Tutorial 1: Explanation of CPP Options, roms.in, and i4dvar.in
Incremental, Strong Constraint, 4D-Var (I4D-Var) Data Assimilation

Tutorial Menu
1. Introduction
2. Error Covariance Normalization
3. I4D-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 [hide]
1. Introduction
2. Model Setup
3. Running I4D-Var
4. Important CPP Options
5. Input NetCDF Files
6. Various Scripts and Include Files
7. Instructions
8. Plotting your Results
9. Results

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 [edit]
The WC13 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_U3HADVECTION 3rd-order Upstream Horizontal advection
#define UV_C4VADVECTION 4th-order Centered Vertical advection
#define DJ_GRADPS splines density Jacobian PGF
#define UV_QDRAG quadratic bottom friction
#define UV_VIS2 harmonic horizontal mixing
#define MIX_S_UV mixing along s-levels
#define SPLINES_VVISC parabolic Splines for Vertical Viscosity

Tracers Equations Options:

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

Surface Forcing Options:

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

Vertical Turbulent Mixing Parameterization Options:

#define GLS_MIXING Generic Length Scale Mixing (K-omega)
 ifdef GLS_MIXING
 #define N2S2_HORAVG smoothing of buoyancy/shear
 #define KANTHA_CLAYSON stability function
 #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
(Basic Configuration)
Algorithm:
#define IS4DVAR primal form of incremental strong constraint 4D-Var
#undef BGQC Background quality control of observations

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
#undef BALANCE_OPERATOR Multivariate balance constraint
#undef BALANCE_OPERATOR
#define ZETA_ELLIPTIC SSH elliptic equation method
#undef ZETA_ELLIPTIC

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
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
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 == 25 ! 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: roms_wc13.in

ROMS/TOMS Standard Input parameters.

Copyright (c) 2002-2019 The ROMS/TOMS Group
Licensed under a MIT/X style license
See License_ROMS.txt

Input parameters can be entered in ANY order, provided that the parameter KEYWORD (usually, upper case) is typed correctly followed by "=" or "==" symbols. Any comment lines are allowed and must begin with an exclamation mark (!) in column one. Comments may appear to the right of a parameter specification to improve documentation. Comments are ignored during reading. Blank lines are also allowed and ignored. Continuation lines in a parameter specification are allowed if preceded by a backslash (\). In some instances, more than one value is required for a parameter. If fewer values are provided, the last value is assigned for the entire parameter array. The multiplication symbol (*), without blank spaces in between, is allowed for a parameter specification. For example, in two grids nested application:

AKT_BAK == 2*1.0d-6  2*5.0d-6  ! m2/s

indicates that the first two entries of array AKT_BAK, in fortran column-major order, will have the same value of "1.0d-6" for grid 1, whereas the next two entries will have the same value of "5.0d-6" for grid 2.

In multiple levels of nesting or multiple connected domains step-ups.
Computing $\Lambda$

(Define NORMALIZATION)

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

(i) Exact method ($\text{Nmethod}=0$):

$C'e_i \rightarrow \text{ } i^{th} \text{ column of } C' \text{; save } c_{ii}$

where $e_i^T = (0,0,\ldots,0,1,0,\ldots,0)$

$\uparrow$

$i^{th} \text{ element}$

Requires $N_{grid}$ runs of diffusion operators: impractical for very large grids.
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
LwrtNRM == F F F F       ! Compute and write normalization

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 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 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>
</tr>
<tr>
<td>Vgamma</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0005</td>
</tr>
</tbody>
</table>

Initial conditions correlations (m):

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

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

Surface forcing correlations (m):

- **HdecayF**
  - **(isUstr)** == 100.0d+3 (surface U-momentum stress (66 convolutions))
  - **(isVstr)** == 100.0d+3 (surface V-momentum stress (66 convolutions))
  - **(isTsur)** == 100.0d+3 100.0d+3 (1:NT surface tracer flux (66 convolutions))
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(isFs ur)</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>Llobc(isFs ur)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>Llobc(isUbar)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>Llobc(isVbar)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>Llobc(isUvel)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>Llobc(isVvel)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>Llobc(isTvar)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Legend:
- **T**: keep edge unchanged
- **F**: force edge to zero
- **4**: free-surface
- **3**: 2D U-momentum
- **1**: 3D U-momentum
- **2**: 3D V-momentum
**4D-Var Parameters: Balance Operator**

**SSH, elliptic solver:**

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

**SSH, integration of hydrostatic equation:**

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

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

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

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

**State Variables switches:**

\[
\text{balance(isSalt)} = T \quad ! \text{salinity}
\]
\[
\text{balance(isFsur)} = T \quad ! \text{free-surface}
\]
\[
\text{balance(isVbar)} = F \quad ! 2D momentum (ubar, vbar)
\]
\[
\text{balance(isVvel)} = T \quad ! 3D momentum (u, v)
\]
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
I4D-Var Parameters File: i4dvar.in

! 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,
I4D-Var Job Script: job_i4dvar.sh

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 initial conditions, boundary conditions and surface forcing error covariance standard deviations files.
   ```bash
   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.
   ```bash
   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 I4DVAR = i4dvar.in
   if (-e $I4DVAR) then
     /bin/rm $I4DVAR
   endif
   cp s4dvar.in $I4DVAR
   ```

   ```bash
   $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
   ```
#!/bin/csh -f
#
# Set path definition to one directory up in the tree.
#
# Copyright (c) 2002-2019 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 covariance input standard deviations files.
# (3) Specify 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 "i4dvar.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.

   ```bash
   setenv MY_ROOT_DIR $(HOME)/ocean/toms/repository
   setenv MY_PROJECT_DIR $(PWD)
   ```

2. Location of your ROMS source code.

   ```bash
   setenv MY_ROMS_SRC $(MY_ROOT_DIR)/trunk
   ```

3. Build script invoked CPP options.

   ```bash
   setenv MY_CPP_FLAGS "$\{MY_CPP_FLAGS\} -D IS4DVAR"
   setenv MY_CPP_FLAGS "$\{MY_CPP_FLAGS\} -D ANA_SPONGE"
   # setenv MY_CPP_FLAGS "$\{MY_CPP_FLAGS\} -D BGQC"
   # setenv MY_CPP_FLAGS "$\{MY_CPP_FLAGS\} -D COLLECT_ALLREDUCE"
   # setenv MY_CPP_FLAGS "$\{MY_CPP_FLAGS\} -D REDUCE_ALLGATHER"
   # setenv MY_CPP_FLAGS "$\{MY_CPP_FLAGS\} -D DEBUGGING"
   # setenv MY_CPP_FLAGS "$\{MY_CPP_FLAGS\} -D POSITIVE_ZERO"
   ```

4. Path of Makefile configuration (*.mk) files

   ```bash
   setenv COMPILERS $(MY_ROMS_SRC)/Compilers
   setenv COMPILERS $(HOME)/Compilers/ROMS
   ```

5. Compiler selection environment variables.

   ```bash
   setenv USE_MPI on
   setenv USE_MPIF90 on
   setenv which_MPI openmpi
   setenv FORT ifort
   ```

6. Use custom library paths

   ```bash
   #setenv USE_MY_LIBS no # use system default library paths
   setenv USE_MY_LIBS yes # use my customized library paths
   
   set MY_PATHS = $(COMPILERS)/my_build_paths.sh
   if ($USE_MY_LIBS == 'yes') then
     source $(MY_PATHS) $(MY_PATHS)
   endif
   ```
#!/bin/csh -f
#
# svn $Id: build_roms.csh 977 2019-07-26 06:01:07Z arango $
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# Copyright (c) 2002-2019 The ROMS/TOMS Group :::::
# Licensed under a MIT/X style license :::::
# See License_ROMS.txt :::::
#::::::::::::::::::::::::::::::::::::::::::::::::::::: Hernan G. Arango :::::
# :::::
# ROMS/TOMS Compiling CSH Script :::::
# :::::
# Script to compile an user application where the application-specific :::::
# files are kept separate from the ROMS source code. :::::
# :::::
# Q: How/why does this script work? :::::
# :::::
# A: The ROMS makefile configures user-defined options with a set of :::::
# flags such as ROMS_APPLICATION. Browse the makefile to see these. :::::
# If an option in the makefile uses the syntax ?= in setting the :::::
# default, this means that make will check whether an environment :::::
# variable by that name is set in the shell that calls make. If so :::::
# the environment variable value overrides the default (and the :::::
# user need not maintain separate makefiles, or frequently edit :::::
# the makefile, to run separate applications).
# :::::
# Usage: :::::
# :::::
# ./build_roms.csh [options] :::::
# :::::
# Options:
Compiler Libraries: my_build_paths.csh

# svn $Id$
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# Copyright (c) 2002-2019 The ROMS/TOMS Group
#Licensed under a MIT/X style license
#See License_ROMS.txt
#::::::::::::::::::::::::::::::::::::::::::::::::::::: Hernan G. Arango:::
#
# ROMS/TOMS Customized Compiling Libraries Script
#
# This C-shell script sets the customized library paths needed by the build script when the enviromental variable USE_MY_LIBS has a 'yes' value.
#
# For example, in build_roms.csh we have:
#
#       if ($USE_MY_LIBS == 'yes') then
#          source ${COMPILERS}/my_build_paths.csh
#       endif
#
# set separator = `perl -e "print ':' x 100;"`

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

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