Tutorial 5: Explanation of CPP Options, roms.in, and rbl4dvar.in
Introduction

Notice: This algorithm began based on the Physical-space Statistical Analysis System (PSAS) algorithm but has evolved into a Restricted B-preconditioned Lanczos 4D-Var (RBL4D-Var). Some plots on this page still have the PSAS title but remain correct. The algorithm and ROMS CPP Options relating to this data assimilation system were renamed in SVN revision 1622 (May 13, 2020) and are explained in Trac ticket #854.

In this tutorial you will apply the strong/weak constraint, dual form of 4-Dimensional Variational (4D-Var) data assimilation based on the Restricted B-preconditioned Lanczos 4D-Var (RBL4DVar) algorithm (RBL4D-Var) to ROMS configured for the U.S. west coast and the California Current System (WCI3). In RBL4D-Var the search for the best ocean circulation estimate proceeds in the space spanned only by the observations, as opposed to the full space spanned by the model (i.e., the primal form, 4D-Var). Formally, the primal and dual formulations yield identical estimates of the ocean circulation so one might wonder if there is any advantage of one form over the other? The practical advantages and disadvantages to both approaches are discussed in Moore et al. (2011b, c).

Model Set-up

The WCI3 model domain is shown in Fig. 1 and has open boundaries along the northern, western, and southern edges of the model domain.
Momentum Equations Options:
#define UV_ADV including advection terms
#define UV_COR including Coriolis term
#define UV_U3ADV 3rd-order Upstream Horizontal advection
#define UV_C4ADV 4th-order Centered Vertical advection
#define DJ_GRADPS splines density Jacobian PGF
#define UV_QFRIC quadratic bottom friction
#define UV_VIS2 harmonic horizontal mixing
#define MIX_S_UV mixing along s-levels
#define SPLINES_VIS2C parabolic Splines for Vertical Viscosity

Tracers Equations Options:
#define TS_U3ADV 3rd-order Upstream horizontal advection
#define TS_C4ADV 4th-order Centered advection
#define TS_DF2 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

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
# define 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 ANAL_SPONGE analytical viscosity/diffusion sponge

WC13 C-preprocessing Options
(Basic Configuration)
## WC13 C-preprocessing Options

### (4D-PSAS Configuration)

**Algorithm:**

<table>
<thead>
<tr>
<th>Define</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>define RBL4DVAR</td>
<td>dual form strong/weak constraint RBL4D-Var</td>
</tr>
<tr>
<td>undef MINRES</td>
<td>Minimal Residual Method for minimization</td>
</tr>
<tr>
<td>undef RPCG</td>
<td>Restricted B-preconditioned Lanczos minimization</td>
</tr>
<tr>
<td>undef TIME_CONV</td>
<td>Weak-constraint 4D-Var time convolution</td>
</tr>
<tr>
<td>undef BGQC</td>
<td>background quality control of observations</td>
</tr>
<tr>
<td>undef POSTERIOR_EOFS</td>
<td>estimate posterior error analysis error cross-covariance EOFs</td>
</tr>
<tr>
<td>indef POSTERIOR_ERROR_I</td>
<td>estimate initial conditions posterior analysis error covariance</td>
</tr>
</tbody>
</table>

**Control Vector:**

<table>
<thead>
<tr>
<th>Define</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>define ADJUST_BOUNDARY</td>
<td>open boundary conditions increments</td>
</tr>
<tr>
<td>define ADJUST_STFLUX</td>
<td>surface tracer flux increments</td>
</tr>
<tr>
<td>define ADJUST_WSTRESS</td>
<td>surface wind stress increments</td>
</tr>
<tr>
<td>define NL_BULK_FLUXES</td>
<td>using bulk fluxes computed by nonlinear model</td>
</tr>
<tr>
<td>define TS_U3HADVECTION_TL</td>
<td>TL/AD 3rd-order upstream horizontal tracer advection</td>
</tr>
<tr>
<td>define TS_C4VADVECTION_TL</td>
<td>TL/AD 4th-order centered vertical tracer advection</td>
</tr>
</tbody>
</table>

**Error Covariance Modeling:**

<table>
<thead>
<tr>
<th>Define</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>define VCONVOLUTION</td>
<td>Vertical correlation modeling</td>
</tr>
<tr>
<td>define IMPLICIT_VCON</td>
<td>Implicit vertical diffusion operator</td>
</tr>
<tr>
<td>undef BALANCE_OPERATOR</td>
<td>Multivariate balance constraint</td>
</tr>
<tr>
<td>ifdef BALANCE_OPERATOR</td>
<td></td>
</tr>
<tr>
<td># define ZETA_ELLIPTIC</td>
<td>SSH elliptic equation method</td>
</tr>
</tbody>
</table>

**Prior:**

<table>
<thead>
<tr>
<th>Define</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>define FORWARD_READ</td>
<td>read basic state linearization in TLM and ADM files</td>
</tr>
<tr>
<td>define FORWARD_WRITE</td>
<td>writing basic state by the NLM</td>
</tr>
<tr>
<td>define FORWARD_MIXING</td>
<td>processing basic state vertical mixing coefficients</td>
</tr>
<tr>
<td>define NL_BULK_FLUXES</td>
<td>surface kinematic fluxes from nonlinear model</td>
</tr>
</tbody>
</table>

**I/O:**

<table>
<thead>
<tr>
<th>Define</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>define OUT_DOUBLE</td>
<td>double precision data in output NLM, TLM, and ADM</td>
</tr>
</tbody>
</table>
** 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
ROMS Standard Input Parameters:  roms_wc13.in

NtileI == 2     ! I-direction partition
NtileJ == 4     ! 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 == 26    ! 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)
   NSFF == 48   ! Steps between adjustment of surface fluxes (daily)
   NOBC == 48   ! Steps between adjustment of open boundary (daily)
   NDEFADJ == 0 ! Steps between creation of new ADM files

APARNAM = psas.in ! I4D-Var standard input parameters
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.
4D-Var Parameters: Normalization

Nmethod == 0
! normalization method: 0=Exact (expensive) or 1=Approximated (randomization)
Nrrandom == 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

NRNameM == wc13_nrm_m.nc
! model error (weak constraint)
NRNameI == wc13_nrm_i.nc
! initial conditions
NRNameB == wc13_nrm_b.nc
! open boundary conditions
NRNameF == 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):

<table>
<thead>
<tr>
<th></th>
<th>IC</th>
<th>Model</th>
<th>OBC</th>
<th>Sur</th>
<th>For</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H_{\gamma})</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>! horizontal operator</td>
</tr>
<tr>
<td>(V_{\gamma})</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0005</td>
<td>! vertical operator</td>
</tr>
</tbody>
</table>

Model error correlations (m):

| \(H_{\text{decayM}}\) (isFsur) | 50.0d+3 | ! free-surface (16 convolutions) |
| \(H_{\text{decayM}}\) (isUbar) | 50.0d+3 | ! 2D U-momentum (16 convolutions) |
| \(H_{\text{decayM}}\) (isVbar) | 50.0d+3 | ! 2D V-momentum (16 convolutions) |
| \(H_{\text{decayM}}\) (isUvel) | 50.0d+3 | ! 3D U-momentum (16 convolutions) |
| \(H_{\text{decayM}}\) (isVvel) | 50.0d+3 | ! 3D V-momentum (16 convolutions) |
| \(H_{\text{decayM}}\) (isTvar) | 50.0d+3 50.0d+3 | ! 1:NT tracers (16 convolutions) |
| \(V_{\text{decayM}}\) (isUvel) | 30.0d0 | ! 3D U-momentum (8 convolutions) |
| \(V_{\text{decayM}}\) (isVvel) | 30.0d0 | ! 3D V-momentum (8 convolutions) |
| \(V_{\text{decayM}}\) (isTvar) | 30.0d0 30.0d0 | ! 1:NT tracers (8 convolutions) |

Initial conditions error correlations (m):

| \(H_{\text{decayI}}\) (isFsur) | 50.0d+3 | ! free-surface (16 convolutions) |
| \(H_{\text{decayI}}\) (isUbar) | 50.0d+3 | ! 2D U-momentum (16 convolutions) |
| \(H_{\text{decayI}}\) (isVbar) | 50.0d+3 | ! 2D V-momentum (16 convolutions) |
| \(H_{\text{decayI}}\) (isUvel) | 50.0d+3 | ! 3D U-momentum (16 convolutions) |
| \(H_{\text{decayI}}\) (isVvel) | 50.0d+3 | ! 3D V-momentum (16 convolutions) |
| \(H_{\text{decayI}}\) (isTvar) | 50.0d+3 50.0d+3 | ! 1:NT tracers (16 convolutions) |
| \(V_{\text{decayI}}\) (isUvel) | 30.0d0 | ! 3D U-momentum (8 convolutions) |
| \(V_{\text{decayI}}\) (isVvel) | 30.0d0 | ! 3D V-momentum (8 convolutions) |
| \(V_{\text{decayI}}\) (isTvar) | 30.0d0 30.0d0 | ! 1:NT tracers (8 convolutions) |

Surface forcing error correlations (m):

| \(H_{\text{decayF}}\) (isUstr) | 100.0d+3 | ! surface U-momentum stress (66 convolutions) |
| \(H_{\text{decayF}}\) (isVstr) | 100.0d+3 | ! surface V-momentum stress (66 convolutions) |
| \(H_{\text{decayF}}\) (isTsor) | 100.0d+3 100.0d+3 | ! 1:NT surface tracer flux (66 convolutions) |
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:**

\[
\text{Nbico} = 200 \quad \text{! biconjugate gradient iteration}
\]

**SSH, integration of hydrostatic equation:**

\[
\begin{align*}
\text{LNM\_depth} & = 1000.0 \text{d0} \quad \text{! level of no motion (m, positive)} \\
\text{LNM\_flag} & = 1 \quad \text{! Integration flag}
\end{align*}
\]

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

**Balanced salinity empirical T-S relationship:**

\[
\begin{align*}
\text{dTdz\_min} & = 0.001 \text{d0} \quad \text{! minimum dT/dz (Celsius/m)} \\
\text{ml\_depth} & = 100.0 \text{d0} \quad \text{! mixed-layer depth (m; positive)}
\end{align*}
\]

**State Variables switches:**

\[
\begin{align*}
\text{balance(isSalt)} & = T \quad \text{! salinity} \\
\text{balance(isFsur)} & = T \quad \text{! free-surface} \\
\text{balance(isVbar)} & = F \quad \text{! 2D momentum (ubar, vbar)} \\
\text{balance(isVvel)} & = T \quad \text{! 3D momentum (u, v)}
\end{align*}
\]
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` ! Limited-Memory Preconditioner: Spectral
- `Lritz = T` ! Limited-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 = 25` ! Posterior error analysis Lanczos iterations
- `NVct = 10` ! Stabilized representer matrix eigenvector to process
4DVar assimilation input parameters.

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
```

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 cases,
RBL4D-Var Job Script: job_rbl4dvar.csh

1. Set path definition to one directory up in the tree.
   ```bash
   set Dir = `dirname ${PWD}`
   ```
2. Set string manipulations perl script.
   ```bash
   set SUBSTITUTE = ${ROMS_ROOT}/ROMS/Bin/substitute
   ```
3. Copy nonlinear model initial conditions file.
   ```bash
   cp -p ${Dir}/Data/wc13_ini.nc wc13_ini.nc
   ```
4. Set model, initial conditions, boundary conditions and surface forcing error covariance standard deviations files.
   ```bash
   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, initial conditions, boundary conditions and surface forcing error covariance normalization factors files.
   ```bash
   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
   ```
   ```bash
   set OBSname = wc13_obs.nc
   ```
7. Get a clean copy of the observation file. This is really important since this file is modified.
   ```bash
   cp -p ${Dir}/Data/${OBSname} .
   ```
8. Modify 4D-Var template input script and specify above files.
   ```bash
   set RBL4DVAR = rbl4dvar.in
   if (-e $RBL4DVAR) then
     /bin/rm $RBL4DVAR
   endif
   cp s4dvar.in $RBL4DVAR
   ```
   ```bash
   $SUBSTITUTE $RBL4DVAR roms_std_m.nc $STDnameM
   $SUBSTITUTE $RBL4DVAR roms_std_i.nc $STDnameI
   $SUBSTITUTE $RBL4DVAR roms_std_b.nc $STDnameB
   $SUBSTITUTE $RBL4DVAR roms_std_f.nc $STDnameF
   $SUBSTITUTE $RBL4DVAR roms_nrm_m.nc $NRMnameM
   $SUBSTITUTE $RBL4DVAR roms_nrm_i.nc $NRMnameI
   $SUBSTITUTE $RBL4DVAR roms_nrm_b.nc $NRMnameB
   $SUBSTITUTE $RBL4DVAR roms_nrm_f.nc $NRMnameF
   $SUBSTITUTE $RBL4DVAR roms_obs.nc $OBSname
   $SUBSTITUTE $RBL4DVAR roms_hss.nc wc13_hss.nc
   $SUBSTITUTE $RBL4DVAR roms_lcz.nc wc13_lcz.nc
   $SUBSTITUTE $RBL4DVAR roms_mod.nc wc13_mod.nc
   $SUBSTITUTE $RBL4DVAR roms_err.nc wc13_err.nc
   ```
#!/bin/csh -f
#
# svn $Id: job_rbl4dvar.csh 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
#
# Strong/Weak constraint RBL4D-Var job CSH 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 covariance input standard deviations files.
# (3) Specify model, initial conditions, boundary conditions, and surface forcing error covariance input/output normalization factors files.
# (4) Copy a clean copy of the observations NetCDF file.
# (5) Create 4D-Var input script "rbl4dvar.in" from template and specify the error covariance standard deviation, error covariance normalization factors, and observation files to be used.
#
Compile: build_roms.sh

1. Set a local environmental variable to define the path to the directories where all this project's files are kept.
   ```
   export MY_ROOT_DIR=${HOME}/ocean/toms/repository
   export MY_PROJECT_DIR=${PWD}
   ```

2. Location of your ROMS source code.
   ```
   export MY_ROMS_SRC=${MY_ROOT_DIR}/trunk
   ```

3. Path of Makefile configuration (*mk) files
   ```
   export COMPILERS=${MY_ROMS_SRC}/Compilers
   export COMPILERS=${HOME}/Compilers/ROMS
   ```

4. Build script invoked CPP options.
   ```
   export MY_CPP_FLAGS="${MY_CPP_FLAGS} -DRBL4DVAR"
   export MY_CPP_FLAGS="${MY_CPP_FLAGS} -DANA_SPONGE"
   #export MY_CPP_FLAGS="${MY_CPP_FLAGS} -DMINRES"
   #export MY_CPP_FLAGS="${MY_CPP_FLAGS} -DTIME_CONV"
   #export MY_CPP_FLAGS="${MY_CPP_FLAGS} -DBGQC"
   #export MY_CPP_FLAGS="${MY_CPP_FLAGS} -DPHASE" # Nouter=1
   #export MY_CPP_FLAGS="${MY_CPP_FLAGS} -DPHASE" # Nouter=1
   #export MY_CPP_FLAGS="${MY_CPP_FLAGS} -DCOLLECT_ALLREDUCE"
   #export MY_CPP_FLAGS="${MY_CPP_FLAGS} -DREDUCE_ALLGATHER"
   #export MY_CPP_FLAGS="${MY_CPP_FLAGS} -DEDEBUGGING"
   #export MY_CPP_FLAGS="${MY_CPP_FLAGS} -DPOSITIVE_ZERO"
   ```

5. Compiler selection environment variables.
   ```
   export USE_MPI=on
   export USE_MPIF90=on
   export which_MPI=openmp
   ```

6. Use custom library paths
   ```
   export USE_MY_LIBS=no    # use system default library paths
   #export USE_MY_LIBS=yes  # use my customized library paths
   MY_PATHS=${COMPILERS}/my_build_paths.bash
   if [ "$USE_MY_LIBS" = "yes" ]; then
     source $MY_PATHS $MY_PATHS
   fi
#!/bin/bash
#
# Q: How/why does this script work?
#
# A: The ROMS makesfile configures user-defined options with a set of
#    flags such as ROMS_APPLICATION. Browse the makesfile to see these.
#    If an option in the makesfile 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 makesfiles, or frequently edit
#    the makesfile, to run separate applications).
#
# Usage:
#
#   ./build_roms.sh [options]
#
# Options:

#!/bin/bash
#
# svn $Id$
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# Copyright (c) 2002-2020 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 bash 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.bash we have:
#: if [ "${USE_MY_LIBS}" = "yes" ]; then
#    source ${COMPILERS}/my_build_paths.sh
#: fi
#:separator=`perl -e "print ':' x 100;"`

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