Tutorial 5:
Explanation of CPP Options,
roms.in, and psas.in
**Physical-space Statistical Analysis System (4D-PSAS)**

### Tutorial Menu

1. Introduction
2. Error Covariance Normalization
3. 4D-VAR
4. 4D-PSAS
5. 4D-PSAS Analysis Observation Impact
6. 4D-PSAS Analysis Observation Sensitivity
7. Array Modes
8. 4D-PSAS Forecast Observation Impact
9. 4D-PSAS Forecast Observation Sensitivity

### Contents

1. Introduction
2. Model Set-up
3. Running 4D-PSAS
4. Important CPP Options
5. Input NetCDF Files
6. Various Scripts and Include Files
7. Instructions
8. Plotting your Results
9. Results

**Introduction**

In this tutorial you will apply the strong/weak constraint, dual form of 4-Dimensional Variational (4D-Var) data assimilation based on the Physical-space Statistical Analysis System (PSAS) algorithm (4D-PSAS) to ROMS configured for the U.S. west coast and the California Current System (WC13). In 4D-PSAS 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 WC13 model domain is shown in Fig. 1 and has open boundaries along the northern, western, and southern edges of the model domain.
WC13 C-preprocessing Options
(Basic Configuration)

Momentum Equations Options:

#define UV_ADV including advection terms
#define UV_COR including Coriolis term
#define UV_U3HADV including 3rd-order Upstream Horizontal advection
#define UV_C4VADV including 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_WVIS parabolic Splines for Vertical Viscosity

Tracers Equations Options:

#define TS_U3HADV including 3rd-order Upstream Horizontal advection
#define TS_C4VADV including 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
#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 ANA_SPONGE analytical viscosity/diffusion sponge

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 ANA_SPONGE analytical viscosity/diffusion sponge
WC13 C-preprocessing Options
(4D-PSAS Configuration)

Algorithm:
# define W4DPSAS dual form strong/weak constraint 4D_PSAS
# undef MINRES Minimal Residual Method for minimization
# undef RPCG Restricted B-preconditioned Lanczos minimization
# undef TIME_CONV Weak-constraint 4D-Var time convolution
# undef BGQC background quality control of observations
# undef POSTERIOR_EOFS estimate posterior error analysis error cross-covariance EOFs
# indef POSTERIOR_ERROR_I estimate initial conditions posterior analysis error covariance

Control Vector:
# define ADJUST_BOUNDARY open boundary conditions increments
# define ADJUST_STFLUX surface tracer flux increments
# define ADJUST_WSTRESS surface wind stress increments
# define NL_BULK_FLUXES using bulk fluxes computed by nonlinear model
# define TS_U3HADVECTION_TL TL/AD 3rd-order upstream horizontal tracer advection
# define TS_C4VADVECTION_TL TL/AD 4th-order centered vertical tracer advection

Error Covariance Modeling:
# define VCONVOLUTION Vertical correlation modeling
# define IMPLICIT_VCON Implicit vertical diffusion operator

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
Options for the California Current System, 1/3 degree resolution.

Application flag: WC13
Input script: roms_wc13.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
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)
! NADJ == 48    ! Steps between writing of ADM data (weak 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 = psas.in  ! 4D-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.

In multiple levels of nesting or multiple connected domains step-ups,
**4D-Var Parameters: Normalization**

- **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)
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>Hgamma</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>Vgamma</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0005</td>
<td></td>
</tr>
</tbody>
</table>

Model error correlations (m):

<table>
<thead>
<tr>
<th></th>
<th>HdecayM(isFsur) == 50.0d+3</th>
<th>HdecayM(isUbar) == 50.0d+3</th>
<th>HdecayM(isVbar) == 50.0d+3</th>
<th>HdecayM(isUvel) == 50.0d+3</th>
<th>HdecayM(isVvel) == 50.0d+3</th>
<th>HdecayM(isTvar) == 50.0d+3 50.0d+3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>free-surface (16 convolutions)</td>
<td>2D U-momentum (16 convolutions)</td>
<td>2D V-momentum (16 convolutions)</td>
<td>3D U-momentum (16 convolutions)</td>
<td>3D V-momentum (16 convolutions)</td>
<td>1:NT tracers (16 convolutions)</td>
</tr>
<tr>
<td></td>
<td>VdecayM(isUvel) == 30.0d0</td>
<td>VdecayM(isVvel) == 30.0d0</td>
<td>VdecayM(isTvar) == 30.0d0 30.0d0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3D U-momentum (8 convolutions)</td>
<td>3D V-momentum (8 convolutions)</td>
<td>1:NT tracers (8 convolutions)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Initial conditions error correlations (m):

<table>
<thead>
<tr>
<th></th>
<th>HdecayI(isFsur) == 50.0d+3</th>
<th>HdecayI(isUbar) == 50.0d+3</th>
<th>HdecayI(isVbar) == 50.0d+3</th>
<th>HdecayI(isUvel) == 50.0d+3</th>
<th>HdecayI(isVvel) == 50.0d+3</th>
<th>HdecayI(isTvar) == 50.0d+3 50.0d+3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>free-surface (16 convolutions)</td>
<td>2D U-momentum (16 convolutions)</td>
<td>2D V-momentum (16 convolutions)</td>
<td>3D U-momentum (16 convolutions)</td>
<td>3D V-momentum (16 convolutions)</td>
<td>1:NT tracers (16 convolutions)</td>
</tr>
<tr>
<td></td>
<td>VdecayI(isUvel) == 30.0d0</td>
<td>VdecayI(isVvel) == 30.0d0</td>
<td>VdecayI(isTvar) == 30.0d0 30.0d0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3D U-momentum (8 convolutions)</td>
<td>3D V-momentum (8 convolutions)</td>
<td>1:NT tracers (8 convolutions)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Surface forcing error correlations (m):

<table>
<thead>
<tr>
<th></th>
<th>HdecayF(isUstr) == 100.0d+3</th>
<th>HdecayF(isVstr) == 100.0d+3</th>
<th>HdecayF(isTsur) == 100.0d+3 100.0d+3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>surface U-momentum stress (66 convolutions)</td>
<td>surface V-momentum stress (66 convolutions)</td>
<td>1:NT surface tracer flux (66 convolutions)</td>
</tr>
</tbody>
</table>
## 4D-Var Parameters: Decorrelation Scales

**Open boundary conditions correlations (m):**

<table>
<thead>
<tr>
<th></th>
<th>1: west</th>
<th>2: south</th>
<th>3: east</th>
<th>4: north</th>
</tr>
</thead>
<tbody>
<tr>
<td>HdecayB(isFsur)</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
</tr>
<tr>
<td>HdecayB(isUbar)</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
</tr>
<tr>
<td>HdecayB(isVbar)</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
</tr>
<tr>
<td>HdecayB(isUvel)</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
</tr>
<tr>
<td>HdecayB(isVvel)</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
<td>100.0d+3</td>
</tr>
<tr>
<td>HdecayB(isTvar)</td>
<td>4*100.0d+3</td>
<td>4*100.0d+3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Boundary edges to adjust (logical switches):**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lobc(isFsur)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>Lobc(isUbar)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>Lobc(isVbar)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>Lobc(isUvel)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>Lobc(isVvel)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>Lobc(isTvar)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

- **West:** 1
- **South:** 2
- **East:** 3
- **North:** 4
**4D-Var Parameters: Balance Operator**

**SSH, elliptic solver:**

\[ Nbico = 200 \]  
  ! biconjugate gradient iteration

**SSH, integration of hydrostatic equation:**

\[ LNM_{\text{depth}} = 1000.0d0 \]  
  ! level of no motion (m, positive)

\[ LNM_{\text{flag}} = 1 \]  
  ! 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:**

\[ dTdz_{\text{min}} = 0.001d0 \]  
  ! minimum \( dT/dz \) (Celsius/m)

\[ ml_{\text{depth}} = 100.0d0 \]  
  ! mixed-layer depth (m; positive)

**State Variables switches:**

balance(isSalt) = T  
  ! salinity

balance(isFsur) = T  
  ! free-surface

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  
  ! 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.

svn $Id: s4dvar.in 971 2019-07-09 03:02:57Z 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, "Ngrids" entries are expected for some of these parameters. In such case,
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 PSAS = psas.in
   if (-e $PSAS) then
     /bin/rm $PSAS
   endif
   cp s4dvar.in $PSAS
   ```
   ```bash
   $SUBSTITUTE $PSAS roms_std_m.nc $STDnameM
   $SUBSTITUTE $PSAS roms_std_i.nc $STDnameI
   $SUBSTITUTE $PSAS roms_std_b.nc $STDnameB
   $SUBSTITUTE $PSAS roms_std_f.nc $STDnameF
   $SUBSTITUTE $PSAS roms_nrm_m.nc $NRMnameM
   $SUBSTITUTE $PSAS roms_nrm_i.nc $NRMnameI
   $SUBSTITUTE $PSAS roms_nrm_b.nc $NRMnameB
   $SUBSTITUTE $PSAS roms_nrm_f.nc $NRMnameF
   $SUBSTITUTE $PSAS roms_obs.nc $OBSname
   $SUBSTITUTE $PSAS roms_hss.nc wc13_hss.nc
   $SUBSTITUTE $PSAS roms_lcz.nc wc13_lcz.nc
   $SUBSTITUTE $PSAS roms_mod.nc wc13_mod.nc
   $SUBSTITUTE $PSAS roms_err.nc wc13_err.nc
   ```
#!/bin/csh -f
#
# svn $Id: job_psas.sh 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
#
# Strong/Weak constraint 4D-PSAS 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 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 "psas.in" from template and specify the error covariance standard deviation, error covariance normalization factors, and observation files to be used.
#
#
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="-D4DPSAS"
   export MY_CPP_FLAGS="-DANA_SPONGE"
   #export MY_CPP_FLAGS="-DMINRES"
   #export MY_CPP_FLAGS="-DTIME_CONV"
   #export MY_CPP_FLAGS="-DBGQC"
   #export MY_CPP_FLAGS="-DPOSTERIOR_EOFS" # Nouter=1
   #export MY_CPP_FLAGS="-DPOSTERIOR_ERROR_I" # Nouter=1
   #export MY_CPP_FLAGS="-DCOLLECT_ALLREDUCE"
   #export MY_CPP_FLAGS="-DRIGHT_ALLGATHER"
   #export MY_CPP_FLAGS="-DDEBUGGING"
   #export MY_CPP_FLAGS="-DPOSITIVE_ZERO"
   ```

5. Compiler selection environment variables.

   ```
   export USE_MPI=on
   export USE_MPIF90=on
   export which_MPI=openmp
   export FORT=ifort
   ```

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
   ```
#!/bin/bash

# svn $Id: build_roms.bash 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 BASH 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.bash [options]
Build Script: my_build_paths.bash

# 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 bash script sets the customized library paths needed by the
# build script when the environment 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.bash
#       fi
#
separator=`perl -e "print ':' x 100;"`

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

# Add MPI library to compile.

#---------------------------------------------------------------