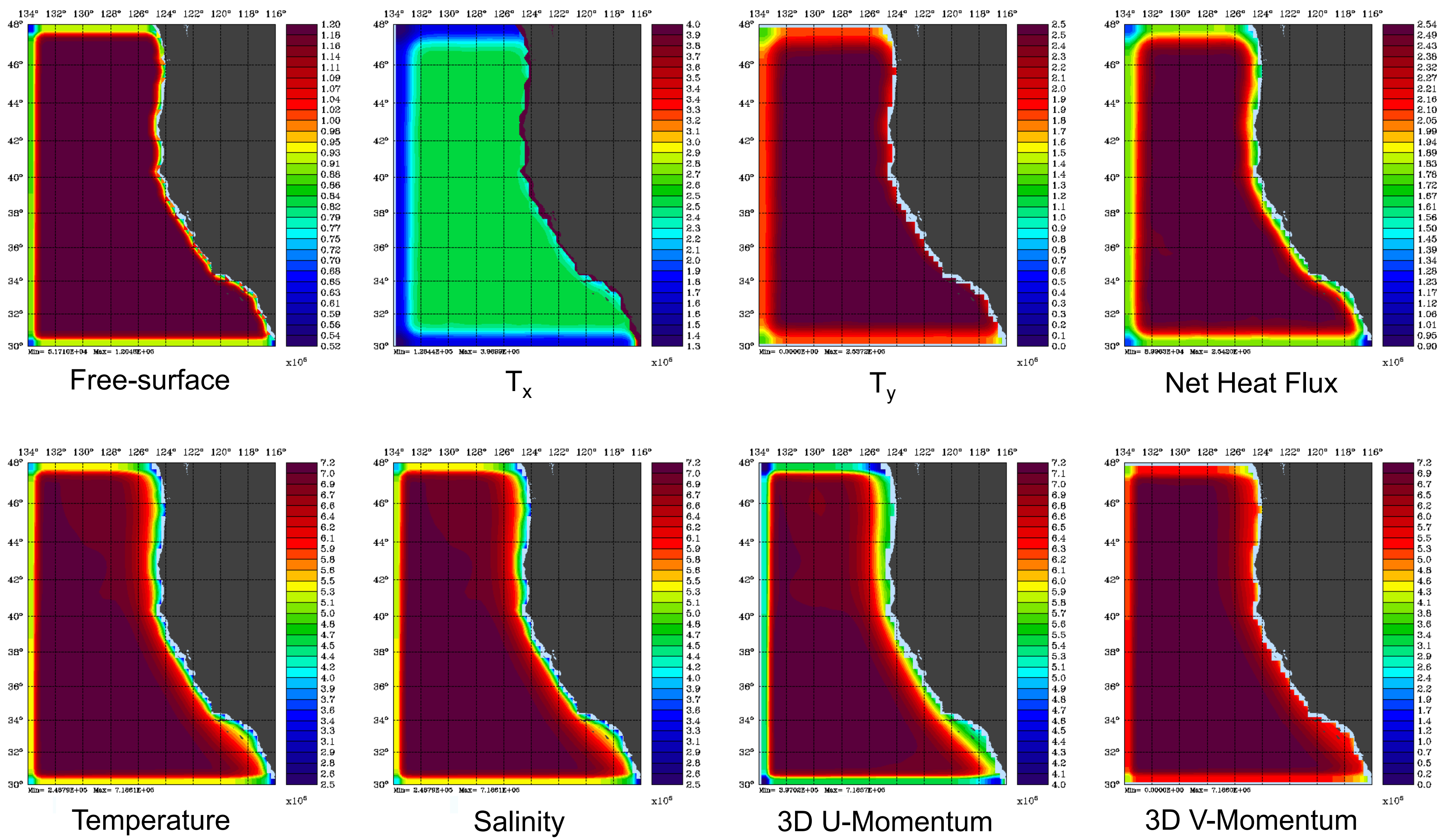
Libraries for PGI

```
if ($?USE_MY_LIBS) then
  switch ($FORT)
    case "pgi"
      setenv ARPACK_LIBDIR      /opt/pgisoft/serial/ARPACK
      if ($?USE_MPI) then
        setenv PARPACK_LIBDIR   /opt/pgisoft/mpich/PARPACK
      endif

      if ($?USE_NETCDF4) then
        if ($?USE_MPI) then
          setenv NETCDF_INCDIR  /opt/pgisoft/mpich/netcdf4/include
          setenv NETCDF_LIBDIR  /opt/pgisoft/mpich/netcdf4/lib
          setenv HDF5_LIBDIR    /opt/pgisoft/mpich/hdf5/lib
        else
          setenv NETCDF_INCDIR  /opt/pgisoft/serial/netcdf4/include
          setenv NETCDF_LIBDIR  /opt/pgisoft/serial/netcdf4/lib
          setenv HDF5_LIBDIR    /opt/pgisoft/serial/hdf5/lib
        endif
      else
        setenv NETCDF_INCDIR    /opt/pgisoft/serial/netcdf3/include
        setenv NETCDF_LIBDIR    /opt/pgisoft/serial/netcdf3/lib
      endif
    breaksw
```

Build Script: build.sh

```
#!/bin/csh -f
#
# svn $Id: build.sh 474 2010-06-25 20:19:44Z arango $
#:::::::::::::::::::::::::::::::::::::::::::::::::::::: John Wilkin :::
# Copyright (c) 2002-2010 The ROMS/TOMS Group                         :::
#   Licensed under a MIT/X style license                              :::
#   See License_ROMS.txt                                              :::
#::::::::::::::::::::::::::::::::::::::::::::::::::::: Hernan G. Arango :::
#                                                                     :::
# ROMS/TOMS Compiling 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.sh [options]                                             :::
#                                                                     :::
# Options:                                                            :::
#                                                                     :::
#    -j [N]      Compile in parallel using N CPUs                     :::
```

Error Covariance Normalization Coefficients



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.