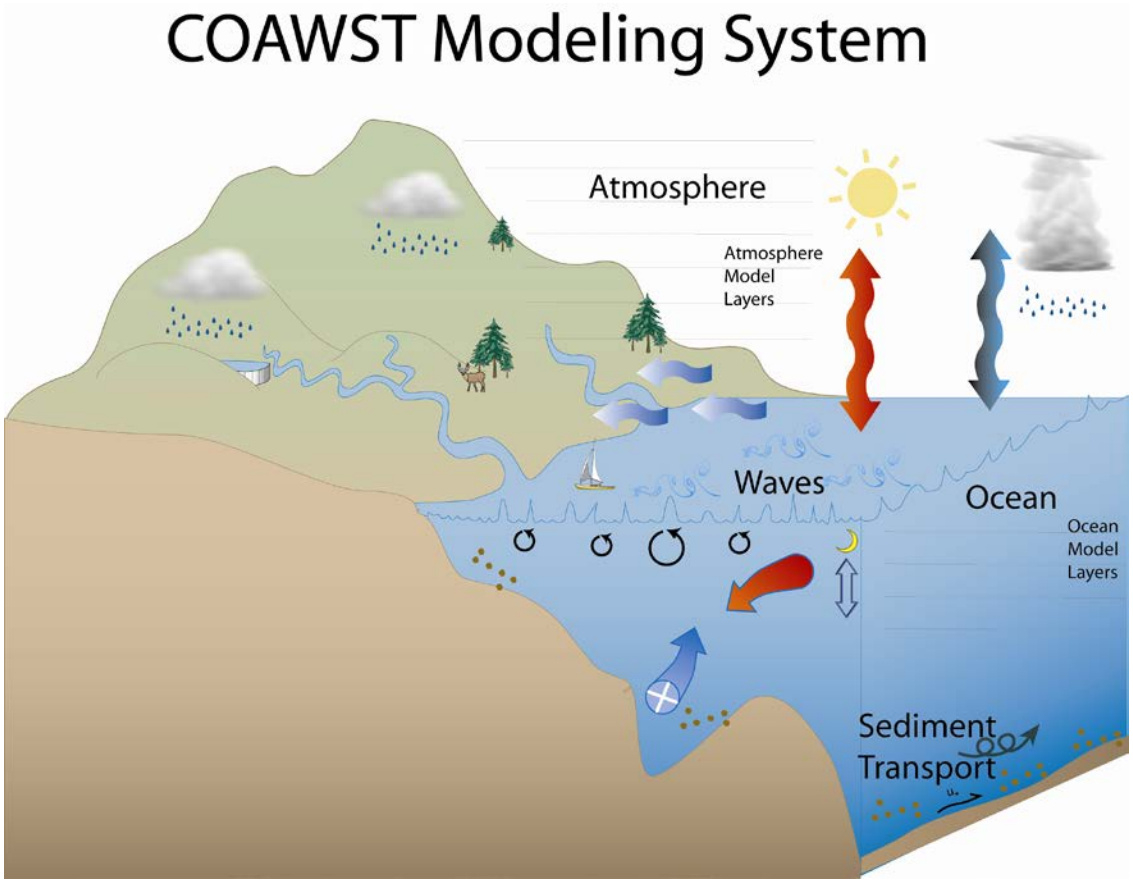
COAWST

User's Manual

Version 3.6



http://woodshole.er.usgs.gov/operations/modeling/COAWST/index.html

April 23, 2020

"Although this program has been used by the USGS, no warranty, expressed or implied, is made by the USGS or the United States Government as to the accuracy and functioning of the program and related program material nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the USGS in connection therewith."

Table of Contents.

COAWST User's Manual

1. Introduction

The **C**oupled-**O**cean-**A**tmosphere-**W**ave-**S**ediment **T**ransport (**COAWST**) Modeling System is an agglomeration of open-source modeling components that has been tailored to investigate coupled processes of the atmosphere, ocean, and waves in the coastal ocean. The modeling system currently contains:

Coupler:	- Model Coupling Toolkit (MCT) v 2.6.0
Ocean:	- Regional Ocean Modeling System (ROMS) 3.8 svn 1013
Atmosphere:	- Weather Research and Forecasting Model (WRF) v 4.1.2
Wave(s):	- Simulating Waves Nearshore (SWAN) v 41.31
	- WAVEWATCH III (WW3) v 6.07.1+
	- Infragravity wave model (InWave) v 1.5
Sediment Transport:	- Community Sediment Transport Modeling Systems (CSTMS)
Sea Ice:	- Sea Ice model

Here are some model specific user forums that can provide useful information:

MCT
http://www.mcs.anl.gov/research/projects/mct/

ROMS
https://www.myroms.org/forum/index.php

WRF
http://forum.wrfforum.com/

SWAN
http://swanmodel.sourceforge.net/

WAVEWATCH III
http://polar.ncep.noaa.gov/waves/wavewatch/

The main reference for the COAWST modeling system is:
Warner, J.C., Armstrong, B., He, R., and Zambon, J.B., 2010, Development of a Coupled Ocean-Atmosphere-Wave-Sediment Transport (COAWST) modeling system: Ocean Modeling, v. 35, no. 3, p. 230-244.

The main website is:
http://woodshole.er.usgs.gov/operations/modeling/COAWST/index.html

For bug reports and Discussion, we are currently using the Trac site:
https://coawstmodel-trac.sourcerepo.com/coawstmodel_COAWST/
Please log onto that site to post questions or submit a bug report.

2. Accessing COAWST

The code is maintained in a svn repository and is distributed thru subversion checkout. Please contact John Warner for code access at jcwarner@usgs.gov You will then receive an email with your username (myusrname) and password. To obtain the source code via the svn repository, I suggest you make a directory called COAWST, cd to that dir, and use the following command (all one line):

svn checkout --username myusrname
https://coawstmodel.sourcerepo.com/coawstmodel/COAWST .

Notice the "." at the end of the command to place the code in that location. Alternatively instead of the "." you can add any path of where you want it to go.

3. Required libraries for model installation

The following libraries are required to run the coupled modeling system. These are:
- **Fortran Compiler**
- **NetCDF**
- **MPI**
- **MCT**
- **SCRIP_COAWST**

These libraries/programs only need to be installed once, typically by an administrator, and set to be available for all users.

- **Fortran Compiler**: The system has been tested with several different fortran compilers, including ifort, pgi, and gfortran. Each compiler is different and the same compiler will have multiple versions. We cannot test them all, but have tried many configurations. You need to use the same fortran compiler to install/compile NetCDF, MPI, MCT, SCRIP_COAWST, and then to compile COAWST itself. Some compilers are free, and some have a cost.

- **NetCDF** is a common data format that can be freely acquired from http://www.unidata.ucar.edu/software/netcdf/ The models WRF, ROMS, and SWAN can each individually use NetCDF v3.x with the fortran interface. However, as of COAWST release v3.2, we require NetCDF v4.x for the SCRIP_COAWST interpolation weights package. We therefore require NetCDF v4.x for full model support. You need to set your environment variables so the system knows where to locate the netcdf files. The way to set environment variables is different for each system, so you may need to use "setenv" or "export" or some other command, but an example is listed here:

```
setenv  NETCDF_INCDIR     /PATH_TO_YOUR_NETCDF/include
setenv  NETCDF_LIBDIR     /PATH_TO_YOUR_NETCDF/lib
setenv  NETCDF            /PATH_TO_YOUR_NETCDF/
setenv  NETCDF_CONFIG     /PATH_TO_YOUR_NETCDF/nc-config
```

The settings of NETCDF_INCDIR and NETCDF_LIBDIR are for ROMS. The setting of NETCDF is for WRF. The setting of NETCDF_CONFIG is for WAVEWATCH III.

多分 Intel compiler module を load し、OpenMPI と conflict を起こしたため、
ITO-A では OpenMPI での実行時のエラーが解決できなかった。
configure.wrf を mpiifort, mpiicc に書き換えるだけで成功した。
WSL-Ubuntu では OpenMPI をビルドして使用する

- **MPI** is a message passing interface software to allow the models to utilize multiple processors. There are many types and versions and we have tested mvapich, openmpi, and MPICH2. MPI is required to run any coupled (more than 1 model) simulation. Some of these are free.

- **MCT** is the Model Coupling Toolkit, and is freely available from http://www.mcs.anl.gov/research/projects/mct/. We recommend you use the version provided with this distribution in COAWST/Lib/MCT, as it has been slightly modified and tested to be compatible with COAWST. Slack 2020.6.26

OpenMPI のインストールが必要 (WSL-Ubuntu, ITO-A も)

コンパイルオプションに -I で openmpi の include file を指定する*

MCT installation

1) Copy the MCT package from COAWST/Lib/MCT and place into a folder that can be shared to all users. It is recommended to only install this once, and just let all users link to this for all their compilations.

2) ./configure

This will create a file Makfile.conf. You should edit this file and see if the paths are correct. Alos, make sure the compiler flags are the same as needed for your system. For example, if your flags need to use –assume-byterecl (ifort) or -frecord-marker=4 -fconvert=big-endian (gfortran) then add these compiler flags. I included the file Makefile.conf_jcw as an example of how my setup was on a Windows machine.

3) make

This will build the package in both the mct and mpeu directories. I modified the Makefiles in both of those directories so that they can also work in a Windows world. The Windows compilers typically build *.obj files, instead of just *.o files. So I modified the mct/Makefile and mpeu/Makefile for the $(F90RULECPP). You can edit those files and see the changes.

4) make install

This will place the MCT libraries and include files into the locations listed at the bottom of the Makefile.conf file. You can change these locations if you want to, but they typically are something like /usr/lib and /usr/include.

5) Set your environment variables so the system knows where to locate required files. The way to set environment variables is different for each system, so you may need to use “setenv” or “export” or some other command, but an example is listed here:

setenv MCT_INCDIR /usr/include
setenv MCT_LIBDIR /usr/lib [or whatever the actual paths are]

- **SCRIP_COAWST** 異なる格子の内挿

The spherical coordinate remapping interpolation package (SCRIP) is freely available from http://oceans11.lanl.gov/trac/SCRIP. This package is required when you have an application using more than one model, and the models are using different grids (different spatial extents and/or different number of mesh points). This SCRIP code will generate interpolation weights that are used to conservatively remap data fields between the model grids. This original SCRIP package has been modified for COAWST v3.2, and we now require users to compile SCRIP_COAWST as a Fortran executable. When you run the executable it will generate a single netcdf file the holds all the interpolation weights for

* ITO-A における MCT のインストールでは OpenMPI を用いず、Slack の linux-build_tips の 2020年11月7日の記事の通り、export FC=mpiifort export cc=icc 等とすれば、Intel MPI でコンパイルすることができた。

any number of grids. To work with SCRIP_COAWST package, the user needs NetCDF.4.x to have the netcdf group feature available.

SCRIP_COAWST installation

1) Copy the COAWST/Lib/SCRIP_COAWST package and place into a folder that can be shared to all users. It is recommended to only install this once, and just let all users link to this for all their compilations.
2) Edit the makefile to enter the fortran version that is installed, for example, set FORT = pgi (or whatever compiler you use)
3) type "make" at the command prompt.
4) This should compile to create an executable called scrip_coawst[.exe]

Section 4. Setting up and running applications

The COAWST system is designed to allow the user to select any combination of the main models (WRF, ROMS, SWAN, or WAVEWATCH III). All of these main models compile in a different way, but we need to have one consistent approach to compile the complete coupled system. This procedure is based on the method to build the ROMS code, with additional options for each model and the coupled system. The user needs to list:

- cpp options to select the model(s) (Section **4.1**)
- cpp options for each model (Section **4.2**)
- cpp options for the coupled system (Section **4.3**)

This is accomplished by listing the options in a header control file, let's call it project.h. This project.h file will list cpp (c pre-processor) choices that determine which options in the code are compiled. These options are listed below and some are described further in the test examples section.

4.1 CPP options to select the model(s)

These are **REQUIRED** cpp option(s). This allows the user to activate a single model or multiple models. Specify which model(s) you want to use:

#define ROMS_MODEL /* if you want to use the ROMS model */
#define SWAN_MODEL /* if you also want to use the SWAN model */
#define WRF_MODEL /* if you also want to use the WRF model */
#define WW3_MODEL /* if you also want to use the WW3 model */

As of COAWST v3.2, the system can run any set of model choices:
WRF only, ROMS only, SWAN only, WW3 only,
WRF+ROMS, WRF+SWAN, ROMS+SWAN, WRF+ROMS+SWAN
WRF+WW3, ROMS+WW3, WRF+ROMS+WW3
We can also run:

- SWAN only with or without grid refinement,
- ROMS only with or without grid refinement,
- WRF only with or without grid refinement (static + moving nest)
- ROMS+SWAN+WRF with grid refinement in all 3 (WRF with static or moving nest). To run with a WRF moving nest, that moving nest needs to be the last child

grid of the WRF system. We currently only support coupling to 1 moving WRF nest.

- ROMS+WW3+WRF: For WW3, we currnently only allow one wave grid. However that WW3 grid can be a different size than any other grids. Also ROMS and WRF can have multiple grids, but only 1 WAVCEWATCHIII grid (for now). You then need to use SCRIP to compute interpolation weights (just as you would for SWAN or any other model).

4.2 CPP options for each model

Each model may have individual options that can be set for compilation. These are described below.

- For WRF: there are no additional cpp options that need to be listed in the project.h file at this time. Most options are set in the namelist.input and determined during the build.
- For WAVEWATCH III: there are no additional cpp options that need to be listed in the project.h file at this time. Most options are set in the WW3 switch file for the build.
- For SWAN: there are no additional options needed to be listed in the project.h file at this time. Most options are set in the INPUT file. However, for completeness, we already preset several options in SWAN and used:

perl switch.pl -unix -impi -mpi -f95 -netcdf *.ftn *.ftn90

This activated MPI routines, unix environments, and netcdf functionality. Users do not need to list anything at this time.

- For ROMS: The user need to become familiar with the vast number of ROMS cpp options that are available. Please look at the ROMS/Inlcude/cppdefs.h and the ROMS wiki pages https://www.myroms.org/wiki/Documentation_Portal that has a lot of documentation for ROMS cpp options. The user needs to select the ROMS cpp options and list them in the project.h file.

4.3 CPP options for the coupled system.

Below are cpp options specific to COAWST.

1つのモデルのみの場合は不要？

4.3.1) To activate model coupling:

#**define MCT_LIB** /* if you have more than one model selected and you want to couple them.*/

The following cpp options are activated internally. The user should NOT list these in their project.h file.

ROMS_COUPLING – roms is being used and is coupled to another model.
SWAN_COUPLING – swan is being used and is coupled to another model.
WW3_COUPLING – wavewatch3 is being used and is coupled to another model.
WRF_COUPLING – wrf is being used and is coupled to another model.
AIR_OCEAN – wrf and roms are active (other models could also be active).
WAVES_OCEAN – swan-or-wavewatch3 and roms are active (other models could also be active).
AIR_WAVES – swan-or-wavewatch3 and wrf are active (other models could also be active).

4.3.2) Some wave-current cpp options that are available for the coupling include:

#**define UV_KIRBY** /* compute "surface-average" current based on Hwave that will be sent from the ocn to the wav model for coupling*/

#**define UV_CONST** /* send vel = 0 from the ocn to wave model */

#**define ZETA_CONST** /* send zeta = 0 from the ocn to wave model */

#**define SST_CONST** /* do not send sst data from roms to wrf */

4.3.3) Atmosphere coupling cpp option:

#define ATM2OCN_FLUXES

This option specifies that the heat and momentum fluxes as computed by the atmosphere model will be used in the ocean model. This will allow both models to be using the identically same fluxes at the interface. When using this option, you should also use

#undef BULK_FLUXES

because the fluxes will be computed by wrf depending on the surface scheme you select.

4.3.4) Methods for grid interpolation.

#define MCT_INTERP_WV2AT /* this allows grid interpolation between the wave and atmosphere models */

#define MCT_INTERP_OC2AT /* this allows grid interpolation between the ocean and atmosphere models */

#define MCT_INTERP_OC2WV /* this allows grid interpolation between the ocean and wave models */

- If you use different grids for the ocean, atmosphere, or wave models, then you need to activate the appropriate option above so that the data fields are interpolated between grids.
- If you have a coupled application with NESTING in ROMS or SWAN or refinement in WRF, then you need to use the appropriate interpolation flag because all the grids talk to all the other grids (for example all WRF grids would communicate to all the ocean grids).
- If you need any of these _INTERP_ cpp options, then you need to create interpolation weights using SCRIP_COAWST. This is described in Section 5 below.

4.3.5) Methods for grid refinement.

#define NESTING /* this activates grid refinement in both roms and in swan.*/

- ROMS has one-way, two-way, and composed grids nesting.
- SWAN has one-way grid refinement.
- WRF has two way nesting and composed grids.
- As of COAWST v3.2, if you are running a simulation with nesting, the models can all have different number of grids. The test case INLET_TEST_REFINED (explained below) is an example where both roms and swan are coupled, with grid refinement, and coupling on the refined grids. The test case SANDY is an example with 2 grids for roms, 2 for swan, and 2 for wrf (2 static – or -- 1 static + 1 moving).
- If the grids for ROMS, SWAN, and WRF are different sizes, then the user needs to provide interpolation weights using SCRIP_COAWST for those grid combinations. The number of grids for ROMS is set in the ocean.in file, the number of grids for SWAN set in the swan.in file, and for WRF set in the namelist.input file.

If your simulations have more than 1 model, and at least one of those models has multiple grids, then you need to create the interpolation weights file using the SCRIP_COAWST library. This creates the netcdf weight files that need to be listed in the coupling.in file.

4.3.6) SWAN –or- WAVEWATCH III wave interactions to ROMS or to WRF:
The following 3 options are available to allow exchange of wave data to ROMS for use in bulk_fluxes for computation of ocean surface stress, and to allow exchange of wave data to WRF for use in MYJSFC and MYNN surface layer schemes to allow increased bottom roughness of the atmosphere over the ocean:

#define COARE_TAYLOR_YELLAND
Taylor, P. K., and M. A. Yelland, 2001: The dependence of sea surface roughness on the height and steepness of the waves. J. Phys. Oceanogr., 31, 572-590.

#define COARE_OOST
Oost, W. A., G. J. Komen, C. M. J. Jacobs, and C. van Oort, 2002:New evidence for a relation between wind stress and wave age from measurements during ASGAMAGE. Bound.-Layer Meteor., 103, 409-438.

#define DRENNAN
Drennan, W.M., P.K. Taylor and M.J. Yelland, 2005: Parameterizing the sea surface roughness. J. Phys. Oceanogr. 35, 835-848.

4.3.7) Implementation of wave-current interaction formulation.
We added a new method based on the vortex force approach. The method is described in detail Kumar et al (2012). The new cpp options for this are:

#define WEC_MELLOR radiation stress terms from Mellor 08
#define WEC_VF wave-current stresses from Uchiyama et al.
#define WDISS_THORGUZA wave dissipation from Thorton/Guza
#define WDISS_CHURTHOR wave dissipation from Church/Thorton
#define WDISS_WAVEMOD wave dissipation from a wave model
#define WDISS_INWAVE wave dissipation from a InWave model
#define ROLLER_SVENDSEN wave roller based on Svendsen
#define ROLLER_MONO wave roller for monchromatic waves
#define ROLLER_RENIERS wave roller based on Reniers
#define BOTTOM_STREAMING wave enhanced bottom streaming
#define SURFACE_STREAMING wave enhanced surface streaming

Interested users should read the Kumar et al. (2012) paper.

4.3.8) Drag limiter option.
#define DRAGLIM_DAVIS is a feature added to WRF and SWAN to limit the ocean roughness drag to be a maximum of 2.85E-3, as detailed in:

Davis et al, Prediction of Landfall Hurricanes with the Advanced Hurricane WRF Model, Monthly Weather Review, 136, pp 1990-2005.

In SWAN, this can be activated when using the Komen wind input. For WRF, it can be activated when using myjsfc or mynn surface layer options.

4.3.9) Vegetation options.
Vegetation module was added to get the 3-D effect of vegetation on wave and current fields. Details of the implementation are in this paper:
Beudin, A., Kalra, T. S., Ganju, N. K., and Warner, J.C. (2016). Development of a coupled wave-flow-vegetation interaction module. Computers and Geosciences, http://dx.doi.org/10.1016/j.cageo.2016.12.010.

The following flags can be used to work with the vegetation module.

# define VEGETATION	Switch on vegetation module
# define VEG_DRAG	Drag terms Luhar M. et.al (2011)
# define VEG_FLEX	Flexible vegetation terms
# define VEG_TURB	Turbulence terms, Uittenbogaard R. (2003)
# define VEG_SWAN_COUPLING	Exchange of VEG data btwn. ROMS and SWAN
# define VEG_STREAMING	Wave streaming effects

Within the vegetation module, we can use two processes to account for marsh dynamics: (a) Lateral wave thrust on marsh edge and (b) vertical accretion due to biomass production and change in marsh vegetation properties.

# define MARSH_DYNAMICS	Main kernel to call marsh routines
# define MARSH_WAVE_THRUST	Lateral wave thrust calculation
# define MARSH_SED_EROSION	Calculate sediment release due to wave thrust
# define MARSH_RETREAT	Lateral retreat calculation
# define MARSH_TIDAL_RANGE	Calculates mean high high water and tidal range
# define MARSH_VERT_GROWTH	Vertical accretion due to biomass production
# define MARSH_BIOMASS_VEG	Updates the stem density from marsh biomass

4.3.10) Estuarine biogeochemistry flags
This routine is a modification of Fennel et al. (2006) ecosystem model to include the Submerged Aquatic Vegetation (SAV) growth model that allows for the fully coupled water column biogeochemistry-sediment-hydrodynamics model.

# define ESTUARYBGC	Modification of Fennel module
# define SPECTRAL_LIGHT	Spectral light model
# SAV_BIOMASS	SAV biomass growth model
# SAV_BIOMASS_DAILY_UPDATE	Only update SAV properties daily

4.3.11) InWave options.
The following cpp options are available to run the InWave model:

# **define INWAVE_MODEL**	activate InWave model

define INWAVE_SWAN_COUPLING activate reading of a SWAN 2D spec file
define DOPPLER use to turn ON or OFF the effect of currents on the dispersion relation
define ACX_ADVECTION use to turn ON or OFF advection of Ac in the xi direction
define ACY_ADVECTION use to turn ON or OFF advection of Ac in the etai direction
define ACT_ADVECTION use to turn ON or OFF advection of Ac in the directional direction
define WDISS_ROELVINK use to turn ON or OFF Roelvink energy dissipation
define WDISS_GAMMA use to turn ON or OFF gamma based energy dissipation

The InWave model is described more in Section 11. For now you can run InWave forced with a SWAN 2d spec file, or you can impose the wave action density along an open boundary. InWave runs coupled to ROMS for wave-current feedbacks. It can also be run with the WEC options, wetting drying, sediment, and morphology.

Section 5. Compiling and Running

This section describes how to:
5.1- How to Compile and Run COAWST
5.2- How to Compile and Run SCRIP_COAWST
5.3- Some general helpful tips and a few new functionalities.

5.1 To Compile and Run COAWST

First, make sure that you read Section 3 and you have selected a Fortran Compiler and then used that compiler to build and install all the required libraries of: NetCDF, MPI, and MCT. Additionally, if your application uses models on different grids and you need to create interpolation weights, then you also need to compile SCRIP_COAWST and run that program to create weights. Section 5.2 describes how to run that program to create the interpolation weights.

To compile COAWST, you need to use the *coawst.bash* file. This file is at the COAWST root level. You need to edit the *coawst.bash* to set the options specific to your system and to choose the location of the projects.h file (described in Section 4). Some important parts are:

coawst.bash

```
export COAWST_APPLICATION=JOE_TC
```

Use capital letters to list the application name (in this example the application name is JOE_TC). This needs to be the same name as your project.h file that contains all your cpp options (See Section 4). The project.h file itself can be named in lower case (for example, the COAWST_APPLICATION is JOE_TC but the file name is joe_tc.h).

```
export    MY_ROOT_DIR=/raid1/jcwarner/Models/COAWST
export    MY_PROJECT_DIR=${MY_ROOT_DIR}
```

The rootdir is the location of the source code. You can have a project dir that you work in. I typically make a new copy of the code each time in case I change something and this means I have multiple rootdirs. Then I can always go back and see exactly the version of code that was used for a particular run. Therefore I set the project dir as the same location as the rootdir.

export MY_ROMS_SRC=${MY_ROOT_DIR}/
Keep this the same as listed here.

For WAVEWATCH III we now have 5 environment variables:

export COAWST_WW3_DIR=${MY_ROOT_DIR}/WW3
The COAWST_WW3_DIR is a pointer to root WW3 code, do not change.

export WWATCH3_NETCDF=NC4
This flag tells WAVEWATCH III to use netcdf 4. do not change.

export WWATCH_ENV=${COAWST_WW3_DIR}/wwatch.env
This WWATCH_ENV points to WW3 environment listing. do not change.

export NETCDF_CONFIG=/share/apps/netcdf-4.1.3_intel-2011.4.191/bin/nc-config
(or wherever your nc-config is.). This is needed to be set by you.

export WW3_SWITCH_FILE=sandy_coupled
This switch file is used by WAVEWATCHIII. It is like the roms project.h file. You need to create and set the name of the file (see Section 11 for more information).

There are several compiler selections:
export USE_MPI=on
export USE_MPIF90=on
The COAWST system was designed to work with MPI for parallel processor applications. If you set the modeling system to build a coupled application, then it will always produce an executable "coawstM". If you set to build an individual model and also set MPI=on, then you will get a "coawstM."

export FORT=pgi
You can set this to be ifort, pgi, or gfortran. Other choices may work.

Header and other source directories:
export MY_HEADER_DIR=${MY_PROJECT_DIR}/Projects/JOE_TC/Coupled
export MY_ANALYTICAL_DIR=${MY_PROJECT_DIR}/Projects/JOE_TC/Coupled
Use these to set the locations for you application. Header dir is where your project.h file is located.
End of user defined coawst.bash features.

After you edit the coawst.bash file you need to run it. To (re)build everything you use:

./coawst.bash –j X
(the –j says to use 'X' number of processors, uise a number for X, like 2 or 4).

This command will just rebuild any changes to roms or swan, and will rebuild all of wrf and all of ww3:
./coawst.bash –noclean

This command will rebuild all of roms and swan and just the changes that have been made to wrf or to ww3:
./coawst.bash –nocleanwrf –nocleanww3

This will rebuild roms, rebuild swan, rebuild ww3, and rebuild only the wrf files that have been changed.
./coawst.bash –nocleanwrf

WRFのコンパイルをより細かく行う方法
If you need to make modifications to the WRF configuration file, here are some tips. you could do this:
cd WRF
./clean -a
./configure
編集する
then select the options that you want. It will create the configure.wrf file and you can edit that file. then cd .. (to the root dir) and use
./coawst.bash -nocleanwrf
This way it will not re-create the configure.wrf file and it will build all of the system.

Another handy tool is the script command
http://www-users.cs.umn.edu/~skim/cs1901/script.html
You could use:
script coawst.log
./coawst.bash
exit
and then save that coawst.log file to see how the system was built.

The system may take a while to build. If it was successful, then you will get a coawstM or coawstM.exe file. If you get that file then the build was successful. In some instances you may see an error or warning "undefined wrf.main." That is the only error that is ok. We can not allow a wrf.exe to be built, nor an ocean.exe, and/or swan.exe to be created. Only one executable is created, and that is called coawstM[.exe].

To Run COAWST, you need to use a shell script that is dependent on your system. One example of a job script used on a PBS system is the file run_nemo:

```
#!/bin/bash
### Job name
#PBS -N cwstv3
### Number of nodes
#PBS -l nodes=2:ppn=8
### Mail to user
```

```
#PBS -m ae
#PBS -M jcwarner@usgs.gov
#PBS -q standard

umask 0002

echo "this job is running on:"
cat $PBS_NODEFILE

NPROCS=`wc -l < $PBS_NODEFILE`

cd /raid1/jcwarner/Models/COAWST_regress/COAWST_v3.3

mpirun -np 16 -machinefile $PBS_NODEFILE ./coawstM
Projects/JOE_TC/Couped/coupling_joe_tc.in > cwstv3.out
```

You need to talk to your system administrator to get a run script for your computer system. Not that the mpirun command calls the coawstM, and the input file is the coupling.in file for this application. This coupling.in file is described below.

To run/submit the the job, you would use the command
qsub run_nemo

To check job status
qstat –an
qstat –f

To kill the job use
qdel 'pid' where pid is the job number.

On a SLURM system, the commands are squeue, and sbatch. Please check with your system administrator for specific commands on your system.

Command Run File: coupling.in
To Run COAWST, your run script points to an input file. For example, in the run script above it pointed to coupling_joe_tc.in. The details of this file are:

Step1) Set the nodes to allocate for each model. This will depend on the application, number of processors you have access to, etc.

```
! Number of parallel nodes assigned to each model in the coupled system.
! Their sum must be equal to the total number of processors.
   NnodesATM =  1              ! atmospheric model
   NnodesWAV =  1              ! wave model
   NnodesOCN =  1              ! ocean model
```

Step 2) Set the coupling interval in seconds.

```
! Time interval (seconds) between coupling of models.
  TI_ATM2WAV =   600.0d0          ! atmosphere to wave coupling interval
```

```
  TI_ATM2OCN =   600.0d0           ! atmosphere to ocean coupling interval
  TI_WAV2ATM =   600.0d0           ! wave to atmosphere coupling interval
  TI_WAV2OCN =   600.0d0           ! wave to ocean coupling interval
  TI_OCN2WAV =   600.0d0           ! ocean to wave coupling interval
  TI_OCN2ATM =   600.0d0           ! ocean to atmosphere coupling interval
```

Step 3) Enter names of the input files for ROMS, WRF, and SWAN –or- WW3.

```
! Enter names of Atm, Wav, and Ocn input files.
! The Wav program needs multiple input files, one for each grid.
   ATM_name = namelist.input                                  ! atmospheric model
   WAV_name = Projects/JOE_TC/Coupled/INPUT_JOE_TC    ! wave model
   OCN_name = Projects/JOE_TC/Coupled/ocean_joe_tc.in       ! ocean model
```

Step 4) This is evolving, and we now suggest users to use the SCRIP_COAWST Fortran code to create a single weights file. We still allow the older approach of multiple files, but this will go away in future releases.

```
! Sparse matrix interpolation weights files. You have 2 options:
! Enter "1" for option 1, or "2" for option 2, and then list the
! weight file(s) for that option.

   SCRIP_WEIGHT_OPTION = 1
!
! Option 1: IF you set "SCRIP_WEIGHT_OPTION = 1", then enter name
!         of the single netcdf file containing all the exchange
!         weights. This file is created using the code in
!         Lib/SCRIP_COAWST/scrip_coawst[.exe]

   SCRIP_COAWST_NAME = Projects/JOE_TC/Coupled/scrip_weights_joe_tc.nc

! Option 2: THIS OPTION WILL BE REMOVED IN FUTURE VERSIONS.
!         IF you set "SCRIP_WEIGHT_OPTION = 2", then enter
!         the names of the separate files. The file names
!         must be provided in a specific order. For example:
! W2ONAME == wav1 to ocn1
!          wav1 to ocn2
!          wav1 to ocn3 ....for all the ocean models.
!          wav2 to ocn1
!          wav2 to ocn2
!          wav2 to ocn3 ....for all the ocean models.

   W2ONAME == wav2ocn_weights.nc
   W2ANAME == wav2atm_weights.nc
   A2ONAME == atm2ocn_weights.nc
   A2WNAME == atm2wav_weights.nc
   O2ANAME == ocn2atm_weights.nc
   O2WNAME == ocn2wav_weights.nc
```

!

That completes the coupling.in file. You point to that file to run a coupled application.

5.2- How to Compile and Run SCRIP_COAWST

SCRIP_COAWST is required to create a netcdf file that contains interpolation weights. These weights are only needed if you have a coupled application (more than 1 model) and the models are on different grids. Then you need to compile and run the SCRIP_COAWST and create the weights netcdf file. The SCRIP_COAWST needs to be built first, as described in Section 3. To run the program, you need to edit one of the scrip_coawst*.in files. Lets use
COAWST/Lib/SCRIP_COAWST/scrip_coawst_sandy.in as an example.
This Example uses 2 roms grids, 2 swan grids, and 2 wrf grids with the last wrf grid as a moving grid.

Step 1) Enter name of output netcdf4 file
OUTPUT_NCFILE='scrip_sandy_moving.nc'
!OUTPUT_NCFILE='scrip_sandy_static.nc' (this line is a comment)

Step 2) Enter total number of ROMS, SWAN, and WRF (max_dom) grids:
NGRIDS_ROMS=2,
NGRIDS_SWAN=2,
NGRIDS_WRF=2,
NGRIDS_WW3=0,

Step 3) Enter name of the ROMS grid file(s):
ROMS_GRIDS(1)='../../Projects/Sandy/Sandy_roms_grid.nc',
ROMS_GRIDS(2)='../../Projects/Sandy/Sandy_roms_grid_ref3.nc',

Step 4) Enter SWAN information:
! -the name(s) of the SWAN grid file(s) for coords and bathy.
! -the size of the SWAN grids, and
! -if the swan grids are Spherical(set cartesian=0) or
! Cartesian(set cartesian=1).
SWAN_COORD(1)='../../Projects/Sandy/Sandy_swan_coord.grd',
SWAN_COORD(2)='../../Projects/Sandy/Sandy_swan_coord_ref3.grd',
SWAN_BATH(1)='../../Projects/Sandy/Sandy_swan_bathy.bot',
SWAN_BATH(2)='../../Projects/Sandy/Sandy_swan_bathy_ref3.bot',
SWAN_NUMX(1)=84,
SWAN_NUMX(2)=116,
SWAN_NUMY(1)=64,
SWAN_NUMY(2)=86,
CARTESIAN(1)=0,
CARTESIAN(2)=0,

```
! 5) Enter WW3 information
!    -the name(s) of the WW3 grid file(s) for x- y- coords and bathy.
!    -the size of the WW3 grids (full number of grid center points).
!
WW3_XCOORD(1)='../../Projects/Sandy/ww3_sandy_xcoord.dat',
WW3_YCOORD(1)='../../Projects/Sandy/ww3_sandy_ycoord.dat',
WW3_BATH(1)='../../Projects/Sandy/ww3_sandy_bathy.bot',
WW3_NUMX(1)=84,
WW3_NUMY(1)=64,

Step 6) Enter the name of the WRF input grid(s). If the grid is a
!    moving child nest then enter that grid name as 'moving'.
!    Also provide the grid ratio, this is used for a moving nest.
WRF_GRIDS(1)='../../Projects/Sandy/wrfinput_d01',
!WRF_GRIDS(2)='../../Projects/Sandy/wrfinput_d02',    !this is a comment line
WRF_GRIDS(2)='moving',
PARENT_GRID_RATIO(1)=1,
PARENT_GRID_RATIO(2)=3,
PARENT_ID(1)=0
PARENT_ID(2)=1

Step 7): at the command prompt, run the program as:
./scrip_coawst[.exe] scrip_coawst_sandy.in
This should run and write out information about the weights as it creates the file.
```

5.3 Some general helpful tips and a few new functionalities.

5.3.1) ROMS is now very sensitive to the vertical coordinate system. The user needs to make sure that the values set in the roms init file are the same as the values set in the ocean.in file. For example the following values are checked for consistency:

```
Vtransform=1;
Vstretching=1;
theta_s=5.0;
theta_b=0.4;
Tcline=50.0;
N=16;
```

User should check that these values are the same in their netcdf files as the *.in file. More info is provided at: https://www.myroms.org/wiki/index.php/Vertical_S-coordinate.

5.3.2) ROMS grid files should not have fill values for mask arrays. A command to remove these are:

```
ncatted -O -a _FillValue,mask_rho,d,, -a _FillValue,mask_u,d,, -a
_FillValue,mask_v,d,, -a _FillValue,mask_psi,d,, USeast_grd17.nc
```

5.3.3) For SWAN, you do not need to specify any OUTPUT commands for the coupling. The coupling and OUTPUT are now completely separate. SWAN now offers writing out of files in netcdf. This option is available with the COAWST format.

5.3.4) For WRF, you can have a 2-way nest in WRF, and have this coupled to roms and /or swan. As of COAWST v3.2, we can now couple ROMS and SWAN to a moving WRF nest. There can only be 1 moving WRF nest, and it needs to be the last WRF child grid.

5.3.5) For WRF-ROMS coupling, you really should set
sst_update = 1
in namelist.input and use the appropriate io_form_auxinput4 settings. This is correct in svn 876. The sst_update computes the correct TSK in WRF and should be activated for ocn-atm coupling.

5.3.6) Some information about heat fluxes for WRF-ROMS.
If WRF_MODEL is defined:

- you still have to define EMINUSP to activate exchange of rain and evap.
- SOLAR_SOURCE is needed, otherwise all the heat goes into the surface layer only.
- longwave outgoing component is estimated in Master/mct_roms_wrf.f so there is no need to define LONGWAVE_OUT in ROMS

If WRF_MODEL is not defined or you are going to use BULK FLUXES:
BULK_FLUXES (in bulk_flux.F) computes turbulent heat fluxes (latent and sensible heat), momentum stress and evaporation (used in the fresh water flux if EMINUPS is also defined -used as salinity surface boundary condition-). Radiative fluxes (i.e., shortwave and longwave radiation flux) are not calculated, nor is the rain component of the EMINUSP calculation. The surface scheme (COARE) implemented in bulk_flux.F requires:
- air temperature (usually at 2m)
- relative humidity (usually at 2m)
- mean sea level pressure
- u-wind component (positive east), usually at 10m v-wind component (positive north), usually at 10m.
With these parameters bulk_flux will estimate latent heat, sensible heat, u-momentum stress, v-momentum stress and evaporation. Note that in the ocean.in, you have to specify:
BLK_ZQ (usually 2m)
BLK_ZT (usually 2m)
BLK_ZW (usually 10m)
these numbers should be consistent with the height of the levels of the surface variables (as said usually wind is at 10m, air temp at 2m, humidity at 2m, but this might be different depending on your surface forcing dataset).

Net shortwave should be provided by you meteo forcing. This is not calculated in bulk_flux.f, but is necessary to compute the full heat flux term.
Net longwave: you have several choices:
- provide net longwave in the forcing file

- provide INCOMING longwave in the forcing file and define LONGWAVE_OUT (ROMS then will estimate the outgoing component based on its SST)
- do not provide the longwave but instead total cloud cover (in the forcing file) and ROMS will estimate the net longwave. You do not need to define CLOUD, as it is defined internally by ROMS if def LONGWAVE

If you want the E-P flux, define EMINUSP and provide in the forcing file the variable rain, while, as said, evaporation is estimated in bulk_flux.F.

So, in the end:
#define BULK_FLUXES
#define LONGWAVE or #define LONGWAVE_OUT or provide the net longwave in the forcing file
#define EMINUSP is needed, otherwise set to zero the surface salinity fux (#define ANA_SSFLUX and set zero stflx(itrc==isalt) in stflux.h)
#define ATM_PRESS if you want the inverted barometric effect (mean sea level pressure must be in the forcing file)

5.3.7) SWAN restart files.
SWAN has the command 'HOTFILE' that is used to write out the model state of two-dimensional spectra to a file at the end of the simulation. This file can be used as a restart for a future simulation. We have recently added the option to have SWAN create restart files during the simulation. This can be useful for a coupled simulation (for example a roms + swan run). If the modeling system reaches a fault during the run, the ROMS model can be restarted with the ROMS rst or his file. Now you can also restart the SWAN model with a restart file for that same instance in time. You need to add the command "RESTART'. For example, the swan.in file would need:

RESTART 'swan_intest_rst.dat' FREE 1 HR
COMPUTE NONSTATIONARY 20000101.000000 60 SEC 20000101.120000
STOP

With this set of commands, swan is going to run from Jan 1, 2000, at hour 0 to Jan 1, 2000 to hour 12, with a 60 sec time step. The 'RESTART' command will update a file called swan_inlet_rst.dat every 1 hour during that simulation. You can make this be any time you require (1 HR or 30 MIN or 12 HR or whatever). It is recommended that the user set the restart times to be the same for WRF, ROMS, and SWAN so that a consistent set of restart files are available.

6. Distributed Projects/examples.

Application	
6.1 Ducknc	ROMS only. Wave-current interaction (essentially x-z) for cross-shore flows at Duck, NC.
6.2 Estuary_test2	ROMS only to test estuarine dynamics, prismatic channel.

6.3 Inlet_test/Coupled	Idealized inlet with wave and tidal driven flows. ROMS+SWAN same grids.
6.4 Inlet_test/DiffGrid	Idealized inlet with wave and tidal driven flows. ROMS+SWAN different grids.
6.5 Inlet_test/InWave	Idealized inlet with wave and tidal driven flows. ROMS+InWave same grids, inlet configuration with infragravity waves.
6.6 Inlet_test/Refined	Idealized inlet with wave and tidal driven flows. ROMS+SWAN have the same parent grids for roms and swan and the same child grids for roms and swan.
6.7 Inlet_test/Swanonly	Idealized inlet with waves only. SWAN only one grid, or with grid refinement.
6.8 InWave_Shoreface	ROMS+InWave same grid, sloping beach.
6.9 JOE_TC/Coupled	Idealized tropical cyclone that travels west from a deep ocean basin onto a shelf that slopes landward to the west. WRF+ROMS+SWAN all 3 on same grid. Also can be used for WRF-ROMS, WRF-SWAN, or WRF only.
6.10 JOE_TC/DiffGrid	Idealized tropical cyclone that travels west from a deep ocean basin onto a shelf that slopes landward to the west. WRF+ROMS+SWAN with ROMS and SWAN on same grid, WRF on different grid.
6.11 Rip_current	ROMS+SWAN same grid, idealized rip current.
6.12 Sandy	WRF + ROMS + SWAN each with 2 grids, WRF can be static or moving nest, coarse resolution of realistic Hurricane Sandy simulation. Also distributed as WRF + ROMS + WW3 each with one grid.
6.13 Sed_floc_toy	ROMS only, tests sediment module with flocculation.
6.14 Sed_bed_toy	ROMS only, tests sediment module.
6.15 Trench	ROMS only, test sediment morphology.
6.16 Veg_test	ROMS+SWAN, test vegetation module.
6.17 Wetdry	ROMS only, test wetting/drying algorithms.
6.18 Sedbio_toy	ROMS + Sediment+Water-column biogeochemistry (Fennel model)
6.19 Veg_growth_test	ROMS+vegetation+sediment+water-column biogeochemistry (including spectral light module), Test Submerged Aquatic Vegetation (SAV) growth model and fully coupled hydrodynamic-SAV-biological and sediment dynamics
6.20 Marsh_test	ROMS+marsh dynamics (lateral wave erosion+biomass growth)

All of the test cases are described below, some in more detail than others. The cases of JOE_TC, Inlet_test, Rip current and Vegetation are examples of how to set up different

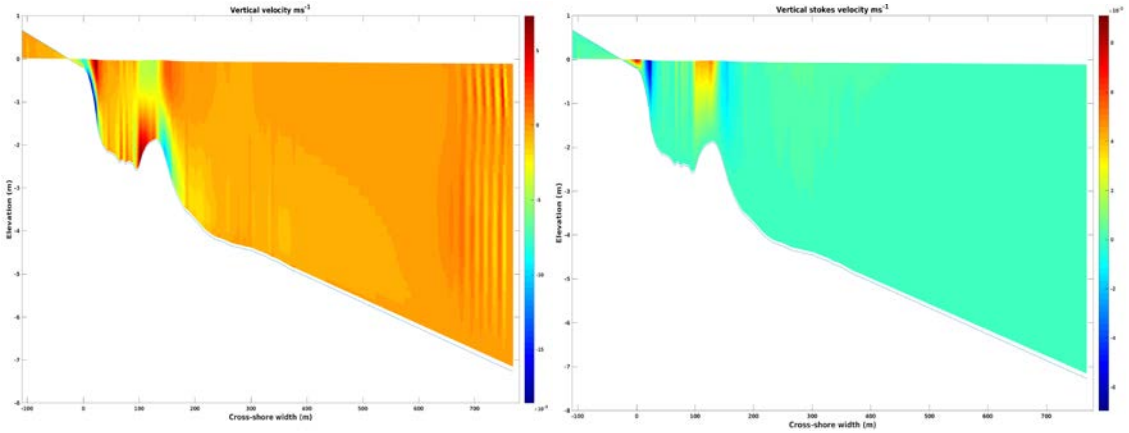
idealized configurations. The test case Sandy is provided for a more realistic application. The references for the test cases are provided in Section 15.

6.1 Ducknc = ROMS only, Wave-current interaction (essentially x-z) for cross-shore flows at Duck, NC

This test case validates the implementation of the Wave effect on Currents (WEC) module in the COAWST framework. The case is based upon experriemnts from the Duck (94) experiment. The experiment includes oblique waves that are incident on a synthetic planar beach and a natural barred beach. The simulations show the capability of the WEC module in COAWST to capture nonlinear processes of wave roller generation and wave-induced mixing. The details of the test case are presented in Kumar et al. (2013). To setup the case modify the coawst.bash to include this application:

export ROMS_APPLICATION=DUCKNC

Figures 6.1(a) and 6.1(b) show the vertical velocity and vertical stokes velocity respectively varying with cross-shore width (x) at a middle along-shore plane (y=4) at the end of 8400 seconds.



(a) Vertical velocity (b) Vertical stokes velocity

Figure 6.1: Results from Ducknc test case

6.2 Estuary_test2 = ROMS only, to test estuarine dynamics, prismatic channel

This application tests the ROMS for a prismatic shaped estuary that is 110 km long with the ocean end being 60 kms wide. The width of the estuary decreases exponentially from 60 km to 2 kms until 80 kms and stays uniform beyond 80 kms until the river end. The vertical layers follow a gaussian shape and consist of 40 vertical levels. The simulation is run for 5 days.

The application is run by editing the coawst.bash file to set

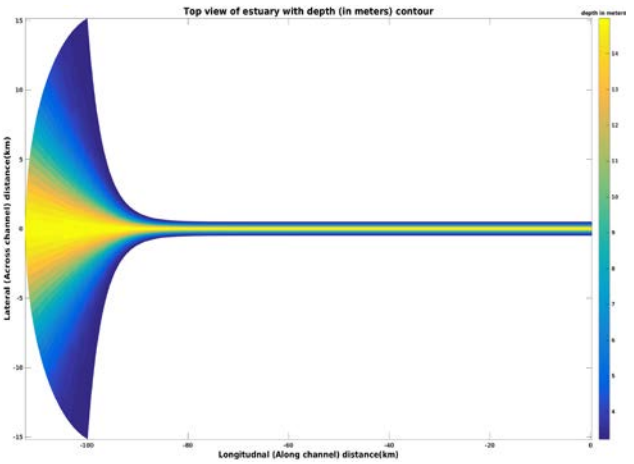
export ROMS_APPLICATION=ESTUARY_TEST2

なぜ必要？

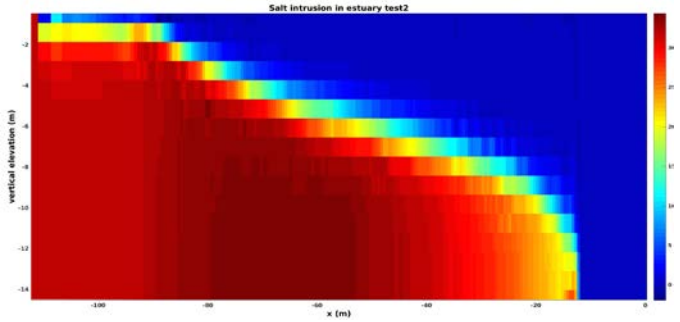
This case requires mpi and the MCT libraries because it is a coupled application. And the run files are at:

export MY_HEADER_DIR=${MY_PROJECT_DIR}/Projects/Estuary_test2

export MY_ANALYTICAL_DIR=${MY_PROJECT_DIR}/Projects/Estuary_test2



a) Estuary grid



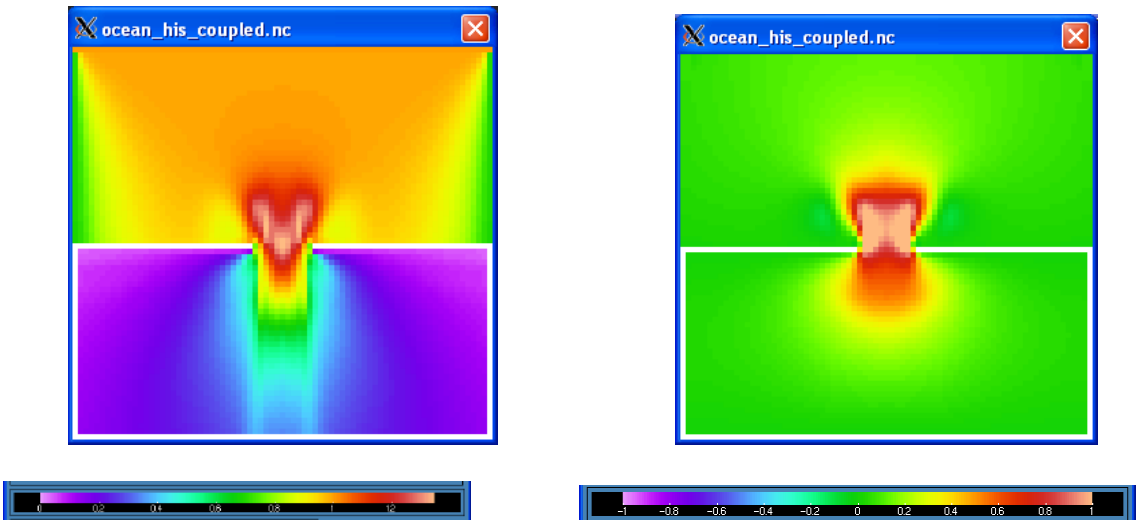
b) Salt intrusion length after 5 days

Figure 6.2: Results from Estuary_test2 test case

The above figures (6.2a) and (6.2b) show the horizontal grid describing the shape of the estuary along with the salt intrusion at the end of 30 days.

6.3. INLET_TEST/Coupled = ROMS+SWAN same grid

This application tests the MCT coupling between ROMS and SWAN for an idealized inlet. The application has an enclosed basin in the southern part of the domain with a small opening near the center of the grid. The model is driven by a sinusoidal water level on the northern edge and is coupled to SWAN which develops waves from the north with a 1m height propagating towards the south. Flow pattern that developed are shown below:



Wave height.
(Hwave)

depth integrated current to wave model
(vWave)

Figure 6.3. Results for Inlet_test coupled.

All of the grids and input files have already been created for this application and are available in Projects/Inlet_test/Coupled. Here is a basic set of steps required to make these files and run this application.

Step 1: Use create_roms_xygrid.m to create the roms and swan grids. These are simple rectangular grids and can be created easily with these tools. User sets the x/y locations, depth, and masking. This m file calls mat2roms_mw to create the roms grid file called inlet_test_grid.nc and calls roms2swan to create the swan grid files. The swan grid files were renamed to inlet_test_grid_coord.grd and inlet_test_bathy.bot.

Step 2: For this case, we are initializing the roms model from rest, so we are relying on ANA_INITIAL default values. The m file create_roms_init could be used to create a more complicated set of init conditions. We are also using ana_sediment to create a uniform field of 10m thick bed with porosity=0.5 and 1 grain size. This could have been created with the create_roms_init file as well. The file sediment_inlet_test.in lists some sediment properties for this application.

Step 3: For this case we are initializing SWAN from rest so there are no swan init files. we could have run swan for with some wind field, but this case has no winds.

Step 4: create the header file: Edit inlet_test.h to see the options selected for this application. we needed to define
#define ROMS_MODEL
#define SWAN_MODEL
#define MCT_LIB

, as well as several boundary conditions and wave current interaction (WEC) options, the wave-enhanced bottom boundary layer option (SSW_BBL), GLS_MIXING, and some sediment options of suspended load and bed morphology updating.

Step 5: Determine settings for roms and swan in the ocean_inlet_test.in and swan_inlet_test.in. These are some standard options in these files.

Step 6: Create the coupling file called coupling_inlet_test.in to enter the roms and swan input file names and determine the coupling interval.

Step 7: edit the coawst.bash file to build this application.

Step 8: compile and run.

6.4. INLET_TEST/DiffGrid = ROMS+SWAN different grids

This test case has a grid for ROMS and a larger grid for SWAN. This is useful for applications where the lateral shadow effects of the wave model can influence the ocean grid. So the wave model can be simulated on a larger grid and the ocean model on a smaller inset grid. This application is very similar to case 4 above i.e. inlet_test/coupled. All of the files generated for this test case are in Projects/Inlet_test/DiffGrid.

Step 1: ROMS and SWAN are on different grids. You need to create a roms grid and a swan grid. This is achieved using create_roms_xygrid.m.

Step 2: Create the interpolation weights file – (*The interpolation weight file for this case is already provided right now in the folder as "scrip_inlet_test_diffgrid.nc").* Follow the steps below to create the interpolation weight file yourself.

Build SCRIP_COAWST
We will use the SCRIP_COAWST package provided in "Lib" folder to create the interpolation weights. The steps for this are described in *Section 3* above.

Create the interpolation weights file -
The interpolation weight file for this case is already provided right now in the folder as "scrip_inlet_test_diffgrid.nc". But here are the steps to create it.
- Edit the flie Lib/SRIP_COAWST/scrip_coawst_inlet_test_diffgrid.in. This is the input file and has already been created for this application. The information has already been set for the output_ncfile name, the number of grids for each model, and the roms and swan grid information.
- Execute the scrip_coawst by going to the path of the SCRIP_COAWST folder and then running the program using
./scrip_coawst scrip_coawst_inlet_test_diffgrid.in
This will create the weights file scrip_inlet_test_diffgrid.nc
You can then copy that weights nc file to the Project folder
cp scrip_inlet_test_diffgrid.nc ../../Projects/Inlet_test/Diffgrid

Step 3: Edit the Projects/Inlet_test/Diffgrid/coupling_inlet_diffgrid.in and use
SCRIP_WEIGHT_OPTION=1
Add the lines:
SCRIP_COAWST_NAME=Projects/Inlet_test/DiffGrid/scrip_inlet_test_diffgrid.nc

Step 4: Edit the ocean model configuration file.
Edit Projects/Inlet_test/DiffGrid/inlet_test.h and add the option
#define MCT_INTERP_OC2WV
This will allow mct interpolation between the ocean and wave models.

Step 5: Change the ocean and wave input file names to be
WAV_name = Projects/Inlet_test/DiffGrid/swan_inlet_test_diff.in ! wave model
OCN_name = Projects/Inlet_test/DiffGrid/ocean_inlet_test.in ! ocean model

Step 6: make, run the program

6.5. INLET_TEST/InWave = ROMS+InWave, Inlet test case.
This case is an example to use the InWave infragravity wave model. The application is an inlet with an enclosed back basin, with offshore open boundaries, similar to the other inlet tests but with a smaller domain size. The offshore northern boundary is driven by a 2Dspec file from SWAN. The model reads that 2d spec file, generates the envelope of the infragravity waves, and propagates them into the domain. (The only other way to drive InWave is to prescribe the wave action density along the open boundary, and that is described with the Projects/InWave_shoreface test case.)

To generate the files for this inlet_test case, you need to edit Tools/mfiles/inwave_tools/master_InWave_create_files.m. This is the main driver to create files for the InWave applications. All of the files generated for this test case are placed in Projects/Inlet_test/InWave.

Step 1: Edit Tools/mfiles/inwave_tools/master_InWave_create_files.m and select to setup the Inlet_test case

```
%1) SET THE CASE TO = 1.
INWAVE_SHOREFACE=0;
INLET_TEST=1;
```

This will tell the program to look at a configuration file called

```
elseif (INLET_TEST)
    inwave_gen_file='InWave_inlet_test_param';
```

So the goal here is to create the InWave_inlet_test_param file, then just run the master m file.

Step 2:
Create (edit) the file Tools/mfiles/inwave_tools/InWave_inlet_test_param.m. Follow the instructions in that m file. We need 3 netcdf files: grid, initial, and boundary files. For this test case we need a grid and an initial file. We do not need to create a boundary file

because this case reads in a 2dspec file and will internally compute the wave energy. So we selected:

```
make_InWave_grd=1;
make_InWave_ini=1;
make_InWave_bry=0;
```

To make the grid, we set up some basic paramters for dx, dy, x, y, depth, angle, masking, f, and spherical. See the m file.
For the initial conditions, we need to set the action density (Ac) and wave celerities (cx, cy, ct). We just start them all at 0. You also need to specify the number of computational bins (Nbins == ND in the ocean.in file), the directions of these bins (direction the wave is coming from), and a representative period (Ta).

Step 3:
Edit the inlet_test.h file, to add InWave cpp options:

```
#ifdef  INWAVE_MODEL
# define INWAVE_SWAN_COUPLING
# define ACX_ADVECTION
# define ACY_ADVECTION
# define ACT_ADVECTION
# undef  DOPPLER
# define WDISS_GAMMA
# undef  WDISS_ROELVINK
#endif
```

The “INWAVE_SWAN_COUPLING” says that it will read a 2d spec file from swan. The AC*_ADVECTION activate the advective terms (like UV_ADV for ROMS). The DOPPLER could be activated, and adds the effects of currents to the celerities. Then there are two choics for wave dissipation – a gamma approach and that of Roelvink.

Step 4:
Edit the Projects/Inlet_test/InWave/ocean_inlet_test.in
The key new options here would be to set

-1) The number of wave directional bins for the computation. This needs to be the same as in the init file

```
      ND == 20          ! Number of wave directional bins
```

- 2) Inwave boundary conditions:

```
! InWave boundary conditions
!                       west    south    east    north
  LBC(isAC3d) ==   Gra     Clo     Gra     Cla        ! 3D wave action density
  LBC(isCT3d) ==   Gra     Clo     Gra     Gra        ! 3D wave theta celerity
  LBC(isCX3d) ==   Gra     Clo     Gra     Gra        ! 3D wave x-dir celerity
  LBC(isCY3d) ==   Gra     Clo     Gra     Gra        ! 3D wave y-dir celerity
```

For this case, we want the 2dspec to be applied on the north (isAC3d is Clamped). The others can be gradient, or closed (on the south as that is a wall).

- 3) Set output or parameters

```
Hout(idACen) == T      ! AC          3D wave action
Hout(idACcx) == T      ! Cx          2D Group velocity in xi dimension
Hout(idACcy) == T      ! Cy          2D Group velocity in etai dimension
Hout(idACct) == T      ! Ct          2D Group velocity in the directional dimension
Hout(idACtp) == T      ! Tp          2D Group peak period
```

- 4) enter the grid and ini files:

GRDNAME == Projects/Inlet_test/InWave/InWave_inlet_test_grd.nc
(this is the same as the roms grid)

IWININAME == Projects/Inlet_test/InWave/InWave_inlet_test_ini.nc
(this is the ini file created with AC, ct, cx, cy, Ta).

Step 5: edit coawst.bash and build this as any other application. Run as any other application. An example of the output is shown here for the wave height.

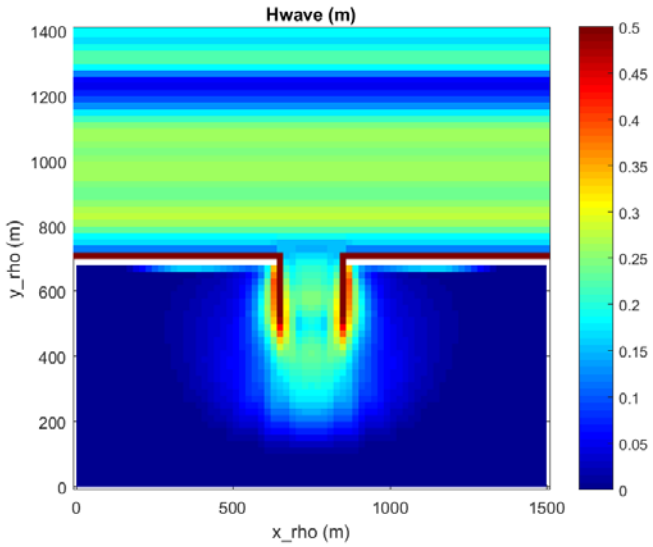


Figure 6.5. Significant wave hegith at 180s for the InWave inlet test.

6.6. INLET_TEST/Refined = ROMS+SWAN same grid + grid refinement in each.

This test case has a coarse and fine grid for ROMS, and a coarse and fine grid for SWAN, run coupled on both grids. All of the files generated for this test case are in Projects/Inlet_test/Refined.

Step1: You need to create a roms grid for the coarse model, roms grid for fine model, swan grid for coarse model, and swan grid for fine model. Suggest you do this:

- First create a roms grid for the coarse model. This was done using Tools/mfiles/create_roms_grid and selecting the inlet_test case.
- To create the refined grid, use the following procedure:

```
F=coarse2fine('inlet_test_grid.nc','inlet_test_grid_ref5.nc',3,24,54,40,56);
Gnames={'inlet_test_grid.nc','inlet_test_grid_ref3.nc'}
[S,G]=contact(Gnames,'inlet_test_contact_ref3.nc');
```

This uses m files from Rutgers and they are in the Tools directory. These calls set a refinement ratio of 3 and set the starting and ending indices of the child grid to range from Istr=24 to Iend =54, and Jstr = 40 to Jend =56. When you run those 3 commands above a series of figures are created that allow you to see the parent and child grids. The inlet test grids and the contact files need to be listed in the ocean.in file ocean_inlet_test_ref3.in.

- To create the two swan grids, you can use Tools/mfiles/roms2swan.m from the 2 roms grids.

Step 2: Create input files for the ocean and the swan grids. You need one ocean.in file, with the parameter values repeated for the 2 grids. See Projects/Inlet_test/Refined/ocean_inlet_test_ref3.in. Edit this ocean*.in and set values for all Numgrids (ie dt, file names, etc.)
- For Lm and Mm the values are -2 of the total number of rho points.

Step 3: You need to create to separate SWAN INPUT files. See that same folder for the 2 swan files (swan_inlet_test.in and swan_inlet_test_ref3.in).

Step 4: Need to create init and bndry files for the largest scale ocean and wave grids.

Step 5: Need to create init file for each ocean and wave refined child grids. The child grids do not need boundary files.

Step 6: Edit the Projects/Inlet_test/Refined/coupling_inlet_test_refined3.in file and list the input files: swan has two, just one for roms.

Step 7: Create the interpolation weights file – (*The interpolation weight file for this case is already provided right now in the folder as "scrip_inlet_test_refined.nc").* Follow the steps below to create the interpolation weight file yourself.

Build SCRIP_COAWST
We will use the SCRIP_COAWST package provided in "Lib" folder to create the interpolation weights. The steps for this are described in *Section 3* above.

Create the interpolation weights file -
The interpolation weight file for this case is already provided right now in the folder as "scrip_inlet_test_refined.nc". But here are the steps to create it.
- Edit the flie Lib/SRIP_COAWST/scrip_coawst_inlet_test_refined.in. This is the input file and has already been created for this application. The information has already been set for the output_ncfile name, the number of grids for each model, and the roms and swan grid information.

- Execute the scrip_coawst by going to the path of the SCRIP_COAWST folder and then running the program using
./scrip_coawst scrip_coawst_inlet_test_refined.in
This will create the weights file scrip_inlet_test_refined.nc
You can then copy that weights nc file to the Project folder
cp scrip_inlet_test_refined.nc ../../Projects/Inlet_test/Diffgrid

Step 8: Edit the Projects/Inlet_test/Refined/coupling_inlet_test_ref3.in and use
SCRIP_WEIGHT_OPTION=1
Add the lines:
SCRIP_COAWST_NAME=Projects/Inlet_test/Refined/scrip_inlet_test_refined.nc

Step 9: Build the model
./coawst.bash –j
and run it using
mpiexec –np 2 ./coawstM.exe Projects/Inlet_test/Refined/coupling_inlet_test_refined3.in

6.7. INLET_TEST/Swanonly = SWAN by itself, also with grid refinement.
This test case is basically the swan files from case 7 (mentioned above), but allows you to run swan by itself, and with a child grid.

To run SWAN by itself, with only 1 grid, the header file should have
#define SWAN_MODEL
#undef NESTING

./coawst.bash to compile.
One note is that this needs to be run with the command pointing to the swan input file(s). So to run the swan only with just one grid, run the model with (X can be any number of processors)
mpiexec –np X ./coawstM.exe Projects/Inlet_test/Swanonly/swan_inlet_test.in

To run SWAN by itself and with a child grid, the header file should have
#define SWAN_MODEL
#define NESTING

./coawst.bash to compile.
The command line needs to point to the swan input files. So to run the swan only with a parent and a child use (X can be any number of processors)
mpiexec –np 1 ./coawstM.exe Projects/Inlet_test/Swanonly/swan_inlet_test.in
Projects/Inlet_test/Swanonly/swan_inlet_test_refined5.in
(all that is on one line, the text is just word wrapping here).

6.8 Inwave_shoreface = ROMS+InWave (Sloping beach).
This case is an example to use the InWave infragravity wave model. The application is an inlet with a simple sloping beach in the north-south direction, eastern wall, open western

coast. The south, north, and western boundaries are driven by wave action density from a netcdf file. This test case is advanced in that the user needs to understand the physics to create the wave actions time series. The other way to drive InWave is to provide a SWAN 2d spec file and that is described with the Projects/Inlet_test/InWave test case.)

To generate the files for this inlet_test case, you need to edit Tools/mfiles/inwave_tools/master_InWave_create_files.m. This is the main driver to create files for the InWave applications. All of the files generated for this test case are placed in Projects/Inlet_test/InWave.

Step 1: Edit Tools/mfiles/inwave_tools/master_InWave_create_files.m and select to setup the Inlet_test case

```
%1) SET THE CASE TO = 1.
INWAVE_SHOREFACE=1;
INLET_TEST=0;
```

This will tell the program to look at a configuration file called

```
if (INWAVE_SHOREFACE)
    inwave_gen_file='InWave_shoreface_param';
```

So the goal here is to create the InWave_shoreface_param file, then just run the master m file.

Step 2:
Create (edit) the file Tools/mfiles/inwave_tools/InWave_shoreface_param.m. Follow the instructions in that m file. We need 3 netcdf files: grid, initial, and boundary files. For this test case we need all 3 – grid, initial, and boundary files so we selected:

```
make_InWave_grd=1;
make_InWave_ini=1;
make_InWave_bry=1;
```

To make the grid, we set up some basic paramters for dx, dy, x, y, depth, angle, masking, f, and spherical. See the m file.

For the initial conditions, we need to set the action density (Ac) and wave celerities (cx, cy, ct). We just start them all at 0. You also need to specify the number of computational bins (Nbins), the directions of these bins (direction the wave is coming from), and a representative period (Ta).

To create the boundary forincg file, the user needs to create the time series of wave action density. This is a user defined option.

Step 3:
Edit the Projects/inwave_shoreface.h file, to add InWave cpp options:
#ifdef INWAVE_MODEL
undef INWAVE_SWAN_COUPLING
define ACX_ADVECTION
define ACY_ADVECTION

```
# define ACT_ADVECTION
# undef  DOPPLER
# define WDISS_GAMMA
# undef  WDISS_ROELVINK
#endif
```

The “INWAVE_SWAN_COUPLING” is to read s 2d spec file from swan, and we are not doing that for this application. The AC*_ADVECTION activate the advective terms (like UV_ADV for ROMS). The DOPPLER could be activated, and adds the effects of currents to the celerities. Then there are two choics for wave dissipation – a gamma approach and that of Roelvink.

Step 4:
Edit the Projects/InWave_shoreface/ocean_inwave_shoreface.in
The key new options here would be to set

-1) The number of wave directional bins for the computation. This needs to be the same as in the init file

```
      ND == 11          ! Number of wave directional bins
```

- 2) Inwave boundary conditions:

```
! InWave boundary conditions
!                      west    south    east    north
  LBC(isAC3d) ==   Cla     Gra     Clo     Gra        ! 3D wave action density
  LBC(isCT3d) ==   Gra     Gra     Clo     Gra        ! 3D wave theta celerity
  LBC(isCX3d) ==   Gra     Gra     Clo     Gra        ! 3D wave x-dir celerity
  LBC(isCY3d) ==   Gra     Gra     Clo     Gra        ! 3D wave y-dir celerity
```

For this case, we want the wave action to be applied on the west (isAC3d is Clamped). The others can be gradient, or closed (on the east as that is a wall).

- 3) Set output or parameters

```
Hout(idACen) == T      ! AC            3D wave action
Hout(idACcx) == T      ! Cx            2D Group velocity in xi dimension
Hout(idACcy) == T      ! Cy            2D Group velocity in etai dimension
Hout(idACct) == T      ! Ct            2D Group velocity in the directional dimension
Hout(idACtp) == T      ! Tp            2D Group peak period
```

- 4) enter the grid and ini files:

```
    GRDNAME == Projects/InWave_shoreface/InWave_shoreface_grd.nc
   (this is the same as the roms grid)

    IWININAME == Projects/InWave_shoreface/InWave_shoreface_ini.nc
   (this is the ini file created with AC, ct, cx, cy, Ta).
```

Step 5: edit coawst.bash and build this as any other application. Run as any other application. An example of the output is shown here for the wave height.

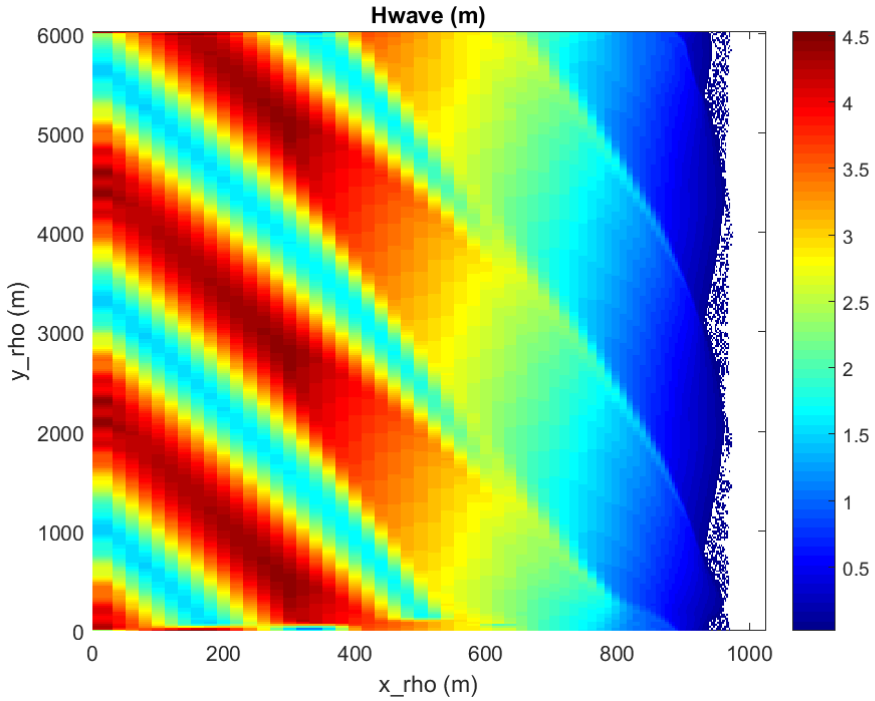


Figure 6.8. Significant wave hegith at 200s for the InWave_shoreface test.

6.9 JOE_TC/Coupled = WRF-ROMS-SWAN all 3 on same grid.

Step 1: Create a folder to hold all the project files. For this application a folder is already made and is called COAWST/Projects/JOE_TC/Coupled. This will be referred to as the 'project folder.'

Step 2: Create WRF grid, init file, and input files:
wrfbdy_d01
wrfinput_d01
namelist.input
These files need to be placed in the root_dir.
Copies of these files have already been created and are located in the Projects/JOE_TC folder. Copy these 3 files from that folder to the root dir.
Edit namelist.input and set in the &domains section the number of processors for WRF to use for the simulation (total WRF = nproc_x*nproc_y).
nproc_x = M
nproc_y = N

step 3: Create a project 'header' file.
The application control file has already been created for this setup. For this case, we can use the file in the project directory
COAWST/Projects/JOE_TC/Coupled/joe_tc.h

We have preselected to use a certain case G, which has the 3 way coupling.
Some key parameters in here are ….
define ROMS_MODEL
define SWAN_MODEL
define WRF_MODEL
If you want to create your own, the best choice is to start out by copying an 'application.h' file from some other project folder which is similar to your configuration, and then tailor it to the new application.

Step 4: Build the system.
Edit coawst.bash and provide appropriate file locations and run the build command:
./coawst.bash
During the build if wrf asks for the location of the netcdf files, you can provide them. If you set NETCDF=___ to the correct location in your environment, then WRF will know where the files are.
During the buld wrf will ask for the system you are using such as:

Select option 3, PC Linux x86_64, PGI compiler 5.2 and higher (RSL_LITE)
And select 1 for normal. Compiling wrf can take up to 20 minutes or longer.

Step 5: Create the ocean grid. The project folder already contains the roms grid. This grid was made from a version of :
COAWST/Tools/mfiles/create_roms_grid.m.
Alternatively, we now distribute wrf2roms_mw.m that can be used to create a roms grid from a wrf grid. After creating the grid, copy the grid file (in this case joe_tc_grd.nc) to the project directory.

Step 6: Create the initial file for the ocean model. The project folder already contains the roms init file but if you want to see how it was made you can use: COAWST/Tools/mfiles/create_roms_init.m.
After creating the init file copy the file (in this case joe_tc_ocean_init.nc) to the project directory.

Step 7: Create an ocean input file.
The application ocean input file has already been created and we can use the file in the project directory
COAWST/Projects/JOE_TC/Coupled/ocean_joe_tc.in
Set the total number of processors to be allocated to roms as:

```
NtileI == 4                    ! I-direction partition
NtileJ == 3                    ! J-direction partition
```

Step 8: Make the swan bathy + grid files.
When you use create_roms_grid, there were other files created:
roms_bathy.bot
grid_coord.grd

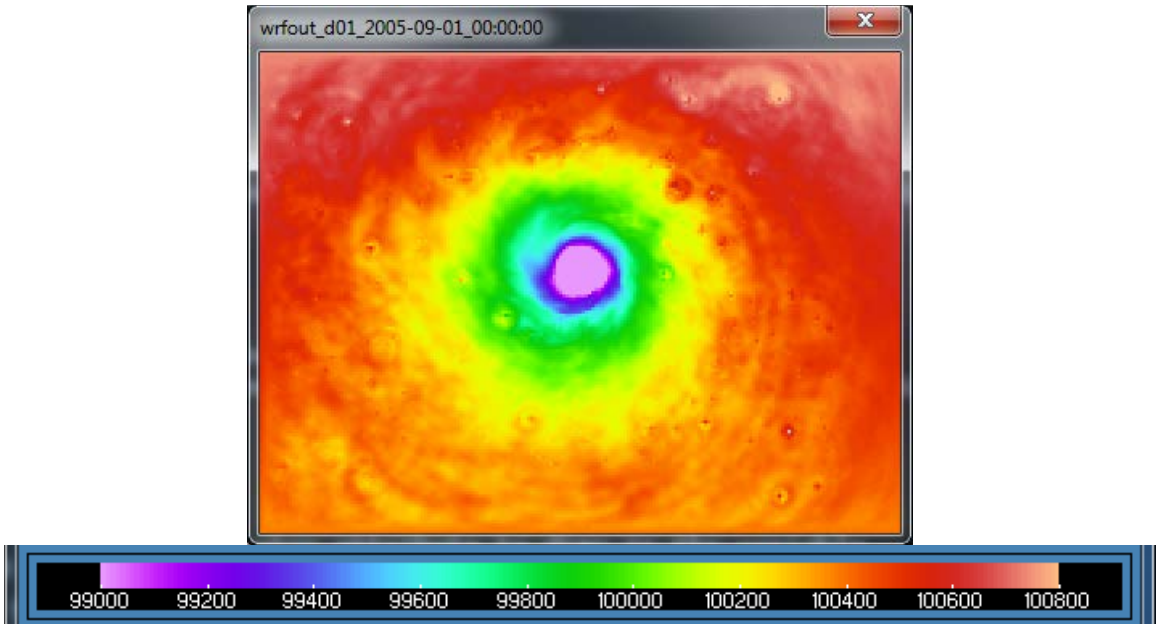
These were created by roms2swan.m and these are the swan files needed. Rename these (such as joe_tc_roms_bathy.bot and joe_tc_grid_coord.grd) and copy these 2 files to the project directory. (This has already been done.)

Step 9: create swan input file
The best step here is to copy an existing file and tailor it to your application. For now we can use:
COAWST/Projects/JOE_TC/Coupled/INPUT_JOE_TC

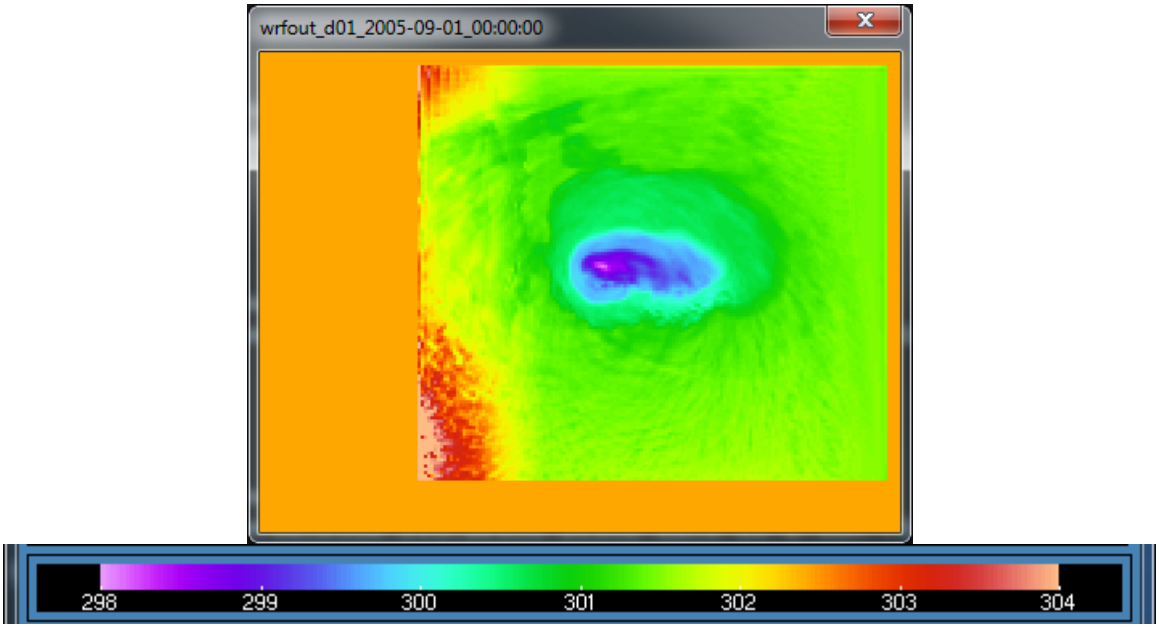
Step 10: modify the coupling.in file COAWST/Projects/JOE_TC/Coupled /coupling_joe_tc.in to:
- allocate processors for all 3 models
- set coupling time interval
- list input file names

```
  NnodesATM =  4                ! atmospheric model
  NnodesWAV = 4                 ! wave model
  NnodesOCN =  4                ! ocean model

! Time interval (seconds) between coupling of models.

 TI_ATM_WAV =   600.0d0          ! atmosphere-wave coupling interval
 TI_ATM_OCN =    600.0d0         ! atmosphere-ocean coupling interval
 TI_WAV_OCN =   600.0d0          ! wave-ocean coupling interval

! Coupled model standard input file name.

   ATM_name = namelist.input                    ! atmospheric model
   WAV_name = Projects/JOE_TCs/INPUT_JOE_TC          ! wave model
   OCN_name = Projects/JOE_TCs/ocean_joe_tc.in       ! ocean model
```

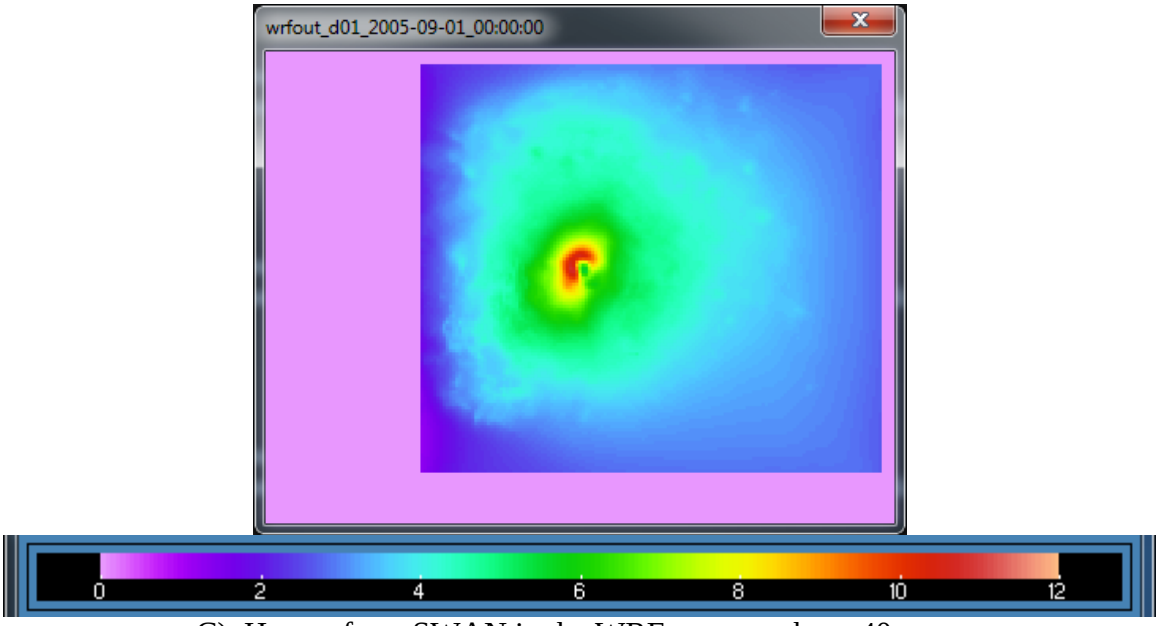
Step 11: run the system using a PBS run script or whatever is for your system
/usr/local/mpi/bin/mpirun -np 12 -machinefile $PBS_NODEFILE ./coawstM Projects/JOE_TC/coupling_joe_tc.in > joe_tc.out



A) PSFC from WRF at hour 40.



B) SST from ROMS in the WRF output at hour 40.



C) Hwave from SWAN in the WRF output at hour 40.

Figure 6.9. Results at hour 40 for JOE_TCs simulation for: A) PSFC from WRF; B) SST from ROMS; and C) HWAVE from SWAN.

Additional notes for JOE_TCS:
This test case is distributed with "ExpG" defined. That case has ROMS+WRF+SWAN all coupled. You can select to activate a different case in the Projects/JOE_TC/Coupled /joe_tc.h file, such as "ExpA1" which is just WRF+SWAN, or "ExpA" which is just WRF+ROMS.

6.10 JOE_TC/DiffGrid simulation using ROMS + SWAN same grid, WRF different grid.

This application provides information on how to run the models on different grids. Each model could be on a separate grid, but for this example roms and swan are on the same grid, and wrf is on a different grid. This will be very close to the setup in 1 but now we will just create new grids for roms and swan. The roms and swan grids will be decreased resolution to have 100 cells in the x-direction and 75 cells in the y-direction (instead of 200x150).

Step 1: Create a new projects folder
Projects/JOE_TC/DiffGrid. (this has already been created).

Step 2: Create roms + swan grids. This can be accomplished using
COAWST/Tools/mfiles/create_roms_xygrid.m or wrf2roms_mw.
We renamed grid_coord.grd and roms_bathy.bot to
joe_tc_coarse_grid_coord.grd and joe_tc_coarse_roms_bathy.bot. Copy these 2 files and the joe_tc_coarse_grd.nc file to the project folder.

Step 3: create roms init files using

COAWST/Tools/mfiles/mtools/create_roms_init.m

Step 4: Create wrf files: we will use the same wrf files from before of:
wrfbdy_d01
wrfinput_d01
namelist.input
These files need to be placed in the root_dir (I also make a copy and put into the project folder).

Step 5: Create the interpolation weights file – (*The interpolation weight file for this case is already provided right now in the folder as "scrip_joe_tc_diffgrid.nc").* Follow the steps below to create the interpolation weight file yourself.

Build SCRIP_COAWST
We will use the SCRIP_COAWST package provided in "Lib" folder to create the interpolation weights. The steps for this are described in *Section 3* above.

Create the interpolation weights file -
- Edit the "scrip_coawst_joetc_diffgrid.in". This is the input file and needs to be edited. (*The input file for this case is provided*). The name of the output interpolation weight file can be added at the beginning of this file. Let us call it "scrip_joe_tc_diffgrid.nc"
- Execute the scrip_coawst by going to the path of the SCRIP_COAWST folder and then running the program using
./scrip_coawst scrip_coawst_joe_tc_diffgrid.in
This will create the weights file scrip_joe_tc_diffgrid.nc
You can then copy that weights nc file to the Project folder
cp scrip_ scrip_joe_tc_diffgrid.nc ../../Projects/JOE_TC/Diffgrid

Step 6: Edit the Projects/JOE_TCd/coupling_joe_tc.in and use
SCRIP_WEIGHT_OPTION=1
Add the lines:
SCRIP_COAWST_NAME=Projects/JOE_TC/DiffGrid/scrip_ joetc_diffgrid.nc

Step 7: change the ocean and wave input file names to be
WAV_name = Projects/JOE_TC/DiffGrid/INPUT_JOE_TC_COARSE ! wave model
OCN_name = Projects/JOE_TC/DiffGrid /ocean_joe_tc_coarse.in ! ocean model

Step 8: Edit the ocean model configuration file.
Edit Projects/JOE_TC/DiffGrid /joe_tc.h and add the option
#define MCT_INTERP_WV2AT
#define MCT_INTERP_OC2AT
This will allow mct interpolation between the wave – atm models, and between the ocean – atm models.
This is set already to define case H which has those cpp options set.

Step 9: make, run the program

6.11. Rip_current test case.

This test case is provided as an example of how to setup a coupled simulation to study a rip current. As of version 750, it supersedes the previous distribution of coupling to RefDif, as described in Haas and Warner (2009). The current test case is coupling of ROMS+SWAN, and the application is described in detail in Kumar et al. (2012) in section 4.3 of that paper. Users can adapt the test case for other types of similar investigations.

The application is run by editing the coawst.bash file to set
export ROMS_APPLICATION=RIP_CURRENT

This case requires mpi and the MCT libraries because it is a coupled application. And the run files are at:
export MY_HEADER_DIR=${MY_PROJECT_DIR}/Projects/Rip_current
export MY_ANALYTICAL_DIR=${MY_PROJECT_DIR}/Projects/Rip_current

This is built like the other tests cases, using ./coawst.bash. The user needs to set the total number of processors for each of roms and swan in the coupling_rip_current.in file. Then you need to set the processor allocation in the ocean_rip_current.in. The model is set up to run for 1 hour of simulation time, with swan on a 5s time step and roms on a 1s time step. Coupling is every 5s. The model is run using:

mpiexec -np 4 ./coawstM.exe Projects/Rip_current/coupling_rip_current.in

You can certainly change the number of processors for your own system. Upon completion of the test case, you can use the included plot_rip_current.m matlab file to create the plot below.

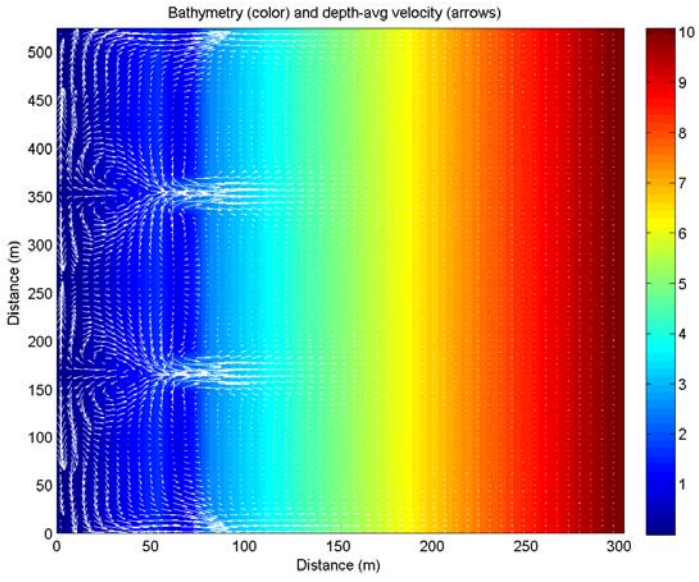


Figure 4. Rip current test case bathy and currents.

6.12 Hurricane Sandy = 2 grids for WRF, 2 grids for ROMS, 2 grids for SWAN.
This test case is used as the basis for the methodology in Sections 8 How to setup a WRF application, Section 9 How to set up a ROMS application, Section 10 how to set up a SWAN application, Section 12 how to setup a coupled application. These are step by step instructions that provide a general guideline to set up these system componenets.

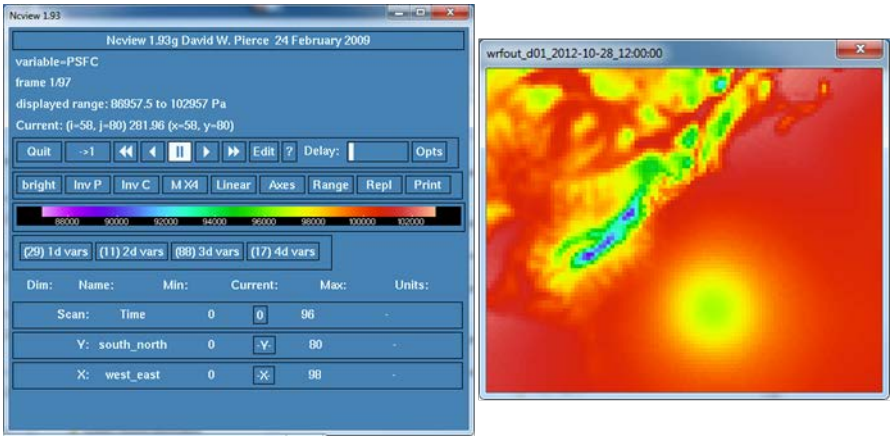


Figure 2: fPSFC of Hurricane Sandy along the US east coast.

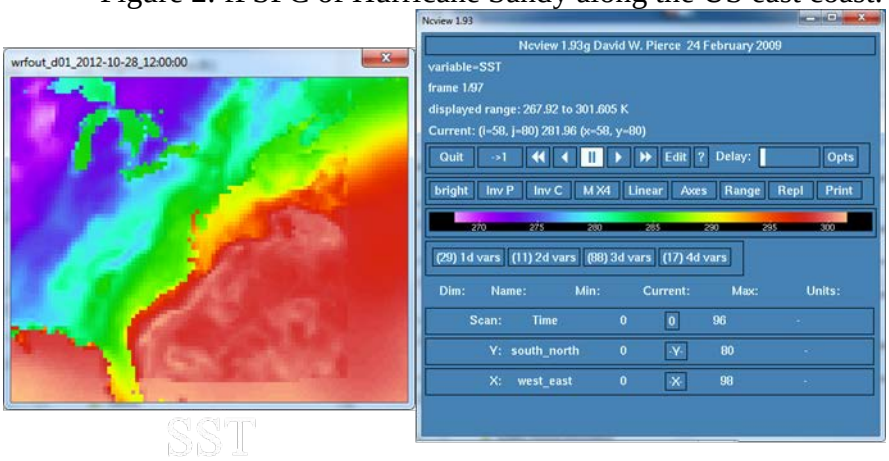
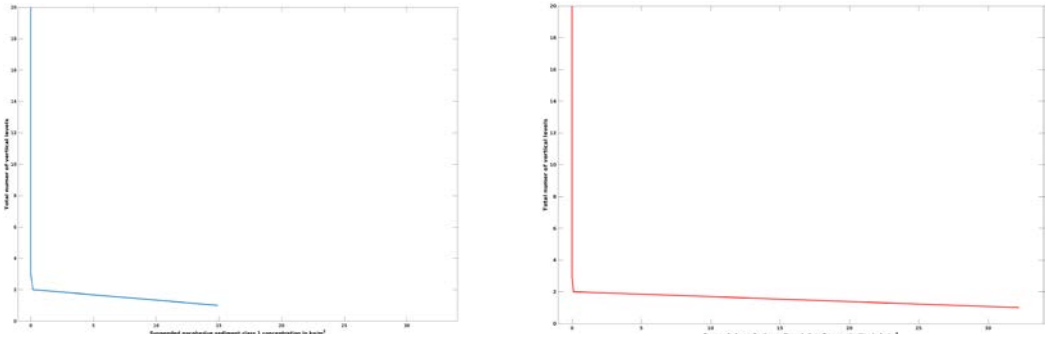


Figure 3: SST combined GFS and ROMS in WRF output.

6.13. Sed_floc_toy test case - ROMS only, tests sediment module with flocculation.
The test case is added to test the additional sediment module routines in COAWST that account for the aggregation, disaggregation of the cohesive sediment class in the water column. These processes are desribed as flocculation dynamics. This test case is still under development.

6.14 Sedbed_toy test case - ROMS only, tests sediment module.
This test case is formed to highlight the capabilities of the sediment module in COAWST framework to simulate various sediment transport processes (for instance sediment erosion, deposition) etc. It can incorporate both cohesive and non-cohesive sediment types. The details of the sediment module and related processes that can be handled are described in Warner et. al(2008). The application is run by editing the coawst.bash file to set:
export ROMS_APPLICATION=SEDBED_TOY

Figures 6.14 (a) and (b) illustrate the results at the end of simulation (). The results show the concentration for two non-cohesive sediment classes varying with the number of vertical levels.



(a) Non-cohesive sediment class 1 (b) Non-cohesive sediment class 2
Figure 6.14 Results from Sedbed_toy test case

6.15 Trench test case - ROMS only, test sediment morphology.12
The test case is based on the morphological evolution of bedload by simulating the experiemental work of Van Rijn (1993). The setup consists of a 30-m straight channel with a vertical trench in a mobile sand bed. More details are present in Warner et al. (2008). The application is run by editing the coawst.bash file:
export ROMS_APPLICATION=TRENCH
Figure 6.15 shows the bed thickness varying with along channel distance at the start and end of the simulation at 500 secs. The results are extracted from a cross channel plane of y=4.

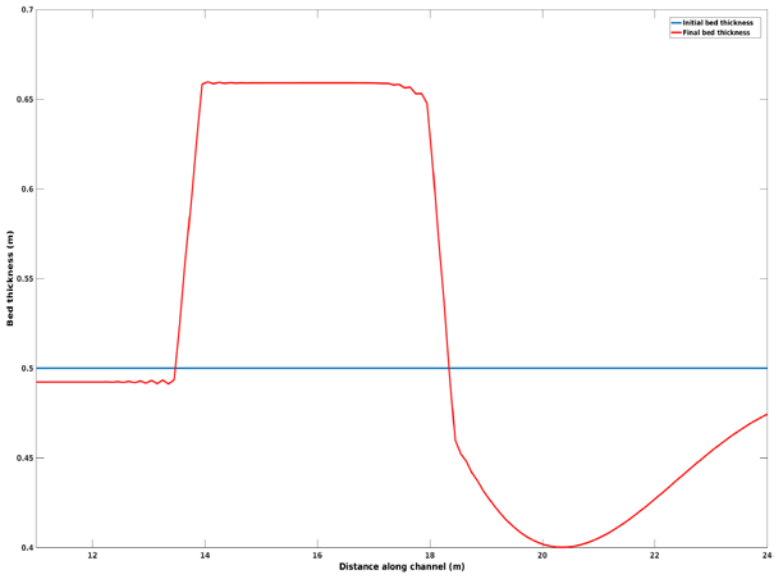


Figure 6.15 Results from Trench test case

6.16. Vegetation test case - ROMS+SWAN, test vegetation module.

This test case presents an idealized tidal and wave basin inhabited by a square patch of submerged aquatic vegetation (SAV). It accounts for the following hydrodynamic effects of vegetation in ROMS: sink of momentum, flexible plant reconfiguration, production and dissipation of turbulent kinetic energy. It also addresses wave damping by vegetation in SWAN and wave-current interactions around a vegetation patch (ROMS-SWAN). The details of this implementation are presented in Beudin et al.

Users can adapt the test case for other types of similar investigations. *All the steps described below have already been undertaken to setup a vegetation test case in the Veg_test folder* distributed with COAWST. The description of this test case is provided in the *veg_test_case.docx* file.

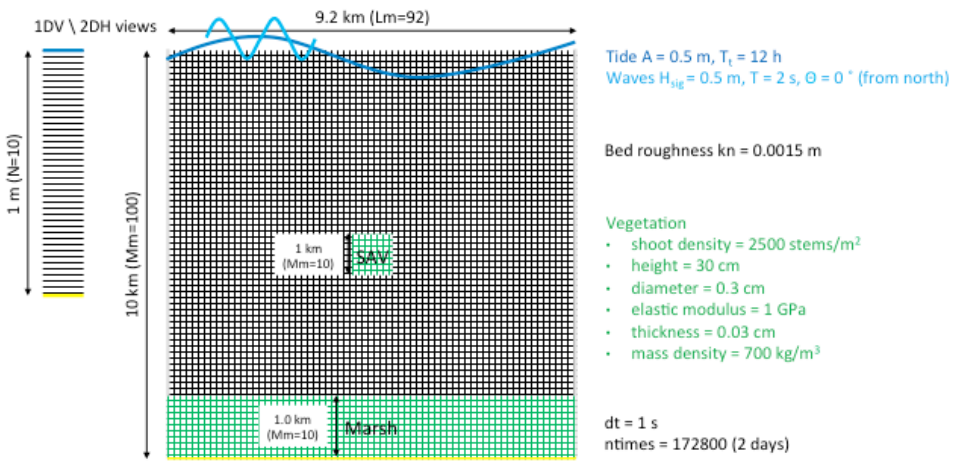


Figure 6.16 (a): Schematic showing vegetation and marsh domains

Step 1: Create roms grid and init files using
COAWST/Tools/mfiles/mtools/create_roms_netcdf_grid_file.m
COAWST/Tools/mfiles/mtools/create_roms_init.m - To setup vegetation properties in the intial file, enter NVEG. Currently plant density, height and diameter and the marsh_mask arrays are set to zero. The user can provide these values depending on the presence of the vegetation. Copy the grid and init file to the vegetation test folder.

Step 2: Create the SWAN grid file and bathymetry file and copy them to the vegetation folder.

Step 3: Edit coupling_veg_test.in.
NnodesWAV and NnodesOCN provide the number of processors for Swan and Roms codes respectively. Provide the path of Swan and Roms input file in WAV_NAME and OCN_name inputs

Step 4: Edit the ocean_veg_test.in for providing number of interior points in I and J directions (Lm, Mm), number of vertical layers. Enter the tiling inputs NtileI and NtileJ accordingly. Provide the name of roms grid and input file as generated in Step 1 above in GRDNAME and ININAME options.

Step 5: Edit the swan input file to enter the swan grid file and bathymetry file. Make sure to have the Keyword “NPLANTS” and “VEGETATION” properties correctly. The Vegetation properties should be consistent with the ROMS input file.

Step 6: Edit the “vegetation.in” to specify sptatially fixed vegetation properties along with the specifying logical options such as Hout(pdens)==T to output vegetation density in the output “.nc” file.

Step 7: Edit ana_fsobc.in for tidal inputs.

Step 8. Compile the code using ./coawst.bash (Check that the application is set to veg_test). Run the code by using the input file “coupling_veg_test.in”).

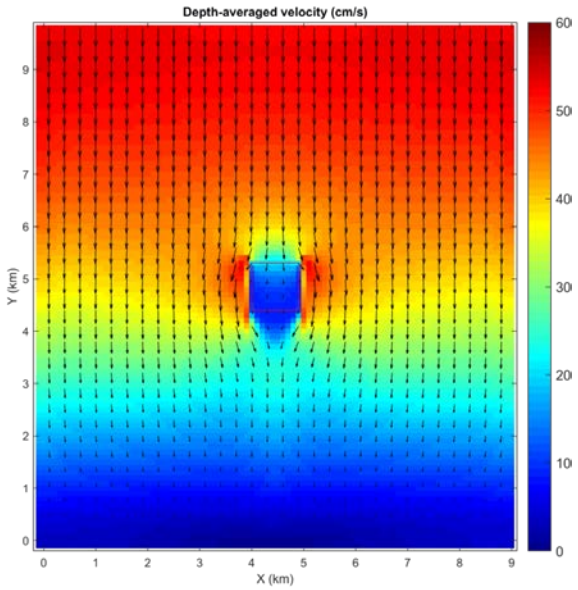
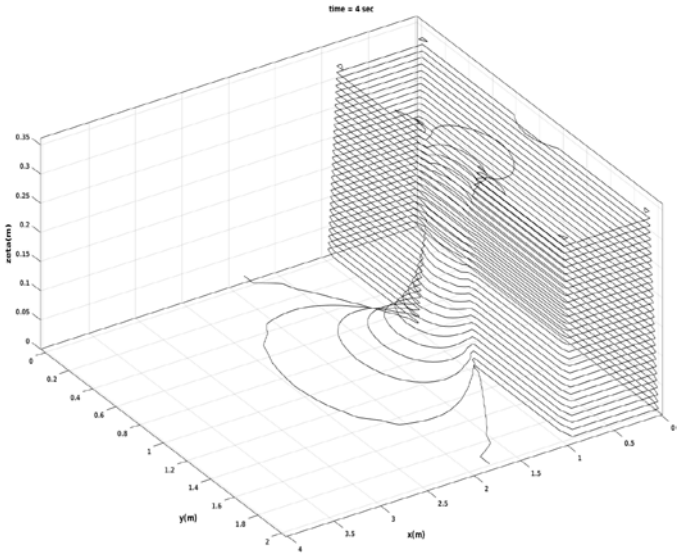


Figure 5: Figure generated from Veg_test case simulation showing depth averaged velocity (cm/s). Vegetation patch is in the outlined red box

6.17 Wetdry test case - ROMS only, test wetting/drying algorithms.

This case is provided to test the ability of COAWST framework to let the computational cells to get indundated/wet or dewatered/dry when total depth in a cell is less than a user defined critical value. This test case is consisting of a basin of rectangular water basin. The test case allows for water to spill through an opening to flood in an initially dry flood plain. More details of the methodology and test case is provided in Warner et. al (2013). Figures 6.17(a) shows the free surface level at 4 secs. Figure 6.17(b) shows the water levels at x=1.54m (40th plane), y=1.0m (14th plane) varying with time.



a) Water level at 4 secs

Figure 6.17 Wetdry test case

6.18 Sedbio_toy test case - ROMS, test coupled sediment transport-biogeochemistry module.

This test case highlights the coupled sediment transport-biogeochemistry modules in the COAWST framework. This model accounts for sediment transport processes (e.g. seabed erosion, deposition), as well as seabed and water column biogeochemical processes including particulate organic matter remineralization, and oxidation of reduced chemical species. The coupling between the sediment transport and biogeochemical modules accounts for resuspension and deposition of particulate organic matter; exchanges of porewater across the seabed-water interface due to diffusion, erosion, and deposition; diffusion of porewater and particulate organic matter in the seabed; and biogeochemical processes in the seabed based on Moriarty et al. (2017, 2018). Additionally, seabed organic matter may affect the properties of the seabed (e.g. seabed layer thickness and mass).

The test case uses a quasi-one-dimensional modeling approach, similar to sedbed_toy test, with a uniform 5-cell by 6-cell grid. To run the model, edit the ocean_sedbio_toy.in, sediment_sedbio_toy.in, and bio_sedbio_toy.in based on user preferences. For example, users may want to edit sediment parameters in sediment_sedbio_toy.h. Users may also want to edit parameters and parameterizations in portions of the code such as ana_wwave.h, sedtr_reactions_pom.F, or other files. Compile the code using ./coawst.bash with COAWST_APPLICATION set to SEDBIO_TOY). Run the code by using the input file “ocean_sedbio_toy.in”. An example matlab script for

analyzing model output, plot_sedbgc_in_COAWST.m, is provided, and a figure of the model is below.

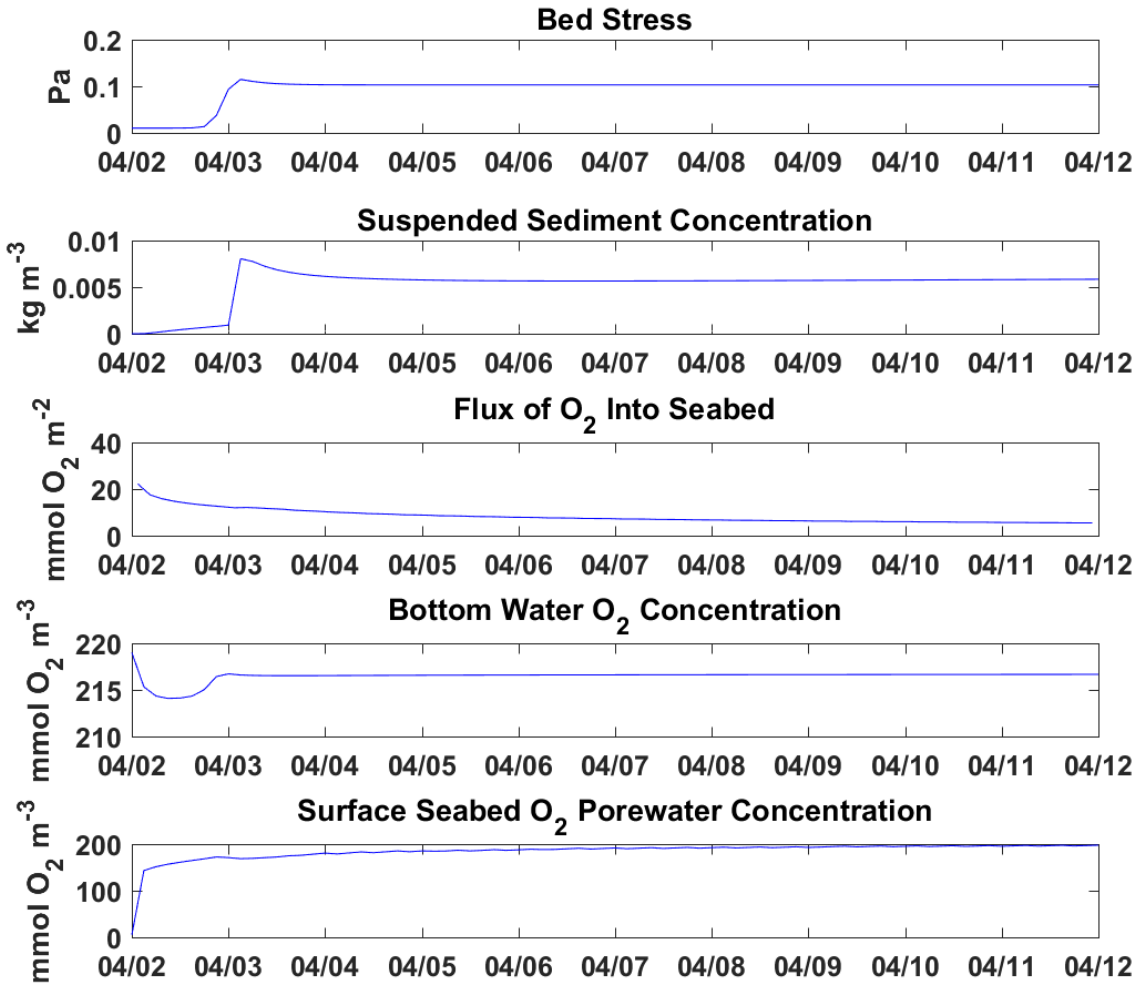


Figure 6.18: Time-series of model utput generated from Sedbio_toy simulation

6.19 Veg_growth test case - ROMS+vegetation+sediment+water-column biogeochemistry (including spectral light module)

This test case is implementated in an idealized domain where the introduction of sediment in the water column (SSC) at one end of the domain provides sub-optimal light conditions that causes SAV dieback in that region. The model is based upon previous SAV models published by Madden and Kemp (1996) and Cerco and Moore (2001). The module is built to interact dynamically with the water-column biogeochemistry module *bio_fennel*, a bio-optical model, and the sediment transport model and this coupled package is called *estuarybgc*. Based on the change in SAV biomass (above ground,

below ground) and epiphyte biomass, SAV density and height evolve in time and space and directly couple to three-dimensional water-column biogeochemical, hydrodynamic, and sediment transport models. SAV biomass is computed from temperature, nutrient loading and light predictions obtained from coupled hydrodynamics (temperature), bio-geochemistry (nutrients) and bio-optical (light) models. In exchange, the growth of SAV sequesters or contributes nutrients from the water column and sediment layers. The presence of SAV modulates current and wave attenuation and consequently affects modelled sediment transport and fate. The resulting modelling framework provides a two-way coupled SAV-biogeochemistry-hydrodynamic and morphodynamic model. This allows for the simulation of the dynamic growth and mortality of SAV in coastal environments in response to changes in light and nutrient availability, including SAV impacts on sediment transport and nutrient, carbon, and oxygen cycling. More details of the modeling methodology are presented in Kalra et al., 2019 (In review).

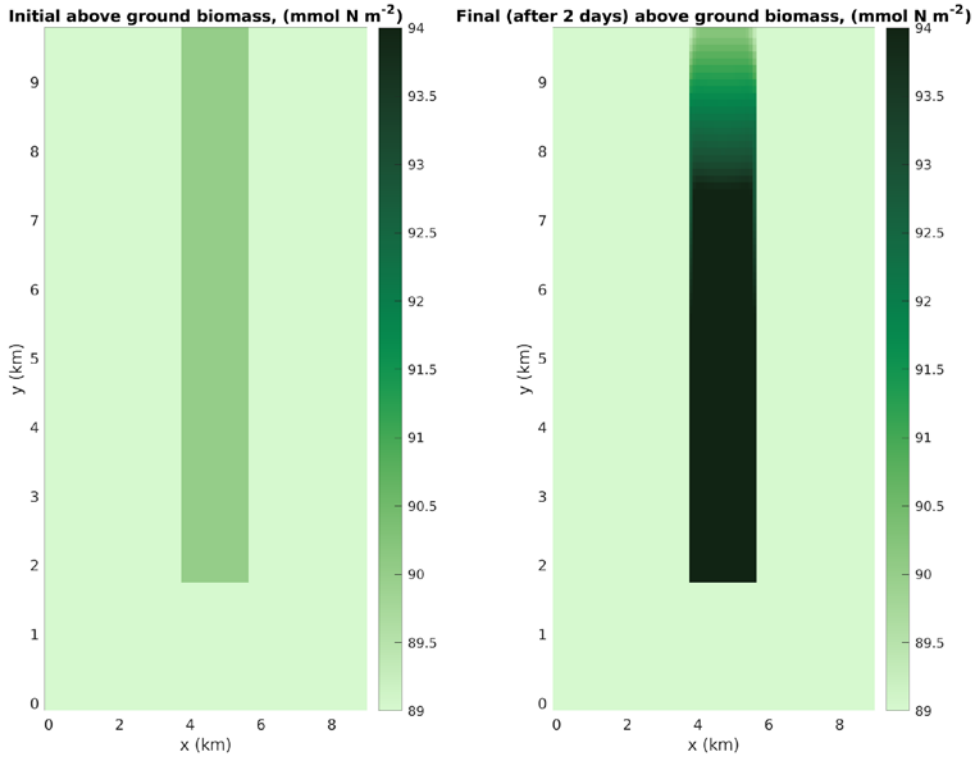


Figure 6.19. Above ground biomass (initial) and final at the end of 2 days.

6.20 Marsh_test – ROMS+ marsh dynamics

This test case is used to verify two marsh dynamics processes: horizontal retreat due to lateral wave thrust on marsh edge and vertical accretion due to biomass production and sediment trapping. The former requires an input of wave forcing and latter is based on a parabolic formulation to get biomass. The lateral wave thrust action on the marsh edge leads to an export of sediment from marsh cells to non-marsh cells. It is done through an exchange of bed mass. This leads to a change in vertical bed thickness and an evolution

of bathymetry (allocothonous changes). Note that, there can be an import of sediment on marsh cells through the suspended sediment in water column. The model also accounts vertical accretion of marsh platform through below ground biomass production (autochthonous supply). The dynamic biomass production can modify the emergent marsh vegetation properties (density, height, diameter) and that in return, can affect the hydrodynamics and sediment dynamics (Kalra et al., 2020, in preparation).

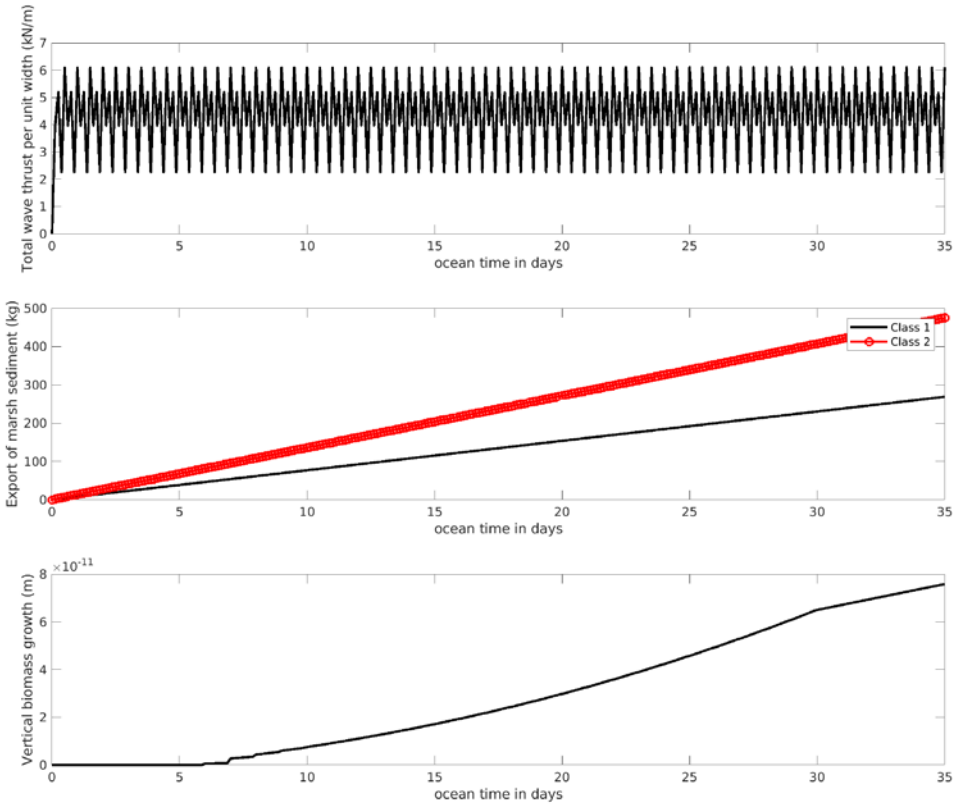


Figure 6.20. Marsh dynamics test case after 35 days of simulation.

Section 7. Visualization Tools.

NCVIEW
Source code at:
http://meteora.ucsd.edu/~pierce/ncview_home_page.html

Some new colormaps at:
https://www.myroms.org/forum/viewtopic.php?f=31&t=1930&p=6926#p6926

VAPORGUI
this from the WRF visualization world.

http://www.vapor.ucar.edu/docs/install/index.php?id=unixbin

Section 8. Setting up a WRF application.

This section describes how to set up a WRF (only) application. Some of this was taken from the WRF ARW User's manual. It is recommended for you to do the online wrf tutorial (http://www2.mmm.ucar.edu/wrf/OnLineTutorial/). These instructions are directed towards the Sandy test case. Specific steps for your own application may differ.

1. Building the WRF Code

You can use our distributed WRF and WPS programs to create the WRF forcings and/or to run WRF by itself.

• Make a new directory and copy the COAWST code into that dir, and cd into that dir.

• You will always need a Projects header file. I recommend that you create a directory called "My_Project" (create your own name, dont actually use 'My_Project'). The directory could be: COAWST/Projects/My_Project. In that directory, create a header file (basic text file) such as: my_project.h. For a wrf only application, you would need to have in that file:

#define WRF_MODEL

• edit coawst.bash and set the Project name, header file location, and build location.

• build the model with

./coawst.bash

choose one of the WRF build options depending on your system (we use 15 for Linux dmpar). There are two choices for parallel applications: dmapr = distributed memory parallel or smpar = shared memory parallel. If WRF is to be coupled, then you have to use a dmpar option, do not use shared memory for a coupled application. For WRF by itself you can choose serial or parallel (dmpar or smpar).

• Then choose then 1 for basic nesting (For the Sandy example we start with basic 1).

• When it is done, at the root dir you should see a link of coawstM to WRF/main/wrf.exe

```
>ls -ltr
...
drwxrwxr-x  2 jwarner domain users     4096 Jan 31 10:40 Build
drwxrwxr-x 21 jwarner domain users     4096 Jan 31 10:41 WRF
lrwxrwxrwx  1 jwarner domain users       16 Jan 31 11:16 coawstM ->
WRF/main/wrf.exe
```

Also list the contents of WRF/main: ls WRF/main/*.exe and you should see ndown.exe, real.exe, and wrf.exe

2. Building the WPS Code

Building WPS requires that WRF is already built. WPS is distributed with the code.

• cd COAWST/WPS

• ./configure

• choose one of the options, usually a serial build is ok for a first test with a smaller grid.

• ./compile

• ls -ltr *.exe and you should see geogrid.exe, ungrib.exe, and metgrid.exe

• ls -ltr util/*.exe and you should see a number of utility executables: avg_tsfc.exe, g1print.exe, g2print.exe, mod_levs.exe, plotfmt.exe, plotgrid.exe, and rd_intermediate.exe

3. geogrid: Creating a Grid

3a) edit WPS/namelist.wps. We distribute a copy for Sandy as: COAWST/Projects/Sandy/namelist.wps. You can copy that file to COAWST/WPS. For this step we focus on the &share and the &geogrid parts, modified the H Sandy test case to be:

namelist.wps

```
&share
 wrf_core = 'ARW',
 max_dom = 2,
 start_date = '2012-10-28_12:00:00','2012-10-28_12:00:00',
 end_date   = '2012-10-30_12:00:00','2012-10-30_12:00:00',
 interval_seconds = 21600
 io_form_geogrid = 2,
/

&geogrid
 parent_id         =   1,   1,
 parent_grid_ratio =   1,   3,
 i_parent_start    =   1,  33,
 j_parent_start    =   1,   8,
 e_we              =  85, 100,
 e_sn              =  82, 100,
 !
 !!!!!!!!!!!!!!!!!!!!!!!!!!!! IMPORTANT NOTE !!!!!!!!!!!!!!!!!!!!!!!!!!!!
 ! The default datasets used to produce the MAXSNOALB and ALBEDO12M
 ! fields have changed in WPS v4.0. These fields are now interpolated
 ! from MODIS-based datasets.
 !
 ! To match the output given by the default namelist.wps in WPS v3.9.1,
 ! the following setting for geog_data_res may be used:
 !
 ! geog_data_res = 'maxsnowalb_ncep+albedo_ncep+default',
'maxsnowalb_ncep+albedo_ncep+default',
 !
 !!!!!!!!!!!!!!!!!!!!!!!!!!!! IMPORTANT NOTE !!!!!!!!!!!!!!!!!!!!!!!!!!!!
 !
 geog_data_res = 'default','default',
 dx = 30000,
 dy = 30000,
 map_proj = 'lambert',
 ref_lat   =  37.50,
 ref_lon   = -75.00,
 truelat1  =  30.0,
 truelat2  =  60.0,
 stand_lon = -75.0,
 geog_data_path = '/vortexfs1/scratch/jwarner/WPS/geog_data'
/

&ungrib
 out_format = 'WPS',
 prefix = 'NAM',
/

&metgrid
 fg_name = 'NAM', ‘RTG’
 io_form_metgrid = 2,
/
```

3b) get the geogrid data from this page:
http://www2.mmm.ucar.edu/wrf/users/download/get_sources_wps_geog.html
I chose the geog_high_res_mandatory.tar.gz. Then untar this file in a directory
tar -xvf geog_high_res_mandatory.tar.gz
Set the geog_data_path in the namelist.wps file to this directory.

3c) make sure there is a link to the correct geogrid table.
> ls –ltr WPS/geogrid/*.TBL
should return
> geogrid/GEOGRID.TBL -> GEOGRID.TBL.ARW

3d) run geogrid
>WPS ./geogrid.exe
and get back a complete successful information.

3e) ls –ltr should show geo_em.d01.nc and geo_em.d02.nc files. Use ncview or some other viewer to check it out. Also look in geogrid.log.

4. ungrib: Getting IC and BC data
4a) Get grib data for initial and boundary conditions. You can start here:
http://www2.mmm.ucar.edu/wrf/users/download/free_data.html
I chose to use NCEP 12km data here: https://rda.ucar.edu/datasets/ds609.0/
You will need to register first and get a username/passwd. Then use direct download or one of the other options for full data files from 2012-10-27 to 2012-11-01. Copy the files to a new directory. The files were:
20121027.nam.t00z.awphys00.grb2.tm00
…..
20121101.nam.t18z.awphys00.grb2.tm00

4b) This data is in grib2 format. We want to convert that format to something that WRF can read so we use the ungrib.exe program. cd to COAWST/WPS
link a Vtable to the NAM data with:
>ln –sf ungrib/Variable_Tables/Vtable.NAM Vtable

4c) link the NAM files (from step 1) to common names that WPS will recognize. For example:
> ./link_grid.csh /WPS/path_the _files_are_in/2012*.tm00
Do an ls -ltr and see all the GRIBFILE.AAA etc ….linking to all the gfs data files.

4d) Run ungrib
> ./ungrib.exe >& ungrib.out &
Edit the .out file and see if there were any problems. do an ls and see files:
NAM:2012-10-29_00 up to NAM:2012-10-30_06

4e) Get SST data
Go here for SST data: ftp://polar.ncep.noaa.gov/pub/history/sst ← サービス終了で使えない
for the times that you want, and put into a folder.
I used:
ftp://polar.ncep.noaa.gov/pub/pub/history/sst/rtg_high_res/
rtg_sst_grb_hr_0.083.201210.gz and rtg_sst_grb_hr_0.083.201211.gz

4f) link a Vtable using WPS/ ln -sf ungrib/Variable_Tables/**Vtable.SST** Vtable

4g) remove links to nam gribfiles using WPS/ rm GRIBFILE*
Then link to the sst data using WPS/ ./link_grib.csh WPS/ path_the _files_are_in/rtg_sst_grb_hr*

4h) edit COAWST/WPS/namelist.wps and change
prefix = 'NAM'
to
prefix = 'RTG'

4i) COAWST/WPS/ ./ungrib.exe >& ungrib_sst.out &
edit the .out file and see that all went well. do an ls and see files:
RTG:2012-10-28_00 up to NAM:2012-10-31_00

5. metgrid: Interpolate ungribbed met data to the grid.
5a) Check the the metgrid table is pointing to the right place.
> > ls –ltr WPS/metgrid
should show METGRID.TBL -> METGRID.TBL.ARW

SST 5b) because we acquired sst data, then edit
WPS/namelist.wps and change the fg_name to
fg_name = 'NAM', 'RTG'

5c) run metgrid
in WPS/ run ./metgrid.exe
As it runs you see …. processing NAM, RTG, …
When done check that the met files were made. Do an ls –ltr and see the met_em.d01** files.
met_em.d01.2012-10-28_12:00:00.nc … met_em.d01.2012-10-30_12:00:00.nc
met_em.d02.2012-10-28_12:00:00.nc … met_em.d02.2012-10-30_12:00:00.nc

6. real.exe to create the init and BC files.
Now we need to run the real program.
6a) >cd WRF/test/em_real
edit the namelist file. I have:

```
&time_control
 run_days                            = 0,
 run_hours                           = 48,
 run_minutes                         = 0,
 run_seconds                         = 0,
 start_year                          = 2012, 2012, 2012,
 start_month                         = 10,   10,   10,
 start_day                           = 28,   28,   28,
 start_hour                          = 12,   12,   12,
 start_minute                        = 00,   00,   00,
 start_second                        = 00,   00,   00,
 end_year                            = 2012, 2012, 2012,
 end_month                           = 10,   10,   10,
```

```
end_day                             = 30,   30,   30,
end_hour                            = 12,   12,   12,
end_minute                          = 00,   00,   00,
end_second                          = 00,   00,   00,
interval_seconds                    = 21600
input_from_file                     = .true.,.true.,.false.,
history_interval                    = 30,   30,   60,
frames_per_outfile                  = 1000, 1000, 1000,
restart                             = .false.,
restart_interval                    = 5000,
io_form_history                     = 2
io_form_restart                     = 2
io_form_input                       = 2
io_form_boundary                    = 2
debug_level                         = 0
auxinput4_inname                    = "wrflowinp_d<domain>"
auxinput4_interval                  = 360, 360, 360,
io_form_auxinput4                   = 2
/

&domains
time_step                           = 180,
time_step_fract_num                 = 0,
time_step_fract_den                 = 1,
max_dom                             = 2,
e_we                                = 85,    100,   94,
e_sn                                = 82,    100,   91,
e_vert                              = 48,    48,    48,
p_top_requested                     = 5000,
num_metgrid_levels                  = 40,
num_metgrid_soil_levels             = 4,
dx                                  = 30000, 10000,  3333.33,
dy                                  = 30000, 10000,  3333.33,
grid_id                             = 1,     2,     3,
parent_id                           = 0,     1,     2,
i_parent_start                      = 1,     33,    30,
j_parent_start                      = 1,     8,     30,
parent_grid_ratio                   = 1,     3,     3,
parent_time_step_ratio              = 1,     3,     3,
feedback                            = 1,
smooth_option                       = 0
nproc_x                             = 1
nproc_y                             = 1
/

&physics
mp_physics                          = 6,     6,     6,
cu_physics                          = 1,     1,     0,
ra_lw_physics                       = 1,     1,     1,
ra_sw_physics                       = 1,     1,     1,
bl_pbl_physics                      = 1,     1,     1,
sf_sfclay_physics                   = 1,     1,     1,
sf_surface_physics                  = 2,     2,     2,
radt                                = 30,    30,    30,
bldt                                = 0,     0,     0,
cudt                                = 5,     5,     5,
icloud                              = 1,
```

```
 surface_input_source                = 1,
 num_soil_layers                     = 4,
 num_land_cat                        = 21,
 sf_urban_physics                    = 0,     0,     0,
 sst_update                          = 1,
 /

 &fdda
 /

 &dynamics
 hybrid_opt                          = 2,
 w_damping                           = 0,
 diff_opt                            = 1,       1,       1,
 km_opt                              = 4,       4,       4,
 diff_6th_opt                        = 0,       0,       0,
 diff_6th_factor                     = 0.12,    0.12,    0.12,
 base_temp                           = 290.
 damp_opt                            = 0,
 zdamp                               = 5000.,   5000.,   5000.,
 dampcoef                            = 0.2,     0.2,     0.2
 khdif                               = 0,       0,       0,
 kvdif                               = 0,       0,       0,
 non_hydrostatic                     = .true., .true., .true.,
 moist_adv_opt                       = 1,       1,       1,
 scalar_adv_opt                      = 1,       1,       1,
 gwd_opt                             = 1,
 /

 &bdy_control
 spec_bdy_width                      = 5,
 spec_zone                           = 1,
 relax_zone                          = 4,
 specified                           = .true., .false.,.false.,
 nested                              = .false., .true., .true.,
 /

 &grib2
 /

 &namelist_quilt
 nio_tasks_per_group = 0,
 nio_groups = 1,
 /
```

Pay attention to time and e_* settings.

6b) cd to WRF/test/em_real and link the met files to here.
> ln –sf /raid1/jcwarner/Models/WRF/WPS/met_em.d01.2003-09* .

6c) run the real program
edit namelist.input and set nproc_x=1, nproc_y=1
run ./real.exe
When done, check to see that it made wrfinput_d01 and wrfbdy_d01 netcdf files.

Edit the rsl.error* and rsl.out* files to see that all went well.
I then removed the met* files.

7. run WRF.
Now we need to run the wrf program. This was already compiled at the start of this procedure. So you should have a coawsM. Make sure you copy several files to the COAWST root directory:
(these files need to be at the same place as the executable)
cp WRF/test/em_real/namelist.input COAWST (this is the root location)
cp all the wrfbdy*, wrfinput*, and wrflow* files to the root location
edit the namelist.input and set nproc_x=NX, nproc_y=NY (your choice here for NX NY)

> ./sbatch run_poseidon with something like:

```
#!/bin/bash
#PBS -N wrf1
#PBS -l nodes=1:ppn=36,walltime=120:00:00
#PBS -q standard

NPROCS=`wc -l < $PBS_NODEFILE`
cd /raid1/jcwarner/Models/COAWST
mpirun -np 16 -machinefile $PBS_NODEFILE ./coawstM > wrf_run1.out
```

So for this example it used np 40 cores, and this number should = NX*NY in your namelist.input

8. look at the output.
ncview wrfout_d01_***

9. For a WRF moving nest.
To run WRF with a moving nest, you need to:
- compile COAWST (WRF) with a moving nest option 3
- change the namelist.input lines from:
input_from_file = .true.,.true.,.false.,
auxinput4_inname = "wrflowinp_d<domain>"
auxinput4_interval = 360, 360, 360,
to:
input_from_file = .true.,.false.,.false.,
auxinput4_inname = "wrflowinp_d01"
auxinput4_interval = 360, 0, 0,
- and then run as before.
- look at the output and the wrfout_d02* should be different with a moving location.

←デフォルトの設定では moving になし.
Slack 2020/7/10 参照

ROMS

Section 9. Setting up a ROMS application.

There are many ways to set up a ROMS application, dependent on what physcis you are looking at, the domain size, nesting, etc. This is just one example. We will continue the Sandy example here to set up the ROMS grids (2 grids nested).

1. ROMS grid

There are many ways to make a roms grid, such as
- Gridgen: http://code.google.com/p/gridgen-c
- Easygrid: https://www.myroms.org/wiki/easygrid ← 使用者多い. ROMS-wiki の標準
- Gridbuilder: http://austides.com/downloads
- COAWST Tools: wrf2roms_mw.m, create_roms_xygrid.,
- many othere tools out there.

スクリプト create_roms_grid.m とする

We will use the wrf grid to get the outline, but with a different resolution. These steps use Matlab. Load the wrf grid to get coastline data, cd to Projects/Sandy and use:

Matlab

mtools/

```
netcdf_load('wrfinput_d01')
figure ← Figureウィンドウ作成
pcolorjw(XLONG,XLAT,double(1-LANDMASK)) ← プロットする
hold on ← 図を重ねる
title('WRF LANDMASK grid, Lambert conformal proj')
xlabel('longitude'); ylabel('latitiude')
```

午前 ～ ↓

Training (火曜日午後) ビデオ で コードの全体の様子が分かる.

coawst/Projects/sandy/ に存在
(注) WRF training で共通ファイルを作成。
coawst/WRF/test/em_real/ に存在。

Matlab スクリプトは coawst/Tools/mfiles/ に存在。

Pick 4 corners for roms parent grid

```
xl=-82; xr=-65;
yb= 28; yt= 43;
```

Pick number of points in the grid

```
numx=86; numy=64;
```

Make a matrix of the lons and lats.

```
dx=(xr-xl)/numx; dy=(yt-yb)/numy;
[lon, lat]=meshgrid(xl:dx:xr, yb:dy:yt);
lon=lon.';lat=lat.';        .' は転置
plot(lon,lat,'k-')
plot(lon',lat','k-')        複素共役転置
text(-75,27,'- - - roms grid')
```

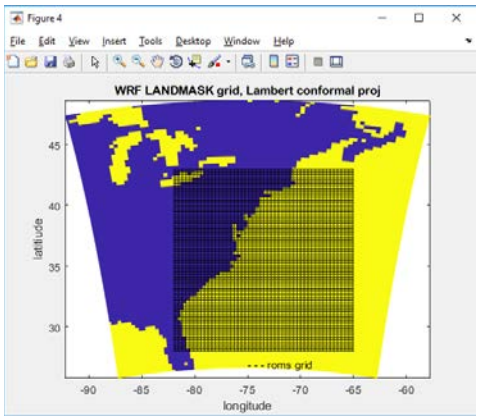


Figure. First steps to show WRF and ROMS grids.

注意 coawst/Tools/mfiles/mtools/ に存在する mat2roms_mw.m の 95, 96 行目の % を削除し、97, 98 行目の先頭に % を付ける. これをしないと次ページの eval で エラー となる

(注) wrfinput_d01 は 自前で作成した ものでも 大丈夫 だった.

新規スクリプトの作り方
1. 「新規スクリプト」ボタンをクリック
2. 開いたエディター で「保存」ボタンの下向き矢印をクリックし、「名前を付けて保存」する
3. コードを入力する (適宜保存する)
4. 「実行」ボタンをクリック する.

Call generic grid creation.

```
roms_grid='Sandy_roms_grid.nc';
```

p57～58 の コード を create_roms_grid.m として 作成 する。
Tools/UTokyo/ に コピー

```
rho.lat=lat; rho.lon=lon;
rho.depth=zeros(size(rho.lon))+100;  % for now just make zeros
rho.mask=zeros(size(rho.lon));       % for now just make zeros
spherical='T';
%projection='lambert conformal conic';
projection='mercator';
save temp_jcw33.mat rho spherical projection
eval(['mat2roms_mw(''temp_jcw33.mat'',''',roms_grid,''');'])
!del temp_jcw33.mat
%User needs to edit roms variables
disp(['     '])
disp(['Created roms grid -->   ',roms_grid])
disp(['     '])
disp(['You need to edit depth in ',roms_grid])
disp(['     '])
```

2. ROMS grid masking

Start out with the WRF mask.

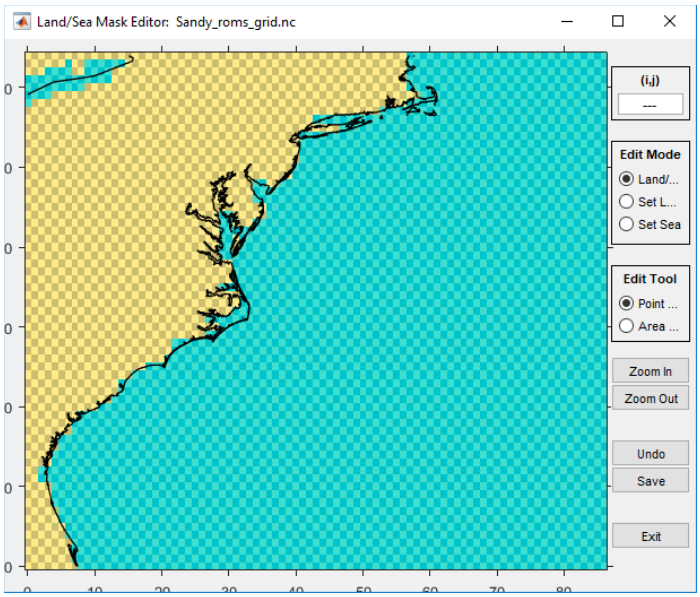
```
F = scatteredInterpolant(double(XLONG(:)),double(XLAT(:)),
double(1-LANDMASK(:)),'nearest');
roms_mask=F(lon,lat);
figure
pcolorjw(lon,lat,roms_mask)
title('ROMS 1-LANDMASK grid, Mercator proj')
xlabel('longitude'); ylabel('latitiude')
```

Compute mask on rho, u, v, and psi points.

```
water = double(roms_mask);
u_mask = water(1:end-1,:) & water(2:end,:);
v_mask= water(:,1:end-1) & water(:,2:end);
psi_mask= water(1:end-1,1:end-1) & water(1:end-1,2:end) &
water(2:end,1:end-1) & water(2:end,2:end);
ncwrite('Sandy_roms_grid.nc','mask_rho',roms_mask);
ncwrite('Sandy_roms_grid.nc','mask_u',double(u_mask));
ncwrite('Sandy_roms_grid.nc','mask_v',double(v_mask));
ncwrite('Sandy_roms_grid.nc','mask_psi',double(psi_mask));
```

Now we want to make it nicer by using editmask. We need a coastline first.
You can use GEODAS software ~~http://www.ngdc.noaa.gov/mgg/geodas/geodas.html~~
I selected lat bounds of 45 / 25 lon of -84 / -60 and saved this as coastline.mat

```
editmask('Sandy_roms_grid.nc','coastline.mat');
```



Editmask for Sandy_roms_grid

最後に Save をクリックすると
Sandy_roms_grid.nc が更新され、
Sandy_roms_grid_ijcst.mat
が作成される。
Exit をクリックして終了する

3. Bathymetry create_roms_bathymetry.m を作成する → Tools/UTokyo/ にコピー

Bathymetry can come from many different places. You can use a source like this:
http://www.ngdc.noaa.gov/mgg/global/global.html
For this application, we use data from a local file

```
load USeast_bathy.mat                    ← coast/Projects/Sandy/ に存在する
netcdf_load('Sandy_roms_grid.nc')
h=griddata(h_lon,h_lat,h_USeast,lon_rho,lat_rho);
h(isnan(h))=5;
%smooth h a little
h(2:end-1,2:end-1)=0.2*(h(1:end-2,2:end-1)+h(2:end-1,2:end-
1)+h(3:end,2:end-1)+h(2:end-1,1:end-2)+h(2:end-1,3:end));

figure
pcolorjw(lon_rho,lat_rho,h)
hold on
load coastline.mat
plot(lon,lat,'k')
caxis([5 2500]); colorbar
title('ROMS bathy')
xlabel('longitude'); ylabel('latitiude')
ncwrite('Sandy_roms_grid.nc','h',h);
```

4. Roms child grid create_roms_child_grid.m

To create a nested child grid, we want to see how the 2WRF and 2 ROMS grids will overlay.

```
netcdf_load('wrfinput_d01')
figure
pcolorjw(XLONG,XLAT,double(1-LANDMASK))
hold on
netcdf_load('wrfinput_d02')
```

```
pcolorjw(XLONG,XLAT,double(1-LANDMASK))
plot(XLONG(1,:),XLAT(1,:),'r');
plot(XLONG(end,:),XLAT(end,:),'r')
plot(XLONG(:,1),XLAT(:,1),'r');
plot(XLONG(:,end),XLAT(:,end),'r')
% plot roms parent grid
netcdf_load('Sandy_roms_grid.nc');
plot(lon_rho(1,:),lat_rho(1,:),'k');
plot(lon_rho(end,:),lat_rho(end,:),'k')
plot(lon_rho(:,1),lat_rho(:,1),'k');
plot(lon_rho(:,end),lat_rho(:,end),'k')
text(-75,29,'roms parent grid')
text(-77,27,'wrf parent grid')
text(-77.2,34,'wrf child grid')
title('LANDMASKS')
xlabel('longitude'); ylabel('latitiude')
```

Select child indices and plot location of roms child grid.

```
Istr=22; Iend=60; Jstr=26; Jend=54;
plot(lon_rho(Istr,Jstr),lat_rho(Istr,Jstr),'m+')
plot(lon_rho(Istr,Jend),lat_rho(Istr,Jend),'m+')
plot(lon_rho(Iend,Jstr),lat_rho(Iend,Jstr),'m+')
plot(lon_rho(Iend,Jend),lat_rho(Iend,Jend),'m+')
ref_ratio=3;
roms_child_grid='Sandy_roms_grid_ref3.nc';
F=coarse2fine('Sandy_roms_grid.nc','Sandy_roms_grid_ref3.nc', ...
              ref_ratio,Istr,Iend,Jstr,Jend);
Gnames={'Sandy_roms_grid.nc','Sandy_roms_grid_ref3.nc'};
[S,G]=contact(Gnames,'Sandy_roms_contact.nc');
```

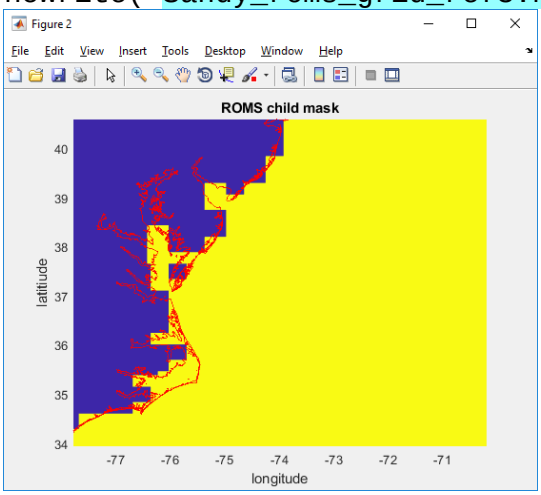
Compute bathy for the child.

```
netcdf_load('Sandy_roms_grid_ref3.nc')
load USeast_bathy.mat
h=griddata(h_lon,h_lat,h_USeast,lon_rho,lat_rho);
h(isnan(h))=5;
h(2:end-1,2:end-1)=0.2*(h(1:end-2,2:end-1)+h(2:end-1,2:end-
1)+h(3:end,2:end-1)+ ...
                        h(2:end-1,1:end-2)+h(2:end-1,3:end));
figure
pcolorjw(lon_rho,lat_rho,h)
hold on
load coastline.mat
plot(lon,lat,'r')
caxis([5 2500]); colorbar
title('ROMS bathy')
xlabel('longitude'); ylabel('latitiude')
ncwrite('Sandy_roms_grid_ref3.nc','h',h);
```

Recompute child mask based on WRF mask

```
netcdf_load('wrfinput_d01');
F = TriScatteredInterp(double(XLONG(:)),double(XLAT(:)), ...
```

```
                        double(1-LANDMASK(:)),'nearest');
roms_mask=F(lon_rho,lat_rho);
figure
pcolorjw(lon_rho,lat_rho,roms_mask)
title('ROMS child mask')
xlabel('longitude'); ylabel('latitiude')
hold on
plot(lon,lat,'r')
water = double(roms_mask);
u_mask = water(1:end-1,:) & water(2:end,:);
v_mask= water(:,1:end-1) & water(:,2:end);
psi_mask= water(1:end-1,1:end-1) & water(1:end-1,2:end) &
water(2:end,1:end-1) & water(2:end,2:end);
ncwrite('Sandy_roms_grid_ref3.nc','mask_rho',roms_mask);
ncwrite('Sandy_roms_grid_ref3.nc','mask_u',double(u_mask));
ncwrite('Sandy_roms_grid_ref3.nc','mask_v',double(v_mask));
ncwrite('Sandy_roms_grid_ref3.nc','mask_psi',double(psi_mask));
```



editmask('Sandy_roms_grid_ref3.nc', 'coastline.mat');

ROMS child grid mask. You can use editmask again to make this nicer!

climatology は nudging用

mfiles

5. 3D BC's, IC's, and Climatology fields. 作成 1時間1分～

This step creates roms init conditions, bc's, and nudging files. There are several tools out there. Here we will use the Tools folder and use Tools/mfiles/~~mtools~~/roms_master_climatology_coawst_mw → coawst/Projects/Sandy/training/ ヘコピー済

This m file will create BC, IC, and Climatology for the parent grid. Here is the header information that you can edit:

```
%%%   START OF USER INPUT  %%%%%%
% (1) Enter start date (T1) and number of days to get climatology
data
T1 = datenum(2012,10,28,12,0,0); %start date
%number of days and frequency to create climatology files for
numdays = 5;
dayFrequency = 1;

% (2) Enter URL of the HYCOM catalog for the requested time, T1
```

```
%     see http://tds.hycom.org/thredds/catalog.html
url =
'http://tds.hycom.org/thredds/dodsC/GLBa0.08/expt_90.9';      %
2011-01 to 2013-08

% (3) Enter working directory (wdr)
wdr =
'F:\data\models\COAWST_tests\coawstv3.4_update\coawst_v3.4_tests\
sandy\Projects\Sandy';

% (4) Enter path and name of the ROMS grid
modelgrid = 'Sandy_roms_grid.nc'

% (5) Enter grid vertical coordinate parameters --These need to
be consistent with the ROMS setup.
theta_s     = 5;
theta_b     = 0.4;
Tcline      = 50;
N           = 16;
Vtransform  = 2;
Vstretching = 4;
%%%%%%%%   END OF USER INPUT  %%%%%%%%%%%%%%%
```

When you run this m file, it will create multiple files:
- Initial conditions: coawst_ini.nc
- Boundary conditions: coawst_bdy_20121028.nc, 29, 30, 31, 01 (5 files) and a merged_coawst_bdy.nc (all 5 of those merged into 1 file).
- Climatology: coawst_clm_20121028.nc, 29, 30, 31, 01 (5 files) and a merged_coawst_clm.nc (all 5 of those merged into 1 file). I renamed:
coawst_ini.nc to Sandy_ini.nc
merged_coawst_bdy.nc to Sandy_bdy.nc
merged_coawst_clm.nc to Sandy_clm.nc

For child grid, you need to create init and clm file. You can use the same procedure as above using the master_climatology file, or you can interpolate the data from the parent using:
create_roms_child_init('Sandy_roms_grid.nc', 'Sandy_roms_grid_ref3.nc', 'Sandy_ini.nc', 'Sandy_ini_ref3.nc')
and
create_roms_child_clm('Sandy_roms_grid.nc', 'Sandy_roms_grid_ref3.nc', 'Sandy_clm.nc', 'Sandy_clm_ref3.nc')

6. 2D BC's

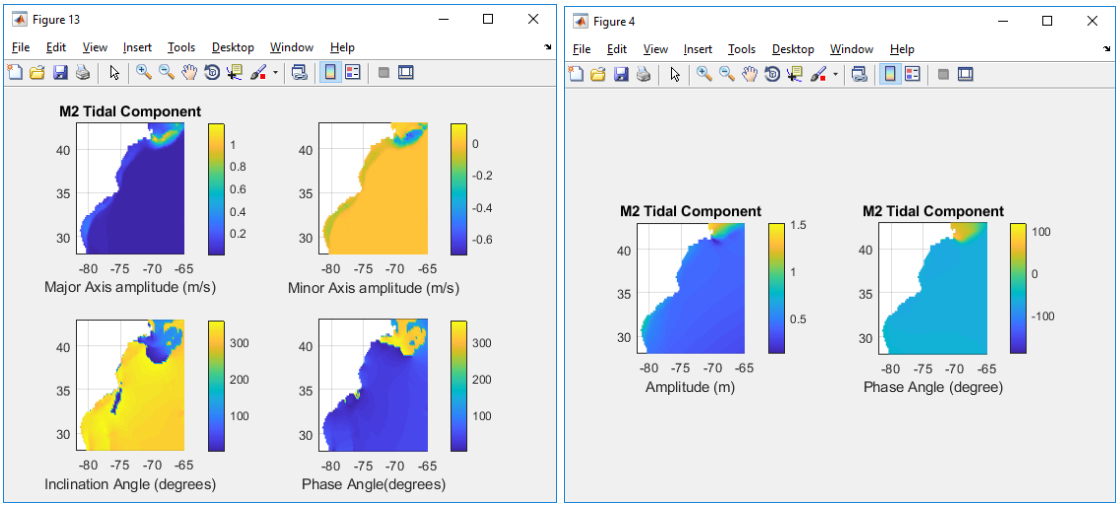
To add tidal forcing, you can use several methods We distribute a method that requires you to get the tidal data bases using:
svn checkout https://coawstmodel.sourcerepo.com/coawstmodel/data .
This will give you: adcirc_ec2001v2e_fix.mat, tpx_h.mat, and tpx_uv.mat.

tides/ をすべてコピー する
(tidal_ellipse/ と t-tide/ もコピーしておく.
roms_tide_forc_file.m と create_roms_tides.m を

Then you need to edit Tools/mfiles/tides/create_roms_tides to select tidal database and the start time. We chose the OSU tidal data base and ran that file as:
Sandy/training へ コピー する)

```
create_roms_tides
```
OSU = 1

to create tide_forc_Sandy.nc



M2 tides to force Sandy example (1 of 9 tidal constituents).

7. Surface forcings.

To create ROMS surface forcing files, there are several options such as:

- create_roms_forcings.m: converts data from matlab workspce to a netcdf forcing file.
- narrnc2roms.m : converts nc files from ftp.cdc.noaa.gov/Datasets/NARR/monolevel to netcdf.
- ncei_2roms.m : uses THREDDS to get data and create a forcing file.

We used ncei_2roms to create the romsforc_NARR_Sandy2012.nc for Sandy.

8. ROMS input files.

There are several input files to edit.

- Projects/Sandy/sandy.h

Here is where you set model compile options. For this current setup, we will just run ROMS with a child grid so set:

```
#define ROMS_MODEL
#define NESTING
#undef  WRF_MODEL
#undef  SWAN_MODEL
#undef  MCT_LIB
#undef  MCT_INTERP_OC2AT
#undef  MCT_INTERP_WV2AT
#undef  MCT_INTERP_OC2WV
```

WRF なし

There are many other ocean model options in that file.

- Projects/Sandy/ocean_sandy.in

There are many settings in here. Some of them are:

```
! Number of nested grids.
      Ngrids =  2

! Number of grid nesting layers.  This parameter is used to allow
refinement and composite grid combinations.
  NestLayers =  2

! Number of grids in each nesting layer [1:NestLayers].
GridsInLayer =  1 1

! Grid dimension parameters. See notes below in the Glossary for
how to set these parameters correctly.
Lm == 82  114       ! Number of I-direction INTERIOR RHO-points
Mm == 62   84       ! Number of J-direction INTERIOR RHO-points
N == 16   16       ! Number of vertical levels
ND ==  0           ! Number of wave directional bins
Nbed =  0            ! Number of sediment bed layers
NAT =  2            ! Number of active tracers (usually, 2)
NPT =  0            ! Number of inactive passive tracers
NCS =  0            ! Number of cohesive (mud) sediment tracers
NNS =  0            ! Number of non-cohesive (sand) sediment

…
! Input NetCDF file names, [1:Ngrids].
     GRDNAME == Projects/Sandy/Sandy_roms_grid.nc \
                Projects/Sandy/Sandy_roms_grid_ref3.nc
     ININAME == Projects/Sandy/Sandy_ini.nc \
                Projects/Sandy/Sandy_ini_ref3.nc
     ITLNAME == ocean_itl.nc
…

! Nesting grids connectivity data: contact points information.
NGCNAME =  Projects/Sandy/Sandy_roms_contact.nc

! Input lateral boundary conditions and climatology file names.
NCLMFILES == 1                          ! number of climate files
CLMNAME == Projects/Sandy/Sandy_clm.nc \
           Projects/Sandy/Sandy_clm_ref3.nc

NBCFILES == 1                           ! number of boundary files
BRYNAME == Projects/Sandy/Sandy_bdy.nc
For COAWST, you can also use multiple Boundary and/or Climatology
files. For example you could use:
NBCFILES == 1                           ! number of boundary files
!    BRYNAME == Projects/Sandy/Sandy_bdy.nc
     BRYNAME == Projects/Sandy/coawst_bdy_20121028.nc |
                Projects/Sandy/coawst_bdy_20121029.nc |
                Projects/Sandy/coawst_bdy_20121030.nc |
                Projects/Sandy/coawst_bdy_20121031.nc |
                Projects/Sandy/coawst_bdy_20121101.nc
```

チュートリアルでは作成法の説明なし

トル

```
NFFILES == 1
FRCNAME == Projects/Sandy/romsforc_NARR_Sandy2012.nc \
           Projects/Sandy/romsforc_NARR_Sandy2012.nc
```

9. Build and run code

- edit and run coawst.bash and set the Project name, header file location, and build location.
- to run the code, edit your run file like run_poseidon with something like:

```
#!/bin/bash
#PBS -N roms_sandy
#PBS -l nodes=1:ppn=36,walltime=120:00:00
#PBS -q standard

NPROCS=`wc -l < $PBS_NODEFILE`
cd /raid1/jcwarner/Models/COAWST
mpirun -np 16 -machinefile $PBS_NODEFILE ./coawstM > wrf_run1.out
```

So for this example it used np 16 cores, and this number should = NtileI*NtileJ in your ocean.in

10. look at the output.

ncview Sandy_ocean_his.nc, ncview Sandy_ocean_ref3_his.nc

Section 10. Setting up a SWAN application.

There are many ways to set up a SWAN application, dependent on what physcis you are looking at, the domain size, nesting, etc. This is just one example. We will continue the Sandy example here to set up the SWAN grids (2 grids nested).

1. SWAN grid

There are many ways to make a swan grid. We will use the roms grids and use them to create SWAN on the same grid. You can use differnet size grids (see for example the inlet_test/diffgrid), but for this application we will use the same grids for both roms and swan.

`roms2swan('Sandy_roms_grid.nc')`
This created swan_coord.grd and swan_bathy.bot and i renamed them to be Sandy_swan_bathy.bot and Sandy_swan_coord.grd

`roms2swan('Sandy_roms_grid_ref3.nc')`
This created swan_coord.grd and swan_bathy.bot and i renamed them to be Sandy_swan_bathy_ref3.bot and Sandy_swan_coord_ref3.grd

The grid size information needs to be entered to the SWAN input. To get the size I usually do this:
`netcdf_load('Sandy_roms_grid.nc')`

```
size(h)  (ans =  87 65)
So we enter swan.in with values that are 1 less:
CGRID CURVILINEAR 86 64 EXC 9.999000e+003 9.999000e+003 CIRCLE 36
0.04 1.0 24
READGRID COORDINATES 1 'Projects/Sandy/Sandy_swan_coord.grd' 4 0
0 FREE
```
Similarly for the child.

2. Wind Forcing

If not coupled to wrf, you need to download wind data from somewhere and create an ascii wind forcing file. See the SWAN manual as to the format. We typically use format number 4. You can obtain winds from many places. We have some files to do this.

To create a surface wind forcing file for SWAN, you can use ncei_2swan. This m-file uses THREDDS to access data via internet to create the forcing file. Here is a shortened part of the header for that file:

```
%(1) Select which variables to include in this ascii forcing file.
%  put a '1' if you want to include it, '0' otherwise.
get_Wind=1;       % surface u- and v- winds (m/s)

%(2) Enter name of output SWAN forcing file
SWAN_forc_name='swan_narr_Oct2012.dat';

%(3) Enter start and end dates - needed if using get_NARR or
get_NAM
namnarr_start = datenum('28-Oct-2012');
namnarr_end   = datenum('01-Nov-2012');

%(4) Select which data to obtain: NAM, NARR, or GFS.
get_NARR=0;  %NARR-A grid 221 32km data, dt = 3 hr
get_NAM=0;   %NAM grid 218 12km data, dt = 3 hr
get_GFS=1;   %GFS 0.5 degree, dt = 6 hr
%

%(5) Select to interpolate to a swan grid or a user defined grid.
% Set one of these to a 1, the other to a 0.
interpto_swan_grid = 0;
interpto_user_grid = 1;
. . .
% !!! I made this 0.25 for the Sandy test case, you need finer
resolution like 0.1  !!!
  lon_rho=[255:0.25:310]+offset;
  lat_rho=[ 10:0.25:50 ];  % Create a 0.1 degree lon-lat grid
...
```
Run this file as
```
ncei_2swan
```
and this will create swan_GFS_Oct2012.dat. Make a copy of this for the child swan grid.
```
cp swan_GFS_Oct2012.dat swan_GFS_Oct2012_ref3.dat
```

In the swan INPUT file, you need to set the WIND Inpgrid commnand line to be consistent with the data file created. The INPGRID line for this example would look like

```
&& KEYWORD TO CREATE WIND GRID &&
INPGRID WIND REGULAR -105 10 0 220 160 0.25 0.25 &
       NONSTATIONARY 20121028.000000 6 HR 20121031.000000
READINP WIND 1 'Projects/Sandy/swan_GFS_Oct2012.dat' 4 0 FREE
```

These values are from the grid size and time stamps:
The -105 is because the roms grid uses longitudes -180:180.
The 10 is coincident with latitude start of 10.
The 0 is angle.
The 220 is number of 'x-direction' points: (310-255)/0.25 = 220.
The 160 is number of 'y-direction' points: (50-10)/0.25 = 160.
The 0.25 and 0.25 are the delta space.
Then list the time start, dt, time end.

If you have more than one SWAN grid, you can use the same wind file but you will need to make a copy of it. For example

```
cp swan_GFS_Oct2012.dat swan_GFS_Oct2012_ref3.dat
and then put in the child swan.in:
&& KEYWORD TO CREATE WIND GRID &&
INPGRID WIND REGULAR -105 10 0 220 160 0.25 0.25 &
       NONSTATIONARY 20121028.000000 6 HR 20121031.000000
READINP WIND 1 'Projects/Sandy/swan_GFS_Oct2012_ref3.dat' 4 0
FREE
```

3. Boundary data

There are several sources and methods to obtain boundary data for SWAN. You can create 1D or 2D Spec files from a buoy or ADCP, or create TPAR files that have parametric values of wave height, period, and direction. In this example we will obtain data from Wave Watch 3 and use this to create boundary files. You will need to download historical ww3 from NOAA's ftp server. First, decide what grid and data format you want to use. Read this:
ftp://polar.ncep.noaa.gov/pub/history/waves/README You have several choices.

To create TPAR (parametric foring files)

1. Get the data. Go here to get ftp data from the grid and time period you need. For example, this is the global data set:
ftp://polar.ncep.noaa.gov/pub/history/waves/nww3/
scroll down to multi_1.glo_30m.xx.YYYYMM.grb2
get the three files for each month: xx = hs, tp, dp files (Hsig, Period, and Direction).
multi_1.at_10m.tp.201210.grb2
multi_1.at_10m.hs.201210.grb2
multi_1.at_10m.dp.201210.grb2
Do not change the file names.

2. The file Tools/mfiles/swan_forc/ww3_swan_input.m is the main driver. This can be set to create SWAN TPAR boundary forcing files. For hindcast mode, the TPAR files will be converted from the grib2 data using nctoolbox. This matlab toolbox is not included in the COAWST distribution, so go here and get it: http://code.google.com/p/nctoolbox/

Edit Tools/mfiles/swan_forc/ww3_swan_input.m
and enter the user required info in the USER section of that m file. This includes:

```
% 1) Enter WORKING DIRECTORY.
working_dir='coawst_v3.4_tests\Projects\Sandy\ww3'

% 2) Enter start dates of data requested.
yearww3='2012';    %input year of data yyyy
mmww3='10';        %input month of data mm
ddww3='00';        %keep this as '00'
long_run=0;

% 3) Enter path\name of SWAN grid. This is set up to use the roms
grid as the same for swan.
modelgrid='coawst_v3.4_tests\Projects\Sandy\Sandy_roms_grid.nc';

% 4) Enter the spacings of the forcing file locations around the
perimeter
specres=20; % spec point resolution

% 5)
% Here you decide to use gridded field output (to make TPAR) or
spectral partition data (to make 2Dspec).
% Pick one of these:
use_field_output=1;     % will create TPAR files.
use_partition_data=0;   % will create 2D spec files.
% Enter the ww3 grid area
ww3_area='multi_1.at_10m';    %western north atlantic
```

Run the m file. This will create multiple TPAR* files and a file called
Bound_spec_command.

3. Edit the SWAN INPUT file (for the parent only) and add the information from the Bound_spec_command file. Place the TPAR files in the Projects folder and make sure path names are correct.

To create 2D Spec files (spectral foring files)
1. Get the data. Go here to get ftp data from the grid and time period you need. For example, this is the global data set:
ftp://polar.ncep.noaa.gov/pub/history/waves/multi_1/
scroll down to 201210/partitions/ and get
multi_1.partition.glo_30m.201210
This is a big file. Do not change the file names.

2. The file Tools/mfiles/swan_forc/ww3_swan_input.m is the main driver. This can be set to create SWAN 2D spec forcing files. For hindcast mode, the 2Dspec files will be converted from the ASCII partition file.
Edit Tools/mfiles/swan_forc/ww3_swan_input.m
and enter the user required info in the USER section of that m file. This includes:

```
% 1) Enter WORKING DIRECTORY.
working_dir='coawst_v3.4_tests\sandy\Projects\Sandy\ww3'

% 2) Enter start dates of data requested.
yearww3='2012';    %input year of data yyyy
mmww3='10';        %input month of data mm
ddww3='00';        %keep this as '00'
long_run=0;

% 3) Enter path\name of SWAN grid. This is set up to use the roms
grid as the same for swan.
modelgrid='coawst_v3.4_tests\Projects\Sandy\Sandy_roms_grid.nc';

% 4) Enter the spacings of the forcing file locations around the
perimeter
specres=20; % spec point resolution

% 5)
% Here you decide to use gridded field output (to make TPAR) or
spectral partition data (to make 2Dspec).
% Pick one of these:
use_field_output=0;     % will create TPAR files.
use_partition_data=1;   % will create 2D spec files.
  partfile='multi_1.partition.glo_30m.201210';
```

Run the m file. This will create multiple 2DSpec files and a file called
Bound_spec_command.

3. Edit the SWAN INPUT file (for the parent only) and add the information from the Bound_spec_command file. Place the 2DSpec files in the Projects folder and make sure path names are correct.

3. Initial Conditions Files
Method 1: (try this first). In the swan input file, use the command
INITIAL DEFAULT

Method 2:
To create an init file for swan, you can run SWAN in stationary mode and create 'hot start' files. To do this, edit your project.h file and activate SWAN model (with nesting if needed).
#undef ROMS_MODEL
#define NESTING

```
#undef  WRF_MODEL
#define SWAN_MODEL
#undef  MCT_LIB
#undef  MCT_INTERP_OC2AT
#undef  MCT_INTERP_WV2AT
#undef  MCT_INTERP_OC2WV
```

Run coawst.bash to build a new executable.
Edit the bottom of swan_sandy.in to be:

```
COMPUTE STAT 20121028.120000
&COMPUTE NONSTAT 20121028.120000 180 SEC 20121030.120000
HOTFILE 'Sandy_init.hot'
```

And the bottom of swan_sandy_ref3.in to be:

```
COMPUTE STAT 20121028.120000
&COMPUTE NONSTAT 20121028.120000 90 SEC 20121030.120000
HOTFILE 'Sandy_ref3_init.hot'
```

And in both swan.in files use

```
& Restart name **********************************
INIT
```

Run SWAN and create the init files use:

```
mpirun -np 1 ./coawstM Projects/Sandy/swan_sandy.in
Projects/Sandy/swan_sandy_ref3.in > cwstv3.out
```

This was all on one line. Notice we have to call both swan input files. Also, the '1' means one processor and we will get one init *.hot file for each grid. We can use that to run multiple processor simulations next. You can look at the output. ncview Sandy_hsig.nc

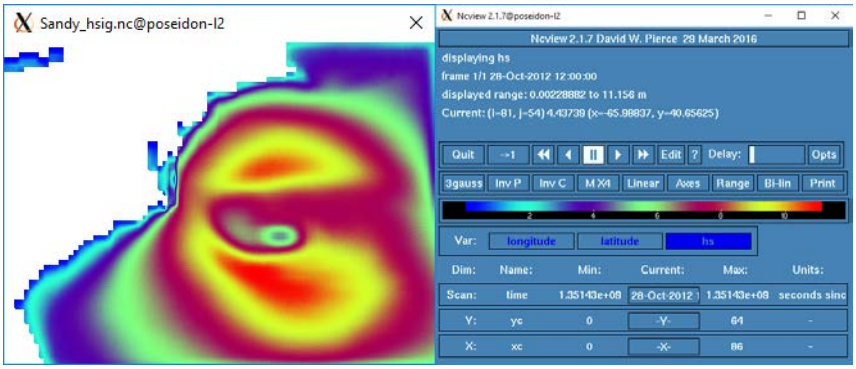


Figure. Initial conditions for Swan.

4. Run SWAN for a longer simulation

To run SWAN for the multi day simulation, edit the swan.in files and list the init files:
For swan_sandy.in use:

```
& Restart name **********************************
INITIAL HOTSTART SINGLE 'Projects/Sandy/Sandy_init.hot'
```

```
&INIT
. . .
RESTART 'swan_rst.dat' FREE 1 HR
% and change the run to be nonstationary
&COMPUTE STAT 20121028.120000
COMPUTE NONSTAT 20121028.120000 180 SEC 20121030.120000
HOTFILE 'Sandy_init.hot'
```

Notice the command to create hourly restart files 'RESTART' is before the COMPUTE command. So every hour, swan will create a restart file (like for roms) that you can use to restart a simulation that dies at some undesired point. You can change the '1 HR' to be other times.

Similarly, for swan_sandy_ref3.in use:

```
& Restart name *********************************
INITIAL HOTSTART SINGLE 'Projects/Sandy/Sandy_ref3_init.hot'
&INIT
. . .
RESTART 'swan_ref3_rst.dat' FREE 1 HR
% and change the run to be nonstationary
&COMPUTE STAT 20121028.120000
COMPUTE NONSTAT 20121028.120000 180 SEC 20121030.120000
HOTFILE 'Sandy_ref3_init.hot'
```

Then run the program
mpirun -np X./coawstM Projects/Sandy/swan_sandy.in
Projects/Sandy/swan_sandy_ref3.in > cwstv3.out

Section 11. Setting up a WAVEWATCH III application.

~ p75

As of July 2019 we have added WaveWatch III from tnbheir github site to the COAWST coupled system. WaveWatch can be run alone or coupled to ROMS, to WRF, or to both. For now we can only run one WW3 grid (ww3_shel), and if WW3 is coupled to another model(s), the other model(s) can run multiple grids. We are looking to incorporate the multi domain capability of WW3 in the future.

As with every other model, we recommend that the user become familiar with that model first. The WW3 user manual is provided in COAWST/WW3 directory. To run WW3, we will go through the Sandy test as an example to run WW3 by itself. This will provide the steps necessary.

1) edit coawst.bash and set:
export COAWST_APPLICATION=SANDY
and set the root location of your code, such as:
export MY_ROOT_DIR=/sand/usgs/users/jcwarner/tests/coawst_v3.3

2) We then need to set 5 WW3 environement settings.
#1) COAWST_WW3_DIR is a pointer to root WW3 code, do not change.

```
export   COAWST_WW3_DIR=${MY_ROOT_DIR}/WW3/model

#2) WWATCH3_NETCDF can be NC3 or NC4. We need NC4 for COAWST. do not
change.
export   WWATCH3_NETCDF=NC4

#3) WWATCH_ENV points to WW3 environment listing. do not change.
export   WWATCH_ENV=${COAWST_WW3_DIR}/wwatch.env

#4) NETCDF_CONFIG is needed by WW3. You need to set this:
export   NETCDF_CONFIG=/usr/bin/nc-config
This may require
export   NETCDF_CONFIG=/usr/bin/nf-config
depending on your system.

#5) WW3_SWITCH_FILE is like cpp options for WW3. You need to create it and
#       list the full name here. This is described below.
export  WW3_SWITCH_FILE=switch_sandy

6) As with all applications, set the MPI and compiler information (this will depend on
your system):
export          USE_MPI=on          # distributed-memory parallelism
export       USE_MPIF90=on            # compile with mpif90 script
export        which_MPI=openmpi       # compile with OpenMPI library
export            FORT=ifort

7) Set the location of the header and analytical files:
export     MY_HEADER_DIR=${MY_PROJECT_DIR}/Projects/Sandy
export MY_ANALYTICAL_DIR=${MY_PROJECT_DIR}/Projects/Sandy

8) For this description, we are saying to run WW3 by itself so edit
Projects/Sandy/sandy.h and only list
#define WW3_MODEL

9) The compiler flags for the WW3 build are taken from the Compilers/* file. For
example, if you work on Linux with ifort, then the Compilers/Linux-ifort.mk file will be
used for the WW3 build.  You may also need to edit some WW3 specific build files.  So
if you selected ifort (for example), then we need to look at
COAWST/WW3/model/bin/comp.Intel
COAWST/WW3/model/bin/link.Intel
Please look at these files (or the files needed for your compiler) to see if the flags are
correct.

10) Create a switch file.
The switch file is like the cppdefs file for roms. It tells WW3 what features to compile.
We have a file COAWST/WW3/model/bin/switch_sandy. List the full switch file name
```

(new feature) in the coawst.bash file. The file should start with ‘switch_’ Here is the switch_sandy:

F90 NOGRB COAWST LRB4 NC4 TRKNC DIST MPI PR3 UQ FLX0 LN1 ST4 STAB0 NL1 BT4 DB1 MLIM TR0 BS0 IC2 IS2 REF1 IG1 XX0 WNT2 WNX1 RWND CRT1 CRX1 TIDE O0 O1 O2 O2a O2b O2c O3 O4 O5 O6 O7

Please read the WW3 manual in the WW3 directory. Some important options are:

NOPA DO NOT USE THIS. This is for stand alone WW3, and I have modified the build structure to be part of COAWST. So do not ever use it.

COAWST ALWAYS list this. I have modified the build strucutrre to be part of COAWST (this is new as of June 2018.).

SCRIP, SCRIPNC I don’t use the WW3 scrip but we may in the future. So these are not really needed.

DIST, MPI These are to use mpi, needed.

11) compile the code by running ./coawst.bash at the command prompt.
If it compiles correctly, you will get a coawstM.

12) Now we need to create some WW3 grids files. Here is a way to do that. There are probalbly many other ways to do this.
You can use Tools/mfiles/mtools/create_ww3_grid_files to create an x, y, bath, and mask files for WW3. This m file uses the roms grid to create the WW3 files placed in Projects/Sandy/WW3:
ww3_sandy_xcoord.dat, ww3_sandy_ycoord.dat, ww3_sandy_bathy.bot, and ww3_sandy_mapsta.inp.

13) Forcing files for WW3: again, there are probably many ways to do this, but here is one way.
You can use Tools/mfiles/mtools/create_ww3_wind_forcing to create a wind forcing file ww3_sandy_wind_forc.dat.

14) We then need to create some WW3 run files using 4 of their utilities:

Utility ww3_grid
- edit Projects/Sandy/WW3/ww3_grid.inp. Here is where you enter number of bins, advection choices, time steps (we chose 180), and the grid settings

```
  'CURV'  T  'NONE'
   84     64
   40 1. 0. 1 1 'FREE' 'NAME' 'ww3_sandy_xcoord.dat'
   40 1. 0. 1 1 'FREE' 'NAME' 'ww3_sandy_ycoord.dat'
   -1.0  -1.0  40 -1. 1 1 '(....)' 'NAME' 'ww3_sandy_bathy.bot'
```

cd to Projects/Sandy/WW3 and run
../../../WW3/model/exe/ww3_grid.exe
This will create mask.ww3, mapsta.ww3, mod_def.ww3.

If you change the time steps in ww3_grid.inp, then you need to rerun ww3_grid.exe and create new mod_def file.

Utility ww3_strt
- edit Projects/Sandy/WW3/ww3_strt.inp. Here is where you enter init information.
cd to Projects/Sandy/WW3 and run
../../../WW3/model/exe/ww3_strt to create restart.ww3.

Utility ww3_prep
- edit Projects/Sandy/WW3/ww3_prep.inp. Here is where you enter forcing data. We have:

```
$
  'WND' 'LL' T T
$ Additional input format type 'LL' ---------------------------------- $
$ Grid range (degr. or m) and number of points for axes, respectively.
$ Example for longitude-latitude grid.
   -105. -50. 276   10. 50. 201
```

This tells WW3 that the wind is on a grid with these lon/lat values.

```
$ Define data files ------------------------------------------------- $
$ The first input line identifies the file format with FROM, IDLA and
$ IDFM, the second (third) lines give the file unit number and name.
 'NAME' 1 1 '(....)' '(....)'
  40 'ww3_sandy_wind_forc.dat'
```

This tells WW3 the name of the wind forcing file.
cd to Projects/Sandy/WW3 and run
../../../WW3/exe/ww3_prep to create wind.ww3.

Utility ww3_shel
- edit Projects/Sandy/WW3/ww3_shel.inp. Here is where you enter run information.
For this test we are just looking to run WW3 alone so we choose:

```
  F F     Water levels
  F F     Currents
  T F     Winds
```

The first letter is to set if the field is to be not used (F), used as read in from a file (T), or used and read in from a coupled model (C). So those settings are to read winds from a file.

If we had coupled WW3 to WRF we could use

```
  F F     Water levels
  F F     Currents
  C F     Winds
```

And that would get coupled winds from WRF.

If we had coupled WW3 to ROMS, then we could use

```
  C F     Water levels
```

C F Currents
T F Winds
To get coupled fields from roms to ww3 and read a wind file.

If we had coupled WW3 to ROMS and to WRF, then we could use
C F Water levels
C F Currents
C F Winds

You need to set the run times here:
$ Type 1 : Fields of mean wave parameters
20121028 120000 1800 20121030 120000
and set to have the fields of Hwave, Dwave, and Lwave written out:
$ HS LM T02 T0M1 T01 FP DIR SPR DP
T T T T T T T T T

We do NOT need to run ../exe/ww3_shel because that would run the wave model. We will run the wave model as coawstM.

15) Now we are ready to run WW3. Cd to the root dir. Now we need to copy (or soft link) those ww3 files to the root location.
cp Projects/Sandy/WW3/* .
and use run_* script to run the code on your system. We have run_coawst as an example of a run sciprt, but you need to get a script that works on your system. You do not need to set the number of processors in any of the ww3 files.
mpirun –np X ./coawstM Projects/Sandy/WW3/ww3_grid.inp

16) When the model is done you can visualize the output by using
./WW3/model/exe/ww3_ounf
This will create a ww3.*.nc file called ww3.201210.nc.

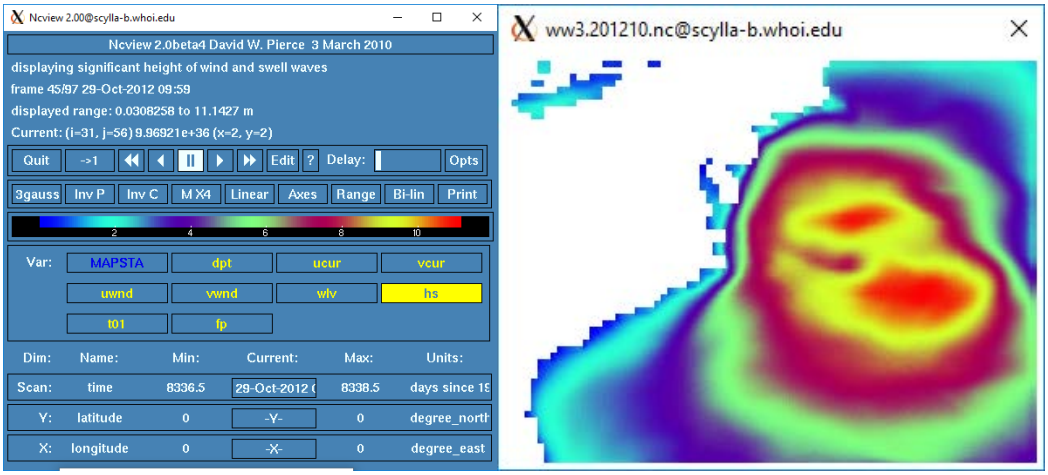


Figure 11.1 Ncview of WW3 wave heigth for Sandy simulation.

Section 12. Setting up a Coupled Application.

A coupled application is when you are using more than one model. To set up a coupled application, we highly recommend that you try to run each individual model first. This will allow you to focus on the grids and input files for each model separately before you run the system coupled. We have been going thru the Sandy example for WRF, ROMS, SWAN, and WaveWatch in Sections 8, 9, 10, and 11. We will now go through the steps to set up a coupled application following the Sandy Project. To run a coupled setup you need to:

- use SCRIP if the model grids are different or if you have nesting in any of the models.
- use the coupling.in input file
- set some cpp options,

1. SCRIP

We use the Spherical Coordinate Remapping Interpolatin Package (SCRIP) from Los Alamos National Laboratory to create interpolation weights. The weights are needed if the grids are different sizes or if you have nested grids. The weights are used by MCT to conservatively remap the scalars and fluxes between the grids. We have modified SCRIP to read in the model grids, compute the weights and then to provide one netcdf output file with all weights. This output file uses structures so it needs Netcdf_4.

- cd to Lib/SCRIP_COAWST and edit the makefile. Select the compiler:

```
FORT = ifort
#FORT = gfortran
#FORT  = pgi
```

The location of the directory is important, as it uses the ../../Compilers directory (just like ROMS and SWAN will) to get the Fortran flags. Build the code using:

```
make
```

this should create a scrip_coawst[.exe]

- Edit the SCRIP input file. We provide several, for this case lets use scrip_coawst_sandy. Here are some of the important parts:

```
! 1) Enter name of output netcdf4 file
OUTPUT_NCFILE='scrip_sandy_static.nc'

! 2) Enter total number of ROMS, SWAN, WW3, and WRF grids:
!
NGRIDS_ROMS=2,
NGRIDS_SWAN=2,
NGRIDS_WW3=0,
NGRIDS_WRF=2,

! 3) Enter name of the ROMS grid file(s):
!
ROMS_GRIDS(1)='../../Projects/Sandy/Sandy_roms_grid.nc',
ROMS_GRIDS(2)='../../Projects/Sandy/Sandy_roms_grid_ref3.nc',
```

```
! 4) Enter SWAN information:
! the name(s) of the SWAN grid file(s) for coords and bathy.
! the size of the SWAN grids (full number of center points), and
! if the swan grids are Spherical(set cartesian=0) or
!                          Cartesian(set cartesian=1).
!
SWAN_COORD(1)='../../Projects/Sandy/Sandy_swan_coord.grd',
SWAN_COORD(2)='../../Projects/Sandy/Sandy_swan_coord_ref3.grd',
SWAN_BATH(1)='../../Projects/Sandy/Sandy_swan_bathy.bot',
SWAN_BATH(2)='../../Projects/Sandy/Sandy_swan_bathy_ref3.bot',
SWAN_NUMX(1)=87,  ! THIS WOULD be the size(h,1) if it was ROMS.
SWAN_NUMX(2)=116, ! THIS WOULD be the size(h,1) if it was ROMS.
SWAN_NUMY(1)=65,  ! THIS WOULD be the size(h,2) if it was ROMS.
SWAN_NUMY(2)=86,  ! THIS WOULD be the size(h,2) if it was ROMS.
CARTESIAN(1)=0,   ! 0 means no Cartesian, yes spherical
CARTESIAN(2)=0,

! 6) Enter the name of the WRF input grid(s). If the grid is a
!    moving child nest then enter that grid name as 'moving'.
!    Also provide the grid ratio, this is used for a moving nest.
!
WRF_GRIDS(1)='../../Projects/Sandy/wrfinput_d01',
WRF_GRIDS(2)='../../Projects/Sandy/wrfinput_d02',
!WRF_GRIDS(2)='moving',
PARENT_GRID_RATIO(1)=1,
PARENT_GRID_RATIO(2)=3,
PARENT_ID(1)=0
PARENT_ID(2)=1
```

Run the SCRIP as:

```
./scrip_coawst scrip_coawst_sandy.in
```

Then I copied the output netcdf file scrip_sandy_static.nc to Projects/Sandy.

2. Coupling.in file

edit Projects/Sandy/coupling_sandy.in Here are some of the important parts:

```
Set the number of cores for each model.
   NnodesATM =  1                   ! atmospheric model
   NnodesWAV =  1                   ! wave model
   NnodesOCN =  1                   ! ocean model

! Time interval (seconds) between coupling of models.
  TI_ATM2WAV =   1800.0d0 ! atmosphere to wave coupling interval
  TI_ATM2OCN =   1800.0d0 ! atmosphere to ocean coupling interval
  TI_WAV2ATM =   1800.0d0 ! wave to atmosphere coupling interval
  TI_WAV2OCN =   1800.0d0 ! wave to ocean coupling interval
  TI_OCN2WAV =   1800.0d0 ! ocean to wave coupling interval
```

```
  TI_OCN2ATM =   1800.0d0 ! ocean to atmosphere coupling interval

! Enter names of Atm, Wav, and Ocn input files.
! The Wav program needs multiple input files, one for each grid.

   ATM_name = namelist.input   ! atmospheric model
   WAV_name = Projects/Sandy/swan_sandy.in \
              Projects/Sandy/swan_sandy_ref3.in   ! wave model
!  WAV_name = WW3/work/ww3_grid.inp
   OCN_name = Projects/Sandy/ocean_sandy.in       ! ocean model

! Sparse matrix interpolation weights files. You have 2 options:
! Enter "1" for option 1, or "2" for option 2, and then list the
! weight file(s) for that option.

   SCRIP_WEIGHT_OPTION = 1
!
! Option 1: IF you set "SCRIP_WEIGHT_OPTION = 1", then enter name
!           of the single netcdf file containing all the exchange
!           weights. This file is created using the code in
!           Lib/SCRIP_COAWST/scrip_coawst[.exe]

!  SCRIP_COAWST_NAME = Projects/Sandy/scrip_sandy_moving.nc
   SCRIP_COAWST_NAME = Projects/Sandy/scrip_sandy_static.nc
!  SCRIP_COAWST_NAME =
Projects/Sandy/WW3/scrip_sandy_nowavenest.nc
! This last file is to couple WW3 with ROMS and WRF.
```

- 3. Cpp options. Edit your project.h file (sandy.h) and activate:

```
#define ROMS_MODEL
#define NESTING
#define WRF_MODEL
#define SWAN_MODEL
#define MCT_LIB
#define MCT_INTERP_OC2AT
#define MCT_INTERP_WV2AT
#define MCT_INTERP_OC2WV
```

- 4. Build the executable

edit and run coawst.bash to build the code

- 5. Run the model

edit your run script and set np = `NnodesATM+NnodesWAV+NnodesOCN = NP`

edit namelist.input to set `NnodesATM = nproc_x*nproc_y`

`edit ocean.in and set NtileI*NtileJ = NnodesOCN`

swan does not set the number of procs in its input file.

Run it as mpirun –np NP ./coawstM Projects/Sandy/coupling_sandy.in

6. For a Moving WRF Nest:
You can have a moving WRF nest coupled to the system. For SCRIP set
```
!WRF_GRIDS(2)='../../Projects/Sandy/wrfinput_d02',
WRF_GRIDS(2)='moving',
```
This will create a different scrip weights file. Select this file in coupling.in
You will need to recompile and select vortex following for WRF.
The run it as before.

Section 13. Setting up an InWave application.

InWave solves the wave action balance conservation equation, which is given by:

$$\frac{\partial(A)}{\partial t}+\frac{\partial\left(C_{g,x}A\right)}{\partial x}+\frac{\partial\left(C_{g,y}A\right)}{\partial y}+\frac{\partial\left(C_{g,\theta}A\right)}{\partial \theta}=-\frac{D_w}{\sigma}$$

where A stands for the time varying energy action or density for each directional bin ($A=E(x,y,\theta,t)/\sigma(x,y,\theta,t)$); $C_{g,x}$, $C_{g,y}$, and $C_{g,\theta}$ represent the wave group celerity in the x, y, and θ directions respectively (Olabarrieta, et al., in prep). D_w is the energy dissipation due to the wave breaking and σ is the intrinsic wave frequency.

There are two ways to run InWave: 1) forcing the model using a 2D spec file from SWAN and 2) imposing time series of wave action density along the open boundaries. Option 1 has an example for the Inlet_test/InWave (Example 6.5) and Option 2 example is Shoreface_InWave (Example 6.8). The cpp options for InWave are described in 4.3.10. More information will be provided soon.

Section 14. MATLAB (.m) scripts for pre/post processing.

- As of release 567, February 8, 2012, we have rewritten the m files to use native matlab netcdf interface. This will allow a more uniform practice amongst users. The caveat of this is that the matlab interface uses the Fortran convention, ie the variables are read and writen as Var(x,y,z,t). Many of the previous toolboxes used the C convention of Var(t,z,y,x). Please be careful when using the native matlab. You can use Tools/mfiles/mtools/netcdf_load to use native matlab to load a netcdf file into the workspace.

- Another optional toolbox that may be needed is nctoolbox found at: http://code.google.com/p/nctoolbox/
Follow instructions on that website to install that toolbox. These nctools are optionally used in the roms_clm routines to read HYCOM data via opendap, and they are optionally used in the swan_forc m files to read the grib2 WW3 data. This nctoolbox is not required to run the COAWST system but provides additional functionality.

mfile listing:
- inwave_tools: under development.

- m_map: set of utilities used in the grid generation routines, used to convert lat/lon to meters, and to support different projections.

- mtools: set of utilites mostly for ROMS to create grids, load a netcdf file, converting grids for scrip, creating wrf grid from roms grid, etc.

- roms_clm: main driver is roms_master_climatology_coawst_mw.m, used to create boundary, init, and climatology files for roms grids using opendap to read HYCOM data.

- rutgers: contains m files from Rutgers for bathymetry, boundary, coastlines, forcing, grid, landmask, netcdf, seawater, and utility folders.

- swan_forc: main driver is ww3_swan_input.m to read WW3 grib2 data and create SWAN TPAR forcing files

- tides: Use this set of m files to create tidal forcings for roms.

Section 15. Tracking model development changes log.

"svn log" gives the whole listing. Here are some key updates:

Rev #	Date	Description
----------	COAWST V1.0	----------
0	01Jan2009	First upload.
17	23Feb2009	Merge grid refinement with atm coupling methods.
83	16Apr2009	Update time stepping for refined grids.
99	24Apr2009	Add SWAN forcing package.
107	28Apr2009	Add InWave.
147	06Jul2009	Add Inlet tests (3).
185	07Aug2009	Update to WRFV3.1.1
196	03Sep2009	Add BLOCK_JONSSON.
203	14Sep2009	Add UV_KIRBY vel avg for wave coupling.
207	16Sept2009	Added hwave/lwave from SWAN to WRF.
----------	COAWST V2.0	----------
250-262	01May2010	Update to ROMS svn 455, Update SWAN 4072ABCDE This was a major update. Complete build is now handled thru coawst.bash script.
----------	COAWST V3.0	----------
300-315		
330	15Jun2010	Removed old Project files.

331-338	07Jul2010	Add ATM2OCN_FLUXES to allow consistent wrf-roms fluxes and to update scrip interpolation to use conservative fluxes.
351	19Jul2010	Update Inlet tests to work with consv fluxes, and be more consistent. Update manual to include Inlet Tests descriptions, added m files to create the refined grids.
393	07Dec2010	SWAN uses urms instead of ubot to compute Madsen bottom stress. Also pass urms to roms for bbl models instead of ubot. Rename Ub_swan to Uwave_rms. If user does not activate a bbl, then swan fric will be based on user supplied value in swan.in. If user does activate bbl, then swan fric is based on ZoNik.
426	27Feb2011	Update to WRF 3.2.1
430	05Mar2011	Incorporate wec-vortex force method.
433	09Mar2011	Update to SWAN 40.81
469	02Jun2011	Add WPS and WRF to run separately
476-477	12Jul2011	Add WRF_CPL_GRID to choose which wrf grid to couple to. Modifications to work for ifort on cx2. Create mct_couple_params file to hold coupled vars. Modify WRF to write to stdout, not to rsl files. modify coupling.in files.
500-516	27Oct2011	Update to conservative 2-way grid refinement for ROMS.
528-531	30Nov2011	Incorporate Seaice module.
561	08Feb2012	Update m files to use native matlab netcdf interface. Go thru the distributed swan_forc, roms_clm, and mtools to ensure functionality. Update /cleanup manual.
599	11Jul2012	last distribution on hosted-projects.com Moved to Source repo.
602	11Jul2012	Allow SWAN only simulations.
603	11Jul2012	Allow WRF only simulations.
607-635	24Jul2012	Update to WRF 3.4.
669	02Oct2012	Add #define ROMS_MODEL to all roms apps.
672	11Oct2012	Change mct_couper_params to mct_wrf_coupler_params in wrf side, update wrf time step counter.
673	11Oct2012	Update ROMS grid refinement to allow refined region to vary spatially in the parent grid (interp2 w/ spline).
680	22Oct2012	Edit Compilers to not use LIBS=netcdf calls for swan only compilations.
681	22Oct2012	Allow SWAN to create its own hot file.
682	24Oct2012	Allow SWAN to run in STAT mode.
686	01Nov2012	Update mask_io to not overwrite point sources in his files.
701-708	04Feb2013	Update SWAN to 4091A.
725	26Apr2013	Update JOE_TC to use sf_surface_physics = 2.
751-752	06Sept2013	Update Rip_current test case to Kumar et al 2012.
758	16Sept2013	Update some roms input files so they all have Ltracer src
762-767	26Sept2013	Allow WRF-SWAN coupling, add DRAGLIM_DAVIS
771	23Oct2013	Add headland test case

789	19Dec2013	Update wrf-swan cpling for diff grids.
801	07Mar2014	Update Compilers and mct_coupler for gfortran with wrf.
813	17Apr2014	Update Compiles and wrf makefile, -I more wrf paths, gfortran flags for recl=4
866	07Aug2014	COAWST v3.1 released.
914	17Sept2014	Add new src weights for atm land points to ocn wet pts.
919	25Sept2014	Reorganize JOE TC and other test cases.
933	20Oct2014	Update wec stokes computations.
936	01Nov2014	Correct tile MPDATA ranges.
943	17Nov2014	Add spectral light and sea grass.
951	10Dec2014	Correct Longwave flux bug for roms from multiple wrf grids. Correct restart for coupled wrf runs, and modify wrf determination of grid number for cpling.
953	17Dec2014	Add some SCRIP as matlab routines
956	19Dec2014	Update to Rutgers nesting and wet/dry masking.
959	06Jan2015	Update way to call SCRIP in weights computation mfiles.
964	05Feb2015	Rutgers bug 658 to use LallocateClima in mod_arrays.F and correct gasdev.F for randum number; Rutgers bug # 659 for set_contact.F for wet_dry weights and mod_mixing.F for FORWARD_MIXING spelling.
965	10Feb2015	Correct swan compute bound locations for child, enforce no-removal of grid rows for tiling during all swan apps
973	04Mar2015	Update SCRIP routines to check for coincident grids.
974	04Mar2015	Add NPLANTS to SWAN read.
982	19Mar2015	Allow SWAN to create RESTART files during time stepping. Add gridnum to write out during SWAN STATIONARY runs.
985	24Mar2015	Add wetdry_psi to wrt_his, add swan restart file to Inlet_test, update inlet test sediment input files to include bcs, update init m files to read a his file and to add bedload.
986	02Apr2015	Updates for ROMS src bug #s 662, 663, ad 664 for t3dmix, lmd mixing, and 3dfldr routines.
990	14Apr2015	update ww3_swan_input to use nctoolbox v1.1.1
994	27Apr2015	update def his for pmask_wet
996	12May2015	correct east west sums in nesting.F
1000	26May2015	add disclaimer
1001	29May2015	change waves in WRF/Registry from i0124rhd to i012rhd, only have one namelist.input file for JOE_TC examples.
1008	14Jul2015	updates to ROMS to include src#665: fennel.h; src#666: nesting.F checksums; src#667 was wetdry mask write for his but we already had this; src#668: check_multifile EXIT removed; src#669: add limit_stflux_coolig to set_vbc; src#670: ad_v2dbc typo; src#671: mod_boundary tl typo
1010	04Aug2015	updates to ROMS to include src#674 MEAN_AGE plus inp_par changes to load_s1d and _s2d; src#675 for ANA_SSFLUX and ANA_PASSIVE; and src#676

		nesting.F deallocation. Update Adjoint, Representer, and Tangent folders, User, and Functionals.
1014	20Aug2015	updates to Rutgers: src#677 npzd powell; src#673 dogbone ngc files; src#678 ntimestep.F for >3 telescoped nested grids; src#679 inimetricval in mod_grid.F; src#680 masking in refined and read_stapar.F; update Projects Sedbed_toy and Sed_floc_toy
1015	31Aug2015	make sstupdate read i1, update joetc scrip master
1023	15Sep2015	update maplev to use nearest not average.
1032	08Oct2015	update to Rutgers: src#682 for timers.F; src#684 for nesting.F check of massflux for debugging only; src#685 for nesting call from main2d and main3d.
1033	08Oct2015	correct fac2 computation in wec_wave_mix.F.
1072-1077	20May2016	Release of COAWST v3.2
1084	24June2016	correct point source indices.
1086	07July2016	correct WRF moving nest GLW to ROMS.
1087	12July2016	update to Rutgers: src#697 output z_* to his files; src#698; src#699 gmake version; src#700 Output switches for sediment wrt to his and station files; src#701 SGRID conventions; src#702 4DV update
1111	22Sep2016	ROMS src updates 705, 706, 707, 709, 710, WRF Registry.EM_COMMON update, Correct WRF rotations to ROMS for U10, V10, USTRESS, and VSTRESS for non-mercator grids, add nomads_2roms.m, add offset in SCRIP-COAWST for coincident grids.
1122	06Dec2016	InWave read of 2d spec files
1130	01Jan2017	Update SCRIP for WRF – ROMS land sea masking mismatches.
1132-1136	15Feb2017	ROMS src updates including 704 and 708 Quick Files, 711-720 for bug in nesting tracer flux, adjoint updates, write water points, nc_config replace nf_config, Copyright to 2017, and reading Qbar.
1142	02Mar2017	set surface dissip props from swan to 0 for initial exchange to roms because they are not computed yet
1143	06Mar2017	update SCRIP for swan-roms coindent points.
1151	13Apr2017	add line contunations to SCRIP
1165	09Jun2017	update InWave Lwave and zeta_filter
1179	31July2017	add sedslump
1197-1200	12Jan2018	COAWST v3.3: Update ROMS to svn 885, WRF to v3.9.1.1, SWAN to v41.20, add WW3 v 5.16
1237-1256	04Jun2018	Update to COAWST v3.3.1 to include ROMS tickets 757-769
1350-1397	13Feb2019	Update to COAWST v3.4 to include ROMS tickets 770-794, 796,799, 800. WRF v 4.0.3.

1396	20Feb2019	Update srflux and ssflux for Sandy
1410	25Jul2019	Add WaveWatchIII from github site
1419	04Aug2019	Update WRF to 4.1.2
1423	11Aug2019	Update to ROMS 3.8
1439-1444	18Oct2019	Update InWave
1445-1452	04Nov2019	Update van der A sediment formula
1458	29Dec2019	Update SCRIP to use MPI and ultiple moving wrf grids
1470-1478	23Apr2020	Update to CAOWST v3.6 to include ROMS tickets 825-849, update WRF to 4.1.2, SCRIP in MPI, multiple moving nests for SCRIP, ROMS Tracer advection now listed *.in.

Section 16. Previous COAWST training workshops and Data.

We have held several COAWST Modeling System Trainings.
- The first was at the USGS Woods Hole Science Center from July 23-27, 2012. It was attended by 46 scientists from over 8 countries onsite.
- The second Training was held on the Woods Hole Oceanographic Institution (WHOI) campus from August 25-28, 2014. It was attended by 78 scientists from over 15 countries.
- The third was also on the WHOI campus from August 15-19, 2016 and was attended by over 85 scientists onsite and via webex worldwide.
- The fourth was at North Carolina State University Campus from February 25-28, 2019, and was attended by 70 scientists on site and ~ 30 online.

All the Trainnigs were recorded and the presentations, photos, agenda, and webex recordings have been posted to the svn site. To view the webex presentations from the earlier meetings, we also distribute the nbr2player.msi to install the viewer. The more recent meetings have recordings in mp4 format that can be viewed directly.

To access the Trainings, some data, and some test cases I suggest you make a folder called COAWST_data, cd to that folder and use:
svn checkout https://coawstmodel.sourcerepo.com/coawstmodel/data .

To access the 2012 meeting use
svn checkout https://coawstmodel.sourcerepo.com/coawstmodel/data/training_23jul2012 .

To access all the 2014 meeting presentations and webex use:
svn checkout https://coawstmodel.sourcerepo.com/coawstmodel/data/training_25aug2014/ .
To access only the 2014 meeting presentations use
svn checkout
https://coawstmodel.sourcerepo.com/coawstmodel/data/training_25aug2014/presentations .
To access only the 2014 meeting webex use
svn checkout
https://coawstmodel.sourcerepo.com/coawstmodel/data/training_25aug2014/webex .

To access all the 2016 meeting presentations and webex use:
svn checkout https://coawstmodel.sourcerepo.com/coawstmodel/data/training_15aug2016/ .
To access only the 2016 meeting presentations use
svn checkout
https://coawstmodel.sourcerepo.com/coawstmodel/data/training_15aug2016/presentations .
To access only the 2016 meeting webex use
svn checkout
https://coawstmodel.sourcerepo.com/coawstmodel/data/training_15aug2016/webex .

To access all the 2019 meeting presentations and online recordings use:
svn checkout https://coawstmodel.sourcerepo.com/coawstmodel/data/training_24feb2019/ .
To access only the 2019 meeting presentations use
svn checkout
https://coawstmodel.sourcerepo.com/coawstmodel/data/training_24feb2019/presentations .

To access only the 2019 meeting Zoom recordings use
svn checkout
https://coawstmodel.sourcerepo.com/coawstmodel/data/training_24feb2019/online .

We have some data files that are distributed to create tidal boundary conditions. That data is available using
svn checkout https://coawstmodel.sourcerepo.com/coawstmodel/data/tide .

We will be adding some of the larger applications to be distributed using
svn checkout https://coawstmodel.sourcerepo.com/coawstmodel/data/apps .

17. List of References and Acknowledgements.

Warner, J.C., Armstrong, B., He, R., and Zambon, J.B., (2010). Development of a Coupled Ocean-Atmosphere-Wave-Sediment Transport (COAWST) modeling system: Ocean Modeling, v. 35, no. 3, p. 230-244.

Kumar, N., Voulgaris, G., and Warner, J.C. (2011). Implementation and modification of a three-dimensional radiation stress formulation for surf zone and rip-current applications, Coastal Engineering, 58, 1097-1117, doi:10.1016/j.coastaleng.2011.06.009.

Olabarrieta, M., J. C. Warner, and N. Kumar (2011), Wave-current interaction in Willapa Bay, J. Geophys. Res., 116, C12014, doi:10.1029/2011JC007387.

Olabarrieta, M., Warner, J.C., and Armstrong, B. (2012). “Ocean-atmosphere dynamics during Hurricane Ida and Nor'Ida: an atmosphere-ocean-wave coupled modeling system application.” Ocean Modelling, 43-44, pp 112-137.

Kumar, N., Voulgaris, G., Warner, J.C., and M., Olabarrieta (2012). Implementation of a vortex force formalism in the coupled ocean-atmosphere-wave-sediment transport (COAWST) modeling system for inner-shelf and surf-zone applications. Ocean Modeling 47, pp 65-95.

Nelson, J. and R. He, (2012), Effect of the Gulf Stream on winter extratropical cyclone outbreaks, Atmosphere Research Letters, doi: 10.1002/asl.400.

Renault, L., J. Chiggiato, J. C. Warner, M. Gomez, G. Vizoso, and J. Tintoré (2012), Coupled atmosphere-ocean-wave simulations of a storm event over the Gulf of Lion and Balearic Sea, J. Geophys. Res., 117, C09019, doi:10.1029/2012JC007924.

Benetazzo, A., Carniel, S., Sclavo, M., and Bergamasco, A. (2013). Wave-current interaction: effect on the wave field in a semi-enclosed basin. Ocean Modeling, 70, 152-165.

Nelson, J. He, R., and Warner, J.C., Bane, J. (2014). Air-Sea Interactions during Strong Winter Extratropical Storms, Ocean Dynamics. doi:10.1007/s10236-014-0745-2.

Grifoll, M., A. L. Aretxabaleta, J. L. Pelegr□ı, M. Espino, J. C. Warner, and A. Sanchez-Arcilla (2014), Seasonal circulation over the Catalan inner-shelf (northwest Mediterranean Sea), J. Geophys. Res. Oceans, doi:10.1002/2014JC010187.

Zambon, J.B., He, R., and Warner, J.C. (2014). Investigation of Hurricane Ivan using the Coupled Ocean-Atmosphere-Wave-Sediment Transport (COAWST) Model, Ocean Dynamics, DOI 10.1007/s10236-014-0777-7.

Grifoll, M., Gracia, V., Aretxabaleta, A., Guillen, J., Espino, M., Warner, J.C. (2014). Formation of fine sediment deposit from a flash-flood river in the Mediterranean Sea. Journal of Geophysical Research, Oceans, 119, 5837-5853, doi:10.1002/2014JC010187.
Rong, Z., Hetland, R., Zhang, W., and Zhang, X. (2014). Current-wave interaction in the Mississippi-Atchafalaya river plume on the Texas-Louisiana shelf. Ocean Modelling, 84, 67-83.

Zambon, J.B., He, R., and Warner, J.C. (2014). Tropical to Extratropical: Marine Environmental Changes Associated with Superstorm Sandy Prior to its Landfall, Geophysical Research Letters, 41, doi:10.1002/2014GL061357.

M.J. Lewis, S.P. Neill, M.R. Hashemi, M. Reza (2014). Realistic wave conditions and their influence on quantifying the tidal stream energy resource, Applied Energy, Vol 136, pp 495-508, ISSN 0306-2619, http://dx.doi.org/10.1016/j.apenergy.2014.09.061.

Spydell, M. S., F. Feddersen, M. Olabarrieta, J. Chen, R. T. Guza, B. Raubenheimer, and S. Elgar (2015), Observed and modeled drifters at a tidal inlet, J. Geophys. Res. Oceans, 120, 4825–4844, doi:10.1002/2014JC010541.

M. Reza Hashemi, Simon P. Neill & Alan G. Davies (2015) A coupled tidewave model for the NW European shelf seas, Geophysical & Astrophysical Fluid Dynamics, 109:3, 234-253, DOI: 10.1080/03091929.2014.944909.

Danqin Ren, Jiantin Du, Feng Hua, Yongzeng Yang, Lei Han (2016). Analysis of different atmospheric physical parameterizations in COAWST modeling system for the Tropical Storm Nock-ten application. Natural Hazards. 1-18, 10.1007/s11069-016-2225-0. http://www.tandfonline.com/doi/abs/10.1080/03091929.2014.944909

Carniel, S., Benetazzo, A., Bonaldo, D., Falcieri, F.M., Miglietta, M.M., Ricchi, A., and Sclavo, M. (2016). Scratching beneath the surface while coupling atmosphere, ocean and waves: Analysis of a dense water formation event. Ocean Modeling, 101, 101-112.

Ricchi, A., Miglietta, M., Falco, P., Benetazzo, A., Bonaldo, D., Bergamasco, A., Sclavo, M., Carniel, S. (2016). On the use of a coupled ocean–atmosphere-wave model during an extreme Cold Air Outbreak over the Adriatic Sea. Atmospheric Research. 172. 10.1016/j.atmosres.2015.12.023.

Beudin, A., Kalra, T. S., Ganju, N. K., and Warner, J.C. (2016). Development of a coupled wave-flow-vegetation interaction module. Computers and Geosciences, http://dx.doi.org/10.1016/j.cageo.2016.12.010.

Warner, J.C. (2016). “Advanced Model Training for Predicting Coastal Storm Impacts.” http://soundwaves.usgs.gov/2016/09/meetings.html

Bruneau, N., and Toumi, R. (2016). A fully-coupled atmosphere-ocean-wave model of the Caspian Sea. Ocean Modeling, 107, 97-111.

Feddersen, F., M. Olabarrieta, R. T. Guza, D. Winters, B. Raubenheimer, and S. Elgar (2016), Observations and modeling of a tidal inlet dye tracer plume, J. Geophys. Res. Oceans, 121, 7819–7844, doi:10.1002/2016JC011922.

Safak, I., List, J.H., and Warner, J.C. (2016). Barrier island breach evolution: Alongshore transport and bay-ocean pressure gradient interactions. J. Geophys. Res. Oceans, 121,doi:10.1002/2016JC012029.

Safak, I., List, J.H., Warner, J.C., and Kumar, N. (2017.) Observations and 3D hydrodynamics-based modeling of decadal-scale shoreline change along the Outer Banks, North Carolina, Coastal Engineering, 120, 78-92.

Warner, J.C., Schwab, W.C., List, J.H., Safak, I., Liste, M., and Baldwin, W. (2017). Inner-shelf ocean dynamics and seafloor morphologic changes during Hurricane Sandy, Continental Shelf Research, 138, 1-18.

Beudin, A., Ganju, N. K., Defne, Z. and Aretxabaleta, A. L. (2017), Physical response of a back-barrier estuary to a post-tropical cyclone. J. Geophys. Res. Oceans. doi:10.1002/2016JC012344

Akan, Ç., S. Moghimi, H. T. Özkan-Haller, J. Osborne, and A. Kurapov (2017), On the dynamics of the Mouth of the Columbia River: Results from a three-dimensional fully coupled wave-current interaction model, J. Geophys. Res. Oceans, 122, 5218–5236, doi:10.1002/2016JC012307.

Beudin, A., N. K. Ganju, Z. Defne, and A. L. Aretxabaleta (2017), Physical response of a back-barrier estuary to a post-tropical cyclone, J. Geophys. Res. Oceans, 122, 5888–5904, doi:10.1002/2016JC012344.

Ricchi, A.; Miglietta, M.M.; Barbariol, F.; Benetazzo, A.; Bergamasco, A.; Bonaldo, D.; Cassardo, C.; Falcieri, F.M.; Modugno, G.; Russo, A.; Sclavo, M.; Carniel, S. Sensitivity of a Mediterranean Tropical-Like Cyclone to Different Model Configurations and Coupling Strategies. Atmosphere 2017, 8, 92.

Zhao, X. and Chan, J. C. L. (2017), Effect of the Initial Vortex Size on Intensity Change in the WRF-ROMS Coupled Model. J. Geophys. Res. Oceans. Accepted Author Manuscript. doi:10.1002/2017JC013283.

Justin S. Rogers, Stephen G. Monismith, Oliver B. Fringer, David A. Koweek, Robert B. Dunbar (2017). A coupled wave-hydrodynamic model of an atoll with high friction: Mechanisms for flow, connectivity, and ecological implications, Ocean Modelling, 110, p 66-82, https://doi.org/10.1016/j.ocemod.2016.12.012.

Kukulka, T., Jenkins, R. L., Kirby, J. T., Shi, F., & Scarborough, R. W. (2017). Surface wave dynamics in Delaware Bay and its adjacent coastal shelf. Journal of Geophysical Research: Oceans, 122, 8683–8706. https://doi.org/10.1002/2017JC013370.

Safak, I., List, J. H., Warner, J. C., & Schwab, W. C. (2017). Persistent shoreline shape induced from offshore geologic framework: Effects of shoreface connected ridges. Journal of Geophysical Research: Oceans, 122, 8721–8738. https://doi.org/10.1002/2017JC012808.

Wandres, M., Wijeratne, E. M. S., Cosoli, S., & Pattiaratchi, C. (2017). The effect of the Leeuwin Current on offshore surface gravity waves in southwest western Australia. Journal of Geophysical Research: Oceans, 122, 9047–9067. https://doi.org/10.1002/2017JC013006.

Çiğdem Akan, James C. McWilliams, Saeed Moghimi, H. Tuba Özkan-Haller. (2018) Frontal dynamics at the edge of the Columbia River plume, Ocean Modelling, Volume 122, pp 1-12, ISSN 1463-5003, https://doi.org/10.1016/j.ocemod.2017.12.001.

J.M.R. Alves, A. Peliz, R.M.A. Caldeira, P.M.A. Miranda. (2018). Atmosphere-ocean feedbacks in a coastal upwelling system, Ocean Modelling, 123, pp 55-65, https://doi.org/10.1016/j.ocemod.2018.01.004.

Zhengchen Zang, Z. George Xue, Shaowu Bao, Qin Chen, Nan D. Walker, Alaric S. Haag, Qian Ge, Zhigang Yao, (2018). Numerical study of sediment dynamics during hurricane Gustav, Ocean Modelling, Volume 126, p 29-42. https://doi.org/10.1016/j.ocemod.2018.04.002.

Dufois, F., Lowe, R., Rayson, M., & Branson, P. (2018). A numerical study of tropical cyclone-induced sediment dynamics on the Australian North West Shelf. Journal of Geophysical Research: Oceans, 123. https://doi.org/10.1029/2018JC013939.

Wenping Gong, Zhongyuan Lin, Yunzhen Chen, Zhaoyun Chen, and Heng Zhang, (2018). Effect of winds and waves on salt intrusion in the Pearl River estuary, Ocean Science, 14, 139-159.

Torres-Garcia, L. M., Dalyander, P. S., Long, J. W., Zawada, D. G., Yates, K. K., Moore, C., & Olabarrieta, M. (2018). Hydrodynamics and sediment mobility processes over a

degraded senile coral reef. Journal of Geophysical Research: Oceans, 123. https://doi.org/10.1029/2018JC013892

Gong, Wenping & Chen, Yunzhen & Zhang, Heng & Chen, Zhaoyun. (2018). Effects of Wave–Current Interaction on Salt Intrusion During a Typhoon Event in a Highly Stratified Estuary. Estuaries and Coasts. 1-20. 10.1007/s12237-018-0393-8.

Lewis, M.J., Palmer, T., Hashemi, R. et al. (2019). Wave-tide interaction modulates nearshore wave height. Ocean Dynamics, 69: 367. https://doi.org/10.1007/s10236-018-01245-z

Shi, Q.; Bourassa, M.A. Coupling Ocean Currents and Waves with Wind Stress over the Gulf Stream. Remote Sens. 2019, 11, 1476. https://doi.org/10.3390/rs11121476

Dmitry Dukhovskoy, 2019, oil spill simulations. https://www.youtube.com/watch?v=XWfP9IxUawk

Tarpley, D.R.N., Harris, C., Friedrichs, C., and Sherwood, C.R. (2019). Tidal Variation in Cohesive Sediment Distribution and Sensitivity to Flocculation and Bed Consolidation in An Idealized, Partially Mixed Estuary, J. Mar. Sci. Eng. 2019, 7(10), 334; https://doi.org/10.3390/jmse7100334.

Kalra, T., Li, X., Warner, J.C., Geyer, W.R., and Wu, H., 2019, Comparison of physical to numerical mixing with different tracer advection schemes in estuarine environments, Journal of Marine Science and Engineering, 7, 338.

Wang, A., Ralston, D., Bi, N., Cheng, Z., Wu, X., and Wang, H., 2019, Seasonal variation in sediment transport and deposition on a muddy clinoform in the Yellow Sea, Continental Shelf Research 179, 37-51.

Jia, Y., Whitney, M.M., 2019, Summertime Connecticut River Water Pathways and Wind Impacts, Journal of Geophysical Research: Oceans, 124, 1897-1914

Chen, L., Gong, W., Scully, M. E., Zhang, H., Cheng, W., & Li, W. (2020). Axial wind effects on stratification and longitudinal sediment transport in a convergent estuary during wet season. Journal of Geophysical Research: Oceans, 125, e2019JC015254. https://doi.org/10.1029/2019JC015254

Anandh, T.S., Das, B.K., Kuttippurath, J. et al. A coupled model analyses on the interaction between oceanic eddies and tropical cyclones over the Bay of Bengal. Ocean Dynamics 70, 327–337 (2020). https://doi.org/10.1007/s10236-019-01330-x

Rajesh Kumar R, Sandeepan BS & Holland, D.M. Impact of different sea surface roughness on surface gravity waves using a coupled atmosphere–wave model: a case of Hurricane Isaac (2012). Ocean Dynamics 70, 421–433 (2020). https://doi.org/10.1007/s10236-019-01327-6

Forecast systems:
Hydro and Agro Informatics Institute
http://www.thaiwater.net/v3/wrfroms/rain_forecast_pre/tab1/image1

USGS Woods Hole
http://woodshole.er.usgs.gov/project-pages/cccp/public/COAWST.htm

North Carolina State University
http://omgsrv1.meas.ncsu.edu:8080/ocean-circulation-useast2/

Madeira Island Oceanic Observatory forecasting
https://oom.arditi.pt/mission072018/

Acknowledgements
We thank all the modeling and tool systems for open access to their codes, and to the Integration and Application Network (ian.umces.edu/symbols), University of Maryland Center for Environmental Science, for the courtesy use of their symbols and diagrams.