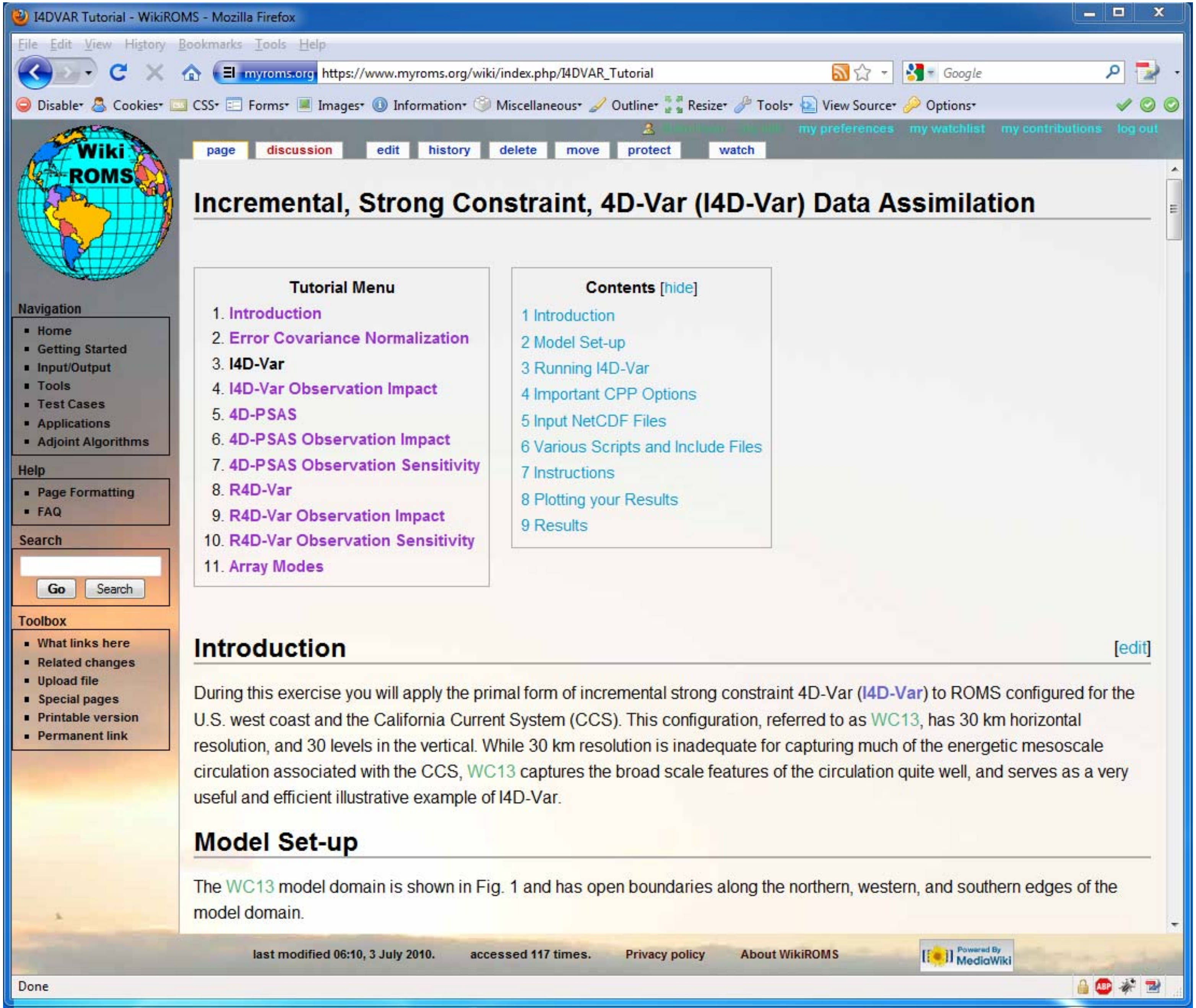
Tutorial 1: Explanation of CPP Options, ocean.in, and i4dvar.in

I4D-Var Tutorial Wiki Page



Incremental, Strong Constraint, 4D-Var (I4D-Var) Data Assimilation

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]

During this exercise you will apply the primal form of incremental strong constraint 4D-Var (I4D-Var) to ROMS configured for the U.S. west coast and the California Current System (CCS). This configuration, referred to as WC13, has 30 km horizontal resolution, and 30 levels in the vertical. While 30 km resolution is inadequate for capturing much of the energetic mesoscale circulation associated with the CCS, WC13 captures the broad scale features of the circulation quite well, and serves as a very useful and efficient illustrative example of I4D-Var.

Model Set-up

The WC13 model domain is shown in Fig. 1 and has open boundaries along the northern, western, and southern edges of the model domain.

last modified 06:10, 3 July 2010. accessed 117 times. Privacy policy About WikiROMS

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

(I4D-Var Configuration)

Algorithm:

```
#define IS4DVAR                primal form of incremental strong constraint 4D-Var
```

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 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, 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
```

ROMS Standard Input Parameters

```
    NtileI == 2             ! I-direction partition
    NtileJ == 2             ! J-direction partition
. . .
    NTIMES == 192           ! Number of time-steps (4 days)
        DT == 1800.0d0      ! Number of time-steps (48 steps per day)
. . .
    Nouter =  1             ! Number of 4D-Var outer loops
    Ninner =  50            ! Number of 4D-Var inner loops
. . .
   LDEFOUT == T             ! Switch to create new history files
      NHIS == 48            ! Steps between writing of NLM data (daily)
   NDEFHIS == 0             ! Steps between creation of new NLM files
. . .
 LcycleTLM == F             ! Switch to recycle records in TLM file
      NTLM == 48            ! Steps between writing of TLM data (daily)
   NDEFTLM == 0             ! Steps between creation of new TLM files
 LcycleADJ == T             ! Switch to recycle records in ADM file
      NADJ == 192           ! Steps between writing of ADM data (strong constraint)
   NDEFADJ == 0             ! Steps between creation of new ADM files
      NSFF == 48            ! Steps between adjustment of surface fluxes (daily)
      NOBC == 48            ! Steps between adjustment of open boundary (daily)
. .
   APARNAM =  i4dvar.in     ! I4D-Var standard input parameters
```

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  !
```

4D-Var Parameters: Normalization

```
Nmethod  == 0                 ! normalization method
Nrandom  == 5000              ! randomization iterations
. . .
LdefNRM  == F F F F           ! Create a new normalization files
LwrtNRM  == F F F F           ! 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)
```

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
```

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
```

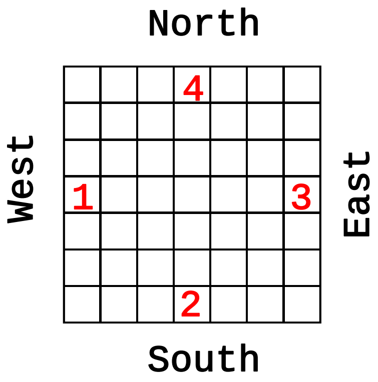
4D-Var Parameters: Decorrelation Scales

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
```



4D-Var Parameters: Balance Operator

SSH, elliptic solver:

```
          Nbico == 200                        ! biconjugate gradient iterarion
```

SSH, integration of hydrostatic equation:

```
      LNM_depth == 1000.0d0                   ! level of no motion (m, positive)

      LNM_flag   =  1                         ! Integration flag

                                                [0] integrate from bottom to surface
                                                [1] integrate from LNM to surface or
                                                    from local depth, if shallower
```

Balanced salinity empirical T-S relationship:

```
       dTdz_min == 0.001d0                    ! minimun dT/dz (Celsius/m)
       ml_depth == 100.0d0                    ! mixed-layer depth (m; positive)
```

State Variables switches:

```
balance(isSalt) =  T                          ! salinity
balance(isFsur) =  T                          ! free-sruface
balance(isVbar) =  F                          ! 2D momentum (ubar, vbar)
balance(isVvel) =  T                          ! 3D momentum (u, v)
```

Other 4D-Var Parameters

Lanczos algorithm parameters:

```
        GradErr = 1.0d-4                ! Upper bound on the relative error of the
                                          gradient

       HevecErr = 1.0d-1                ! Maximum error bound on Hessian eigenvectors

     LhessianEV = T                     ! Compute approximated hessian eigen pairs
```

Preconditioning:

```
       Lprecond = F                     ! Limitted-Memory Preconditioner: Spectral

          Lritz = T                     ! Limitted-Memory Preconditioner: Ritz

        NritzEV = 0                     ! If preconditioning, number of eigenvectors
                                          if NritzEV = 0, use HevecErr
```

Weak constraint:

```
      LhotStart = T                     ! Hot start in subsequent outer loops

         NpostI = 50                    ! Posterior error analysis Lanczos iterations

           Nvct = 50                    ! Stabilized representer matrix eigenvector
                                          to process

      tl_M2diff == 0.0d0                ! RPM relaxation (m2/s), 2D momentum
      tl_M3diff == 0.0d0                ! RPM relaxation (m2/s), 3D momentum

       tl_Tdiff == 0.0d0  0.0d0         ! RPM relaxation (m2/s), tracers
```

I4D-Var Parameters File: i4dvar.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  !
```

I4D-Var Job Script: job_i4dvar.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 initial conditions, boundary conditions and surface forcing error covariance standard deviations files.

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

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

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

6. Set observations file.

```
set OBSname = wc13_obs.nc
```

7. Get a clean copy of the observation file. This is really important since this file is modified.

```
cp -p ${Dir}/Data/${OBSname} .
```

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

```
set I4DVAR = i4dvar.in
if (-e $I4DVAR) then
  /bin/rm $I4DVAR
endif
cp s4dvar.in $I4DVAR

$SUBSTITUTE $I4DVAR ocean_std_i.nc $STDnameI
$SUBSTITUTE $I4DVAR ocean_std_b.nc $STDnameB
$SUBSTITUTE $I4DVAR ocean_std_f.nc $STDnameF
$SUBSTITUTE $I4DVAR ocean_nrm_i.nc $NRMnameI
$SUBSTITUTE $I4DVAR ocean_nrm_b.nc $NRMnameB
$SUBSTITUTE $I4DVAR ocean_nrm_f.nc $NRMnameF
$SUBSTITUTE $I4DVAR ocean_obs.nc   $OBSname
$SUBSTITUTE $I4DVAR ocean_hss.nc   wc13_hss.nc
$SUBSTITUTE $I4DVAR ocean_lcz.nc   wc13_lcz.nc
$SUBSTITUTE $I4DVAR ocean_mod.nc   wc13_mod.nc
$SUBSTITUTE $I4DVAR ocean_err.nc   wc13_err.nc
```

I4D-Var Job Script File: job_i4dvar.sh

```
#!/bin/csh -f
#
# svn $Id: job_i4dvar.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                                              #
#######################################################################
#                                                                     #
# Incremental strong constraint I4D-Var 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 initial conditions, boundary conditions, and surface  #
#       forcing error convariance input standard deviations files.    #
#   (3) Specify initial conditions, boundary conditions, and surface  #
#       forcing error convariance input/output normalization factors  #
#       files.                                                        #
#   (4) Copy a clean copy of the observations NetCDF file.            #
#   (5) Create 4D-Var input script "i4dvar.in" from template and      #
#       specify the error covariance standard deviation, error        #
#       covariance normalization factors, and observation 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 "-DIS4DVAR"
```

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                       :::
```