Using ROMS with DART

Release 0.0.1

Ben Johnson

Dec 07, 2022

CONTENTS

1 What you're about to do **3**

1.1 Compile DART executables in ROMS subdirectory 3
1.2 Compile COAWST on Cheyenne 5
1.3 General plan for adding and testing a model 7
1.4 Observing system simulation experiment / perfect_model_obs 8
1.5 Generate an ensemble of initial states 8
1.6 Convert your observations into DART's `obs_seq` file format 9
1.7 Compile and stage your experiment 10

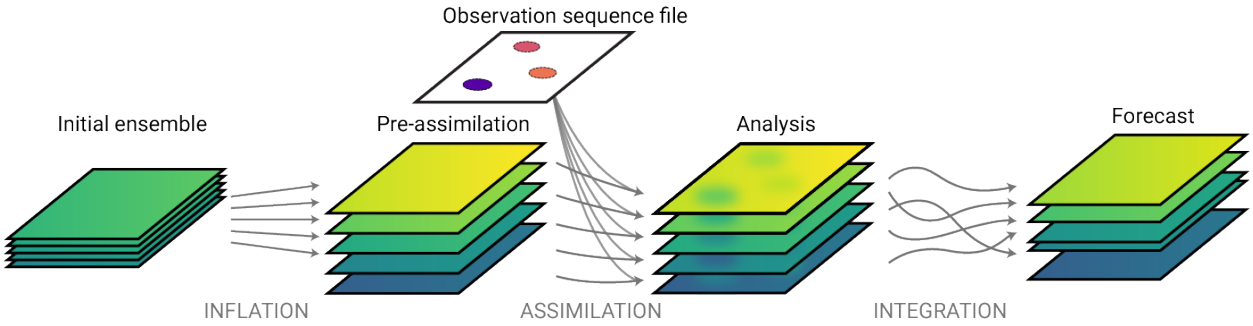
These pages document how to use DART's interface to the Regional Ocean Modeling System (ROMS).

CHAPTER

ONE

WHAT YOU'RE ABOUT TO DO

The majority of the infrastructure needed to complete an assimilation cycle is already implemented in the DART source code, so these instructions will show you how to work with the existing code to carry out your experiment. This diagram provides a summary of a single assimilation cycle:



In this diagram, the blue/green trapezoids represent an ensemble of model states. The observations to be assimilated are converted to DART's observation sequence, or `obs_seq`, file format.

The steps needed to complete an assimilation cycle are:

1. *Compile DART executables in ROMS subdirectory*.
2. *Generate an ensemble of initial states* using the `perturb_single_instance` executable.
3. *Convert your observations into DART's obs_seq file format* using a suitable observation converter.
4. *Compile and stage your experiment* using the build scripts in the ROMS subdirectory of the DART repository.

1.1 Compile DART executables in ROMS subdirectory

1.1.1 Setting your mkmf.template

If you haven't done so already, change directory to the `build_templates` subdirectory:

```
cd DART/build-templates
ls mkmf.*
mkmf.template.absoft.osx        mkmf.template.pgi.cray
mkmf.template.g95               mkmf.template.pgi.linux
mkmf.template.gfortran          mkmf.template.pgi.osx
mkmf.template.intel.linux       mkmf.template.rttov.gfortran
mkmf.template.intel.osx         mkmf.template.rttov.intel
mkmf.template.lahey.linux       mkmf.template.sgi.altix
```

(continues on next page)

(continued from previous page)

```
mkmf.template.nag.linux        mkmf.template.xlf.aix
mkmf.template.pathscale.linux
```

and identify which `mkmf.template` is suitable for your system. For example NCAR's Cheyenne supercomputer is built using Intel processors and uses the Linux operating system, so it should use the *mkmf.template.intel.linux* template.

```
mv mkmf.template.intel.linux mkmf.template
```

The `$NETCDF` environmental variable is already set on this system, so it doesn't need to be set within the `mkmf.template`:

```
echo $NETCDF
/glade/u/apps/ch/opt/netcdf/4.8.1/intel/19.1.1/
```

If the `$NETCDF` environmental variable isn't set on your system, identify the path to your netCDF installation and set it in your `mkmf.template`.

1.1.2 Compiling the ROMS executables

Next, change directory to the ROMS work subdirectory:

```
cd ../models/ROMS/work
ls
input.nml           obs_seq_fo.out  ocean.in.template  s4dvar.in.template
input.nml.template  obs_seq.out     quickbuild.sh
```

The `quickbuild.sh` script will compile all of the necessary DART executables for ROMS.

```
./quickbuild.sh
[ ... ]
ls
advance_time              Makefile           obs_seq_verify
closest_member_tool       model_mod_check    ocean.in.template
create_fixed_network_seq  obs_common_subset  perfect_model_obs
create_obs_sequence       obs_diag           perturb_single_instance
dart_log.nml              obs_selection      preprocess
dart_log.out              obs_seq_coverage   quickbuild.sh
fill_inflation_restart    obs_seq_fo.out     s4dvar.in.template
filter                    obs_seq.out        wakeup_filter
input.nml                 obs_seq_to_netcdf
input.nml.template        obs_sequence_tool
```

If `quickbuild.sh` ran properly, you should now see the work directory populated with newly built executables.

1.2 Compile COAWST on Cheyenne

These compilation instructions are adapted from the set up COAWST model document provided by Jose.

1.2.1 Download the source code from the GitHub repository

```
cd /glade/work/$USER/git
git clone https://github.com/jcwarner-usgs/COAWST.git
cd COAWST
COAWST_ROOT=$(pwd)
git checkout -b dart_test
Switched to a new branch 'dart_test'
```

1.2.2 Build the MCT libraries

Use configure to create a makefile for MCT:

```
module list
Currently Loaded Modules:
1) ncarenv/1.3    3) ncarcompilers/0.5.0   5) netcdf/4.8.1   7) nco/5.0.3
2) intel/19.1.1   4) mpt/2.25              6) ncl/6.6.2
cd $COAWST_ROOT/Lib/MCT
./configure
[ ...  ]
config.status: creating Makefile.conf
Please check the Makefile.conf
Have a nice day!
make
```

Copy the compiled mpeu library into the mct directory:

```
cp mpeu/libmpeu.a mct/
```

1.2.3 Edit files in the compiler directory

```
cd $COAWST_ROOT/Compilers
vim Linux-ifort.mk

227    MCT_INCDIR ?= /glade/work/$USER/git/COAWST/Lib/MCT/mct
228    MCT_LIBDIR ?= /glade/work/$USER/git/COAWST/Lib/MCT/mct
```

1.2.4 Copy the build script into the project directory

For this example, we will be using the `Inlet test` project included in COAWST, since its `CPPDEFS` are similar to those used for Jose's Sanibel Island domain.

```
cd $COAWST_ROOT/Projects/Inlet_test
cp ../../coawst.bash ./
```

Edit the setup script to declare the location of the COAWST installation

```
vim coawst.bash
132 export   MY_ROOT_DIR=/glade/work/johnsonb/git/COAWST
```

1.2.5 Edit the project's header script to output verification observations

Note: ROMS should be configured to output a data assimilation restart file, named the `ocean_dai.nc` file in order to provide verification observations for DART. DART refers to the ROMS verification observations as "precomputed forward operators." For more information on this capability, read the ROMS upgrade ticket description for 4D-Var or EnKF initial/restart file, ROMS depths, Observations.

In order to configure ROMS to output the `ocean_dai.nc` file, the CPP-preprocessing `ENKF_RESTART` option must be set in the project's header file.

```
vim Coupled/inlet_test.h
#define ENKF_RESTART
```

1.2.6 Run the setup script

```
chmod 777 coawst.bash
qsub -X -I -l select=1:ncpus=36:mpiprocs=36 -l walltime=01:00:00 -q regular -A $DARES_
↪PROJECT
./coawst.bash -j 36
```

1.2.7 Edit the configuration files

The `coupling_<COAWST project name>.in` file, in this case the `coupling_inlet_test.in` file, should set the number of processors allocated to each of the component models in the experiment.

Important: Using the term "Nodes" here is a misnomer. In actuality, the user is setting the number of processors for each component model.

This experiment is being run on a single node of Cheyenne, which has 36 processors. There are two component models, ROMS and SWAN, so `NnodesWAV` and `NnodesOCN` should be set in a manner that adds up to 36.

```
vim Coupled/coupling_inlet_test.in
41   NnodesWAV =  11                    ! wave model
42   NnodesOCN =  25                    ! ocean model
[ ... ]
61    WAV_name = /glade/work/$USER/git/COAWST/Projects/Inlet_test/Coupled/swan_inlet_
↪test.in      ! wave model
62    OCN_name = /glade/work/$USER/git/COAWST/Projects/Inlet_test/Coupled/ocean_inlet_
↪test.in     ! ocean model
```

Since 25 processors were set for `NnodesOCN` in the `coupling_inlet_test.in` file, the product of `NtileI` and `NtileJ` in the `ocean_<COAWST project name.>.in` file, in this case the `ocean_inlet_test.in` file, should equal 25.

```
vim Coupled/ocean_inlet_test.in
111    NtileI == 5                                ! I-direction partition
112    NtileJ == 5                                ! J-direction partition
```

1.2.8 Running the executable

Now that those configuration files are set, the executable can be run.

```
cd $COAWST_ROOT
mpirun -np 36 ./coawstM Projects/Inlet_test/Coupled/coupling_inlet_test.in
[ ... ]
------------------------------------------------------------------------------
Model Input Parameters:  ROMS/TOMS version 3.9
                         Wednesday - December 7, 2022 -  1:37:22 PM
------------------------------------------------------------------------------
[ ... ]
ROMS/TOMS: DONE... Wednesday - December 7, 2022 -  1:40:34 PM
ls -lart
[ ... ]
ocean_dai.nc
[ ... ]
```

COAWST runs properly and outputs an `ocean_dai.nc` restart file.

1.3 General plan for adding and testing a model

1.3.1 Overview

This project uses the Coupled-Ocean-Atmosphere-Wave-Sediment Transport (COAWST) modeling system. COAWST includes many model components and uses the Model Coupling Toolkit (MCT) to couple the components together. The primary model components are:

- the Weather Research and Forecasting (WRF) model for the atmosphere
- the Regional Ocean Modeling System (ROMS) model for the ocean
- the Simulating WAves Nearshore (SWAN) model for ocean surface waves

As a starting point, we will focus on the ROMS component of COAWST. The Ocean and Atmosphere Studies Laboratory provides a user guide for COAWST in which the ROMS section begins on page 53.

1.3.2 General steps

First stage: ensure the basic functioning of the system using synthetic observations.

1. Compile a single instance of the model that can be used in an observing system simulation experiment (OSSE). This step is covered in depth in the *Observing system simulation experiment / perfect_model_obs* page.
2. Generate an ensemble of model states. This step is covered in the *Generate an ensemble of initial states* page.
3. Use the synthetic observations from the OSSE to run a test assimilation.

Second stage: attemp to assimilate real observations.

1. Find a suitable observation converter in the DART repository, as described in the *Convert your observations into DART's obs_seq file format* page, or
2. Write an observation converter for your observations.

1.4 Observing system simulation experiment / perfect_model_obs

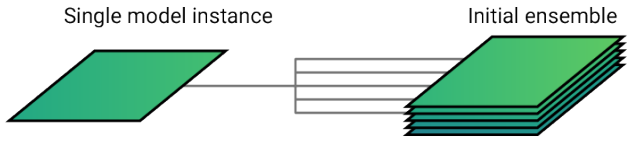
The DART executable `perfect_model_obs` takes restart files from a single instance of a model run and creates observation sequence files from it containing synthetic "observations" from the model run.

The most straightforward way to use `perfect_model_obs` is to use two precursor programs to format a file that can be fed to `perfect_model_obs`:

1. `create_obs_sequence` creates a template for the observations
2. `create_fixed_network_sequence` repeats that template at multiple times
3. `perfect_model_obs` harvests the observation values.

1.5 Generate an ensemble of initial states

The DART executable `perturb_single_instance` takes a single ROMS restart file as input and will generate additional copies of the restart file with small perturbations to the first file, in order to generate an ensemble of initial model states.



You should always use a restart file from your experimental domain. But for demonstration purposes, DART profiles an archive that contains several restart files. To download the demo archive:

```
cd DART/models/ROMS/work
wget https://www.image.ucar.edu/pub/DART/ROMS/dart_roms_test_data.tar.gz
tar -xzvf dart_roms_test_data.tar.gz
```

This will extract a directory named `wc12` that contains `.in` files and netCDF files from the West Coast domain described in Moore et al. (2020)[1].

```
ls wc12
ocean.in                      varinfo.dat
roms_posterior_0001_37700.nc wc12_avg.nc
roms_posterior_0002_37700.nc wc12_dia.nc
roms_posterior_0003_37700.nc wc12_his.nc
roms_posterior_0004_37700.nc wc12_ini_0001.nc
s4dvar.in
```

The `perturb_single_instance` executable expects the initial input file to be named `roms_input.nc` so use a symbolic link to associate one of the netCDF restart files in wc12 with that name.

```
ln -s wc12/roms_posterior_0001_37700.nc roms_input.nc
```

Next, add a namelist in the `input.nml` file named `perturb_single_instance_nml` to instruct the executable how many perturbed copies it should create. Open `input.nml` with a text editor and add a namlist similar to:

```
&perturb_single_instance_nml
   ens_size               = 3
   input_files            = 'roms_input.nc'
   output_files           = 'wc12/roms_posterior_0005_37700.nc','wc12/roms_posterior_
↪0006_37700.nc','wc12/roms_posterior_0007_37700.nc'
   output_file_list       = ''
   perturbation_amplitude = 0.2
 /
```

Save the namelist and run the executable:

```
./perturb_single_instance
```

You should see as many perturbed files as you specified in the namelist:

```
ls -art wc12
[ ...  ]
roms_posterior_0005_37700.nc
roms_posterior_0006_37700.nc
roms_posterior_0007_37700.nc
```

1.5.1 References

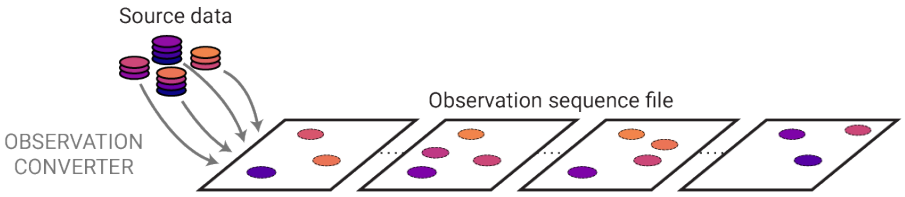
1.6 Convert your observations into DART's obs_seq file format

DART has a proprietary file format used to contain observations called an `obs_seq` or observation sequence file.

This file format is necessary because ensemble data assimilation techniques require that each observation has a specified observation error variance in order to apply Bayes' Theorem during the assimilation.

DART provides source code to compile a type of executable known as an observation converter that takes source data and creates a suitable `obs_seq` file for assimilation.

[1] Moore, A., J. Zavala-Garay, H. G. Arango, C. A. Edwards, J. Anderson, and T. Hoar, 2020: Regional and basin scale applications of ensemble adjustment Kalman filter and 4D-Var ocean data assimilation systems. *Progress in Oceanography*, **189**, 102450, https://doi.org/10.1016/j.pocean.2020.102450.



It is possible that there is already source code in the repository to compile an observation converter for the data you intend to assimilate.

Examine the `obs_converters` directory to see if there is already a converter for your source data:

```
cd DART/observations/obs_converters
ls
AIRS           GSI2DART         SIF
Ameriflux      GTSPP            snow
AURA           MADIS            SSEC
AVISO          MIDAS            SST
CHAMP          MODIS            SSUSI
cice           MPD              tec
CNOFS          NASA_Earthdata   test_obs
CONAGUA        NCEP             text
COSMOS         NSIDC            text_GITM
DWL            obs_error        tpw
even_sphere    ocean_color      tropical_cyclone
GMI            ok_mesonet       USGS
gnd_gps_vtec   quikscat         utilities
GOES           radar            var
gps            README.rst       WOD
GPSPW          ROMS
GRACE          SABER
```

Comprehensive descriptions are also available in the Available observation converter programs page in the DART documentation.

1.7 Compile and stage your experiment

After generating your initial ensemble and staging `obs_seq` files for assimilation, you're ready to compile and stage your experiment.

The `shell_scripts` subdirectory in the ROMS model directory contains scripts that you can adapt in order to stage your experiment.

```
cd DART/models/ROMS/shell_scripts
ls
advance_ensemble.csh.template
batch_job_resource_explanation.txt
cycle.csh.template
ensemble.sh
get_ocean_time.csh
run_filter.csh.template
stage_experiment.csh
submit_multiple_cycles_lsf.csh
submit_multiple_jobs_slurm.csh
```

The `stage_experiment.csh` shell script is the primary script you will be adapting to build your experiment.

Ensure to add a new `hostname` case at the top of the script corresponding to the hostname of your computing cluster. Add directory paths corresponding to the appropriate paths on your machine.

There are scripts for submitting jobs on both `LSF` and `SLURM` job management systems. If your cluster uses `PBS` or some alternative system, you will need to adapt one of those scripts for your system.

For more detailed information, read the Shell scripts section of DART's ROMS interface documentation.