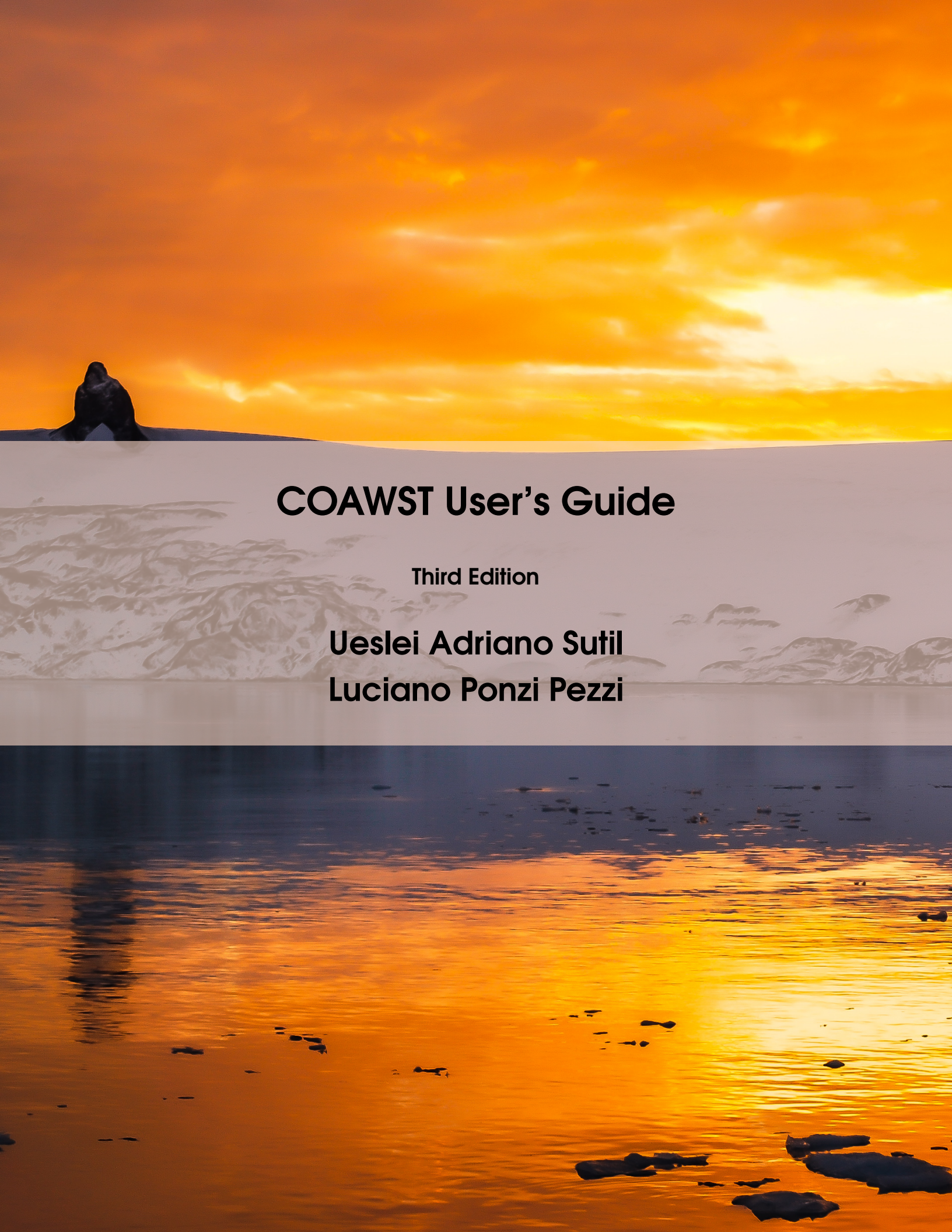


COAWST User's Guide

Third Edition

Ueslei Adriano Sutil
Luciano Ponzi Pezzi

Ocean and Atmosphere Studies Laboratory
OBT/INPE

UESLEI ADRIANO SUTIL
uesleisutil1@gmail.com
https://www.uesleisutil.com.br

LUCIANO PONZI PEZZI
luciano.pezzi@inpe.br

This guide is a contribution from the Ocean and Atmosphere Studies Laboratory (LOA) to the scientific community. It was funded by the Institutional Training Scholarship from the National Institute for Space Research, through the Ministry of Science, Technology and Innovation and the National Council for Scientific and Technological Development (MCTI/CNPq) **process 301110/2017**.

The image on the cover was taken by the first author during his second cruise to Antarctica, in October 2017. It is licensed under a *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License*.

The image on the header of each chapter was taken by the first author during his first cruise to Antarctica, at Brazilian Antarctic Station Commander Ferraz, in October 2016. It is licensed under a *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License*.

The template *Legrand Orange Book* was produced by Mathias Legrand (*legrand.mathias@gmail.com*) and modified by Vel (*vel@latextemplates.com*) and Andrea Hidalgo (*andrea@inaoep.mx*). It is licensed under a *Creative Commons Attribution-NonCommercial 3.0 Unported*.

The *model2roms* toolbox (*https://github.com/trondkr/model2roms*) was created by Trond Kristiansen (*me@trondkristiansen.com*) and is licensed under a *MIT License*.

Third edition
September 2020

Reviewers

The authors would like to thank the following peers for reviewing the **COAWST User's Guide - First Edition**:

- MSc. Eliana Bertol Rosa
  CV Lattes: *http://lattes.cnpq.br/4723025083192543*

- MSc. Mylene Cabrera
  CV Lattes: *http://lattes.cnpq.br/1575145014336724*

The authors would like to thank the following peers for reviewing the **COAWST User's Guide - Second Edition**:

- Dr. Douglas Francisco Marcolino Gherardi
  CV Lattes: *http://lattes.cnpq.br/5421394642444587*

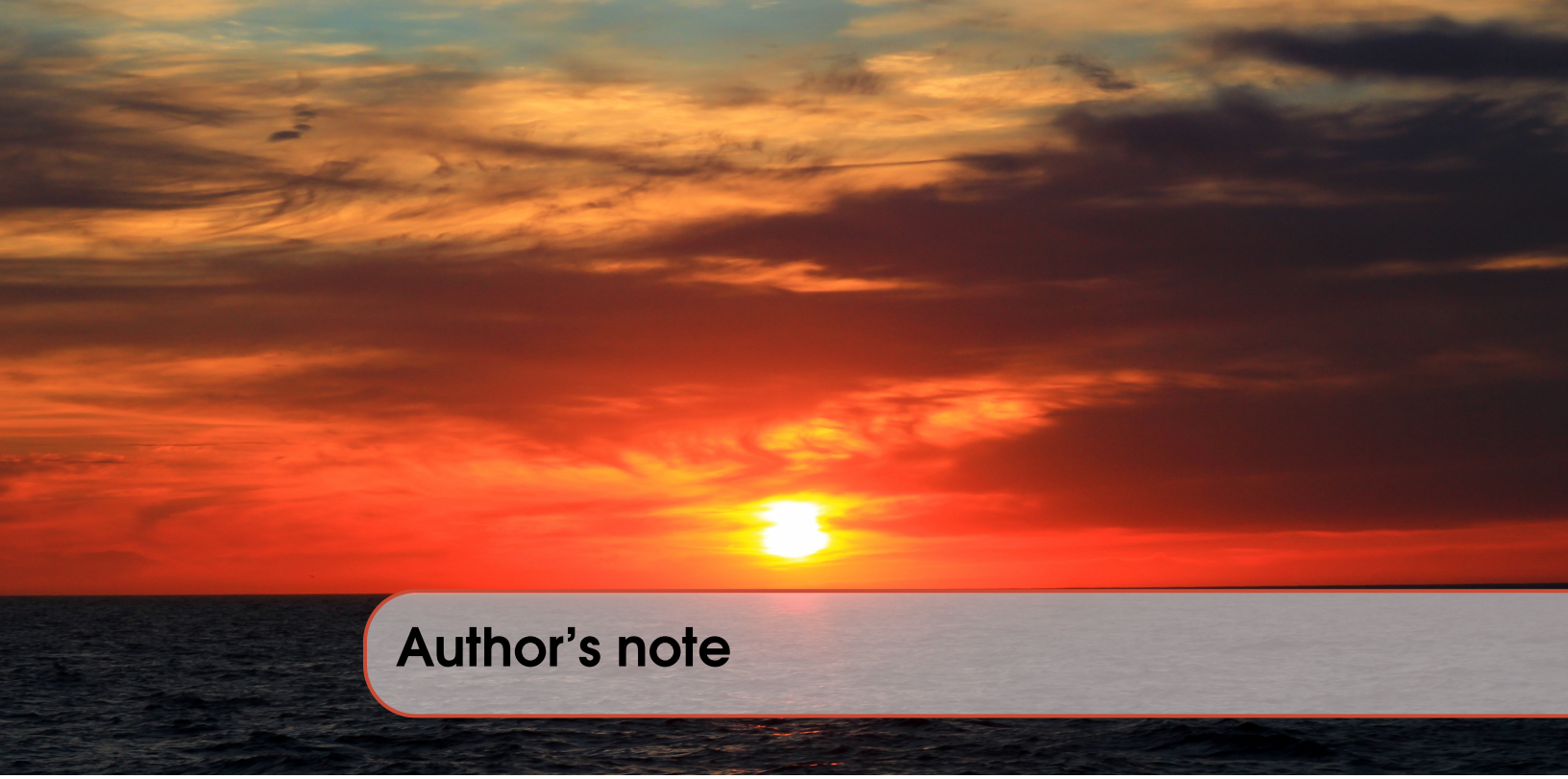
- Dr. Jonas Takeo Carvalho
  CV Lattes: *http://lattes.cnpq.br/8827254187143196*

- MSc. Clarissa Akemi Kajiya Endo
  CV Lattes: *http://lattes.cnpq.br/7557267210025953*

- MSc. Giullian Nicola Lima dos Reis
  CV Lattes: *http://lattes.cnpq.br/9263946357414407*

- MSc. Mylene Cabrera
  CV Lattes: *http://lattes.cnpq.br/1575145014336724*

Thank you.



Author's note

This guide is designed to assist new users to configure and use the Coupled Ocean-Atmosphere-Wave-Sediment Transport System (COAWST). The main idea behind this guide is to teach the necessary steps to use COAWST, starting with its installation, then a simulation of a test case and then, the configuration of your own project. To achieve this goal, we use several programming languages, such as Fortran, Python and MATLAB. In the future we intend to adapt all scripts to a free programming language.

When we started writing this guide, we wanted to pass on our expertise of using a numerical modeling system that is considered the state of the art in our field, throughout reading, understanding how it works and how to use the COAWST, allying theory with practice.

However, a major difficulty in this process was the generation of the boundary and initial conditions for the oceanic model, the Regional Ocean Modeling System (ROMS), which rely on paid software. To get around this problem we chosed to work with the *model2roms* toolbox package. This set of routines was developed in Python and Fortran language by Trond Kristiansen (*http://www.trondkristiansen.com*).

We emphasize that in some chapters, the reader will find how to use the COAWST in a cluster that is available for use by the Ocean and Atmosphere Studies Laboratory (LOA) of the National Institute for Space Research (INPE). This is a high performance computing system that allows to use softwares on parallel computing processes. Therefore, this guide will serve as an inspiration for COAWST users to implement their projects on their own computational systems.

To cite the **third edition**, use the following:

UnderConstruction

To cite the **first edition**, use the following:

SUTIL, U. A.; PEZZI, L. P. Guia prático para utilização do COAWST. São José dos Campos: INPE, 2018. 86 p. IBI: <8JMKD3MGP3W34R/3RQSQ2L>. ISBN: <978-85-17-00093-5>. Avaliable at: <http://urlib.net/rep/8JMKD3MGP3W34R/3RQSQ2L>.

To cite the **second edition**, use the following:

SUTIL, U. A.; PEZZI, L. P. Guia prático para utilização do COAWST - 2ª Edição. São José dos Campos: INPE, 2019. 100 p. IBI: <8JMKD3MGPCW/3DT298SL>. ISBN: <978-85-17-00098-0>. Avaliable at: <http://urlib.net/rep/8JMKD3MGP3W34R/3TUTUJB>.

We wish you a good reading and success in your research.

The authors.

Contents

1. Introduction

This guide provides a brief introduction to the models that make up the Coupled Ocean-Atmosphere-Wave-Sediment Transport Modeling System (COAWST), as well as the additional user settings required for specific projects. As shown in Figure 1.1, COAWST uses several models and programs that will be presented below:

- **COAWST**: Core of the coupled numerical modeling system. More information in the section 1.7;
- **ROMS**: The hydrodynamic model. More information in the Section 1.1;
- **Sea Ice**: The sea ice model, coupled to ROMS. In this guide we use the Budgell's Sea Ice Model. More information in the Section 1.2;
- **WRF**: The atmospheric model. More information in the Section 1.3;
- ***wrf***: Executable program to start the WRF atmospheric simulation. More information in the section 4.4;
- ***real***: Executable program to generate the initial condition and the boundary forces of the WRF. More information in the Section 4.3.5;
- ***WPS***: Package with three programs to generate the files to be used in *real*. More information in the Section 4.2;
- ***geogrid***: Program to, mainly, generate the WRF grid domain. More information in the section 4.3.1;
- ***ungrib***: Briefly, the program that extracts the data in *GRIB* format. More information in the Section 4.3.3;
- ***metgrid***: Briefly, the program that interpolates the data generated by *ungrib*. More information in the Section 4.3.4;
- **SWAN**: The wave model. More information in the Section 1.4;
- **MCT**: The set of codes that couples the previous models. More information in the Section 1.5;
- **SCRIP**: Package that interpolates and remaps the model's grids to allow the models to be coupled. More information in the section 1.6.

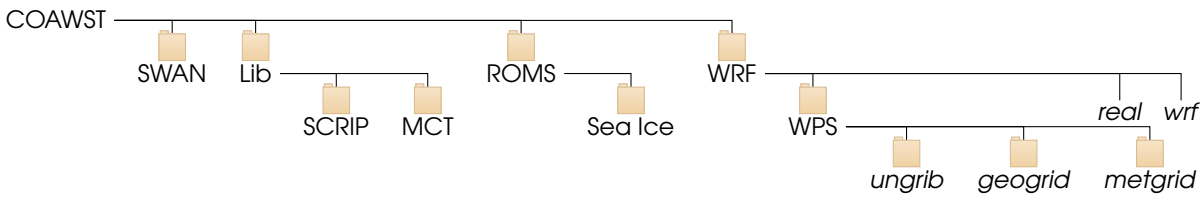


Figure 1.1. COAWST folder structure.

When writing this guide, we chosed to use Python as a programming language for some steps. It is a programming language designed with a philosophy of emphasizing the importance of the programmer's effort over the computational effort and prioritizes code readability over speed or expressiveness. The language has a wide and active community, which facilitates the search information by the user, as there is an extensive collection of libraries and documents on the internet.

Python (*http://python.org*) offers great help for beginners, providing introductions about the language and also a guide to use Python for scienfic purposes.

We also use MATLAB as a language. It is a paid, but high-performance interactive software focused on the numerical calculation. The MATLAB Answers website is a platform created to offer help to the users about the language. The site is avaliable at *https://www.mathworks.com/matlabcentral/answers/index*.

1.1 Regional Ocean Modeling System

The Regional Ocean Modeling System (ROMS; **Shchepetkin2005**) is a three-dimensional ocean model with free surface, vertical sigma coordinate with sigma vertical coordinates (that follow the terrain) and solve primitive equations. The model uses the Reynolds average and the finite difference method to solve the Navier-Stokes equations using hydrostatic and Boussinesq approximations (**Haidvogel2008**).

The hydrostatic equations of momentum use a split-explicit time-step scheme, where the barotropic and baroclinic modes are solved separately, in different finite numbers of steps of time, to solve the free surface equations and it is integrated vertically. This structure separate time-steps frames maintains the volume conservation and consistency preservation that are necessary for the tracers (**Shchepetkin2005**; **Haidvogel2008**).

The model solves the horizontally equations through orthogonal curvilinear coordinates of the Arakawa-C grid type (**Arakawa1977**). Vertically, the coordinates follow the features of the terrain and allow you to adjust the resolution along the water column. To guarantee the conservation of momentum, the grid uses econd order finite differences (**Haidvogel2008**).

ROMS is a model that has free code and its development has the contribution of the user community. Currently, the version used in COAWST is managed by Dr. Hernan Arango of Rutgers University. To access the model code, is necessary to register on the ROMS website (*https://www.myroms.org/*) The site has a extremely useful and very active forum to discuss about questions and suggestions. You can access here: *https://www.myroms.org/forum*.

We recommend to read the ROMS Technical Manual, written by **hedstrom2018**. This manual has several information about the equations and algorithms of the model and examples of test cases.

1.2 Budgell's Sea Ice Model

The Sea Ice Model, proposed by **Budgell2005**, has the same time and grid steps as the ROMS model and shares the same parallel encoding structure for use with Message Passing Import (MPI). Thus, allows dynamic and thermodynamic modeling where sea ice predominates, such as at high latitudes.

The main attributes of the model, according to **hedstrom2018**, are:

- **Hunke1997** and **Hunke2001** elasctic-viscous-plastic dynamics;
- **Mellor1989** thermodynamics;
- Orthogonal-curvilinear coordinates;
- **Arakawa1977** grid;
- **Smolarkiewicz1990** advection of tracers;
- **Lemieux2015** landfast ice parameterization.

1.3 Weather Research & Forecasting Model

The Weather Research and Forecasting (WRF; **Skamarock2008**) is a model developed by the National Centers for nvironmental Prediction (NCEP), the National Center for Atmospheric Research (NCAR) and research groups from different universities.

To integrate the governing equations over time, the Advanced Research WRF (ARW) uses low frequency modes that are integrated using the third-order Runge-Kutta scheme, and the integrated acoustic and gravity (high frequency) modes with a lower time step. By this way, the numerical stability is maintained through a "forward-backward" scheme for acoustic modes that propagate horizontally and an implicit scheme for acoustic modes for vertical propagation and buoyancy oscillations (**Skamarock2008**).

The WRF model uses an Arakawa-C type grid (**Arakawa1977**), where normal speeds are staggered halfway through the grid of thermodynamic variables, as shown in the schematic representation illustrated in Figure 1.2.

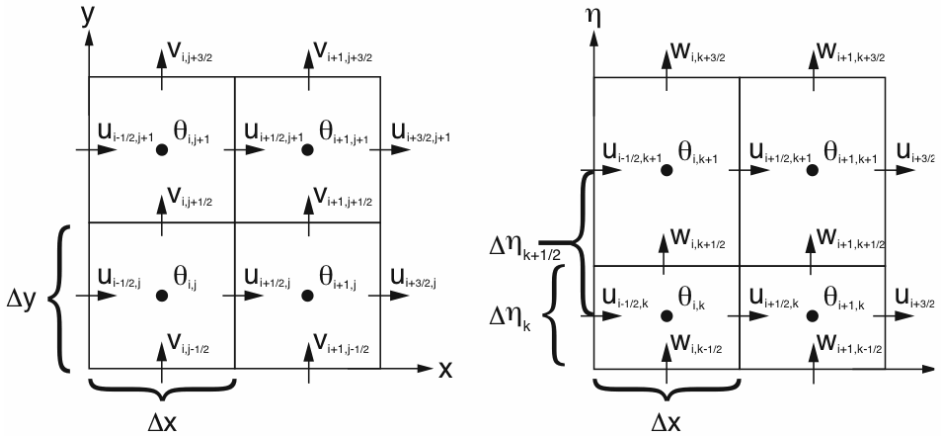


Figure 1.2. Horizontal and vertical grid of Weather Research and Forecast (WRF) using the Arawaka-C grid. The horizontal and vertical components of velocity (**u**, **v** and **w**) are positioned along the faces of the grids and the thermodynamic variables (θ) are positioned in the center of each grid.
Author: **Skamarock2008**.

It is important to note that the WRF, without coupling with other models, simulates the surface roughness based on the ratio of roughness to wind shear proposed by **Charnock1955**, as exemplified in the following equation 1.1:

$$Z_0 = Z_{ch}\frac{u_*^2}{g} \tag{1.1}$$

Where Z_0 é a roughness, Z_{ch} is the Charnock parameter (a dimensionless value of 0,018), u_* the frictional speed (m/s) and g the gravity acceleration(9,81 m/s^2).

The WRF is avaliable at: *http://www2.mmm.ucar.edu/wrf/users/download/get_source.html*.

1.4 Simulating Waves Nearshore

The Simulating Waves Nearshore (SWAN; **Booij1999**; **Booij1996**) is a third generation model, designed to simulate coastal regions with shallow waters and local currents. The model is widely used in the numerical forecast of the Simulating Waves Nearshore (SWAN; **Booij1999**; **Booij1996**) is a third generation model, designed to compute in coastal regions with shallow waters and local currents. The model is widely used in the numerical forecast of waves in coastal regions, estuaries, channels and others, being able to use fields of wind, bathymetry and currents provided by other models (**Booij1999**; **Booij1996**).

Dasilva2013 and **Booij1999**; **Booij1996** list the main characteristics of the SWAN:

- Wave propagation in time and space, shoaling, refraction due to current and depth, frequency shifting due to currents and non-stationary depth;
- Wave generation by wind;
- Whitecapping, bottom friction and depth-induced breaking;
- Dissipation due to aquatic vegetation, turbulent flow and viscous fluid mud;
- Wave-induced set-up;
- Propagation from laboratory up to global scales;
- Transmission through and reflection (specular and diffuse) against obstacles;
- Diffraction.

More features can be found in SWAN website, avaliable at: *http://swanmodel.sourceforge.net/*.

1.5 Model Coupling Toolkit

The Model Coupling Toolkip (MCT; **Larson2005**; **Jacob2005**; **Warner2008**) is a set of open-source scripts, written in Fortran90 that allow the transmission and transformation of the different data necessary for model coupling. During initialization, model domains are broken down into segments that are distributed between processors, allowing models to be coupled also in parallel.

According to the MCT website, (*http://www.mcs.anl.gov/research/projects/mct/*), the toolkit provides the following core coupling services:

- A component model registry;
- Domain decomposition descriptors;
- A time averaging and accumulation buffer datatype;
- A general spatial grid representation capable of supporting unstructured grids;
- Parallel tools for intergrid interpolation;
- Tools for merging data from multiple components for use by another component;
- A programming model similar to that of the Message Passing Interface.

1.6 Spherical Coordinate Remapping Interpolation Package

O Spherical Coordinate Remapping Interpolation Package (SCRIP; **Jones1999**; **Jones1998**) is freely available for download at *https://github.com/SCRIP-Project/SCRIP*. The package is distributed together with COAWST modeling system. This package is used for projects that use more than one model and with different grids (with different spatial resolutions). SCRIP will generate the interpolation weights that will be used to remap the data between the different grids of the different models.

In COAWST, SCRIP was modified to generate a single file (in NetCDF format) that contains the weights based in each model grid.

1.7 Coupled-Ocean-Atmosphere-Wave-Sediment Transport Modeling System

The COAWST ()**Warner2010**; **Warner2008**) uses the WRF as the atmospheric model, the ROMS as the hydrodynamic model, the SWAN as the wave model and the sediment transport model Community Sediment Transport Modeling Project (CSTM; **Warner2008**), each one coupled by the MCT (**Warner2010**; **Warner2008**). The frequency with which this information is exchanged between the different models is adjusted by the user.

The coupling between the models allows the different physical processes that occur in both oceanic and atmospheric environments to be identified and analyzed with greater accuracy when compared to simulations without active coupling.(**Pullen2018**; **Miller2018**).

> **WARNING** This guide does not use CSTM. In case you are interested, there is a study on the transfer of sediments during Hurricane Isabel (2003) by **Warner2010**.

As shown in Figure 1.3, the informations exchanged between models are:

- WRF -> ROMS: surface shear and liquid heat fluxes (calculated in ROMS from the components of latent and sensitive heat fluxes) shortwave and longwave radiation, atmospheric pressure, relative humidity, air temperature, clouds, precipitation and wind components ;
- ROMS -> WRF: sea surface temperature;
- SWAN -> ROMS: surface and bottom wave direction, height, length, period, energy dissipation and lower orbital speed;
- ROMS -> SWAN: bathymetry, surface elevation, height of the sea and average currents in depth;

- SWAN -> WRF: roughness of the sea surface (calculated in WRF from the significant wave height, length and period);
- WRF -> SWAN: wind at 10m.

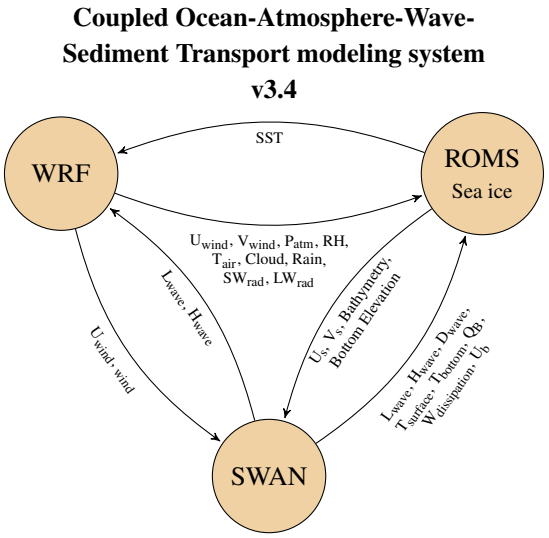


Figure 1.3. Diagram about the exchange of information between the main models that make up the COAWST modeling system. (**Warner2008**)

The Woods Hole Coastal and Marine Science Center page provides an experimental presentation on real time of COAWST (Sea Surface Temperature, Sea Surface Height, Significant Height Wave, Current and Wind Vectors and Sediment Dispersion) in the eastern United States and Gulf of Mexico. The content is available at: *https://woodshole.er.usgs.gov/project-pages/cccp/public/COAWST.htm*.

1.8 Requirements to use this guide

To compile COAWST, this guide uses a cluster with parallel communication and a nonparallel Linux OS with Gfortran. If you choose to compile in on a computer with different specifications, use the original COAWST user manual as a reference, available in the modeling system repository. In this case, see how to download COAWST in the 2.6 section.

We use **Ubuntu 19.04 LTS**. It is important to maintain the same systems operational, as other version may conflict with some libraries used by the *model2roms* toolkit or even the COAWST itself.

WARNING This guide uses COAWST v. 3.6.

2. COAWST in a parallel cluster

2.1 About our cluster Kerana

The CRAY XE machine, also called Kerana, is a cluster with massively parallel architecture, with 84 processing nodes and 2688 cores and is located at CPTEC/INPE, in Cachoeira Paulista, São Paulo. For having the ability to parallelize operations through the MPI interface, the cluster is ideal for using numerical models with high spatial resolution and temporal.

2.2 Signing up a user account

To start the process of applying for a user account in Kerana, it is necessary that the computer in the INPE's premises have a fixed IP. Your advisor or supervisor is required to send an email to *Helpdesk* (*helpdesk.cptec@inpe.br*) informing the MAC address, hostname and reason for the request. If the computer already has a fixed IP assigned, it also need to be informed to be changed.

With the fixed IP configured, contact the *Helpdesk* (*helpdesk.cptec@inpe.br*) to request a form to use Kerana.

2.3 Signing in to the Kerana cluster

Access will be done entirely through the computer terminal. You will need two primary commands: *ssh* for accessing and manipulating files and folders and *sftp* to download and upload files.

To access and modify files and folders, type in the terminal, replacing *name.surname* with the user provided by *Helpdesk*:

WARNING From now on, whenever the guide shows the user as *name.surname*, change to your username provided by the Helpdesk.

```
ssh -Y name.surname@acesso-hpc.cptec.inpe.br -p 2000
```

To download and/or upload, use:

```
sftp -P2000 name.surname@acesso-hpc.cptec.inpe.br
```

WARNING It is not possible to download and upload multiple files at the same time using *sftp*. One tip is to compress into a single *tar* file and then unzip them.

To add files from your computer to Kerana:

```
put file.tar.gz
```

To extract Kerana files to your computer:

```
get file.tar.gz
```

2.4 File repository

Certain files are needed in each user's area to facilitate the use of the cluster. You will find them in the directory:

```
/scratch/adriano.sutil/repositorio/
```

To copy the files to your area, type:

```
cp -r /scratch/adriano.sutil/repositorio /scratch/name.surname
```

WARNING From now on this guide will use the files that are inside this repository, so it is essential that they are in your area.

2.5 Kerana environment

It is necessary to activate some modules in the cluster to compile COAWST. In this case, open the file *.bashrc* which is located at the root directory of your user.

```
vim .bashrc
```

Add the following commands at the end of the file, changing only *name.surname* to your username:

```
module load java
module load netcdf

export PATH=/scratch/name.surname/repositorio/Softs/nedit/5.5:$PATH
export PATH=/scratch/name.surname/repositorio/Softs/bin:$PATH

export PHDF5=${HDF_DIR}
export WRFIO_NCD_LARGE_FILE_SUPPORT=1

export PATH=/home/luciano.pezzi/local/bin:$PATH
export JASPERINC=/home/luciano.pezzi/local/include
export JASPERLIB=/home/luciano.pezzi/local/lib
export LD_LIBRARY_PATH=/home/luciano.pezzi/local/lib:$LD_LIBRARY_PATH
```

Save and type in the terminal:

```
source .bashrc
```

2.6 Downloading COAWST

WARNING COAWST v3.6 is already in the repository within the Kerana cluster, as discussed in Section 2.4.

To download COAWST, send an email to Dr. John Warner (*jcwarner@usgs.gov*), one of the heads behind the coupled regional modeling system.

After access is granted, with the user credentials and password provided by Dr. John Warner, type in the command the command below, changing the textit myusrname to your username.

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

Add the COAWST folder to your Kerana desktop using *sftp*, as in the Section 2.3.

2.7 Automating the compilation of COAWST in Kerana

> **WARNING** This guide uses COAWST v3.6.

To speed up the process, it is possible to automate some compilation steps. Enter the directory:

```
cd /scratch/name.surname/COAWST/WRF/arch
```

Open the file *Config.pl*:

```
nedit Config.pl
```

Look for the lines:

```
printf "\nEnter selection [%d-%d] : ",1,$opt ;
$response = <STDIN> ;
```

And replace <*STDIN*> to 42, as in the example below:

```
printf "\nEnter selection [%d-%d] : ",1,$opt ;
$response = 42 ;
```

Open the file *configure.defaults*:

```
nedit configure.defaults
```

Look for the *TRADEFLAG*, option, which is on line 1262, approximately:

```
TRADEFLAG = CONFIGURE_TRADEFLAG
```

And modify to:

```
TRADEFLAG = -traditional
```

These modifications will select the configurations of the Kerana cluster (*CRAY CCE (ftn/gcc): Cray XE and XC (dmpar)*), among those available to use COAWST, as shown in Figure 2.1.

```
  1. (serial)   2. (smpar)   3. (dmpar)   4. (dm+sm)   PGI (pgf90/gcc)
  5. (serial)   6. (smpar)   7. (dmpar)   8. (dm+sm)   PGI (pgf90/pgcc): SGI MPT
  9. (serial)  10. (smpar)  11. (dmpar)  12. (dm+sm)   PGI (pgf90/gcc): PGI accelerator
 13. (serial)  14. (smpar)  15. (dmpar)  16. (dm+sm)   INTEL (ifort/icc)
                                         17. (dm+sm)   INTEL (ifort/icc): Xeon Phi (MIC architecture)
 18. (serial)  19. (smpar)  20. (dmpar)  21. (dm+sm)   INTEL (ifort/icc): Xeon (SNB with AVX mods)
 22. (serial)  23. (smpar)  24. (dmpar)  25. (dm+sm)   INTEL (ifort/icc): SGI MPT
 26. (serial)  27. (smpar)  28. (dmpar)  29. (dm+sm)   INTEL (ifort/icc): IBM POE
 30. (serial)               31. (dmpar)                PATHSCALE (pathf90/pathcc)
 32. (serial)  33. (smpar)  34. (dmpar)  35. (dm+sm)   GNU (gfortran/gcc)
 36. (serial)  37. (smpar)  38. (dmpar)  39. (dm+sm)   IBM (xlf90_r/cc_r)
 40. (serial)  41. (smpar)  42. (dmpar)  43. (dm+sm)   PGI (ftn/gcc): Cray XC CLE
 44. (serial)  45. (smpar)  46. (dmpar)  47. (dm+sm)   CRAY CCE (ftn/gcc): Cray XE and XC
 48. (serial)  49. (smpar)  50. (dmpar)  51. (dm+sm)   INTEL (ftn/icc): Cray XC
 52. (serial)  53. (smpar)  54. (dmpar)  55. (dm+sm)   PGI (pgf90/pgcc)
 56. (serial)  57. (smpar)  58. (dmpar)  59. (dm+sm)   PGI (pgf90/gcc): -f90=pgf90
 60. (serial)  61. (smpar)  62. (dmpar)  63. (dm+sm)   PGI (pgf90/pgcc): -f90=pgf90
```

Figure 2.1. Computational options available for selection when compiling COAWST.

Now, open the *Config_new.pl* file and then look for the following lines:

```
printf "Compile for nesting? (1=basic, 2=preset moves, 3=vortex following) [default 1]: " ;
}
$response = <STDIN> ;
```

And change *<STDIN>* to the basic WRF atmospheric model nesting mode, as in the following example:

```
printf "Compile for nesting? (1=basic, 2=preset moves, 3=vortex following) [default 1]: " ;
}
$response = 1 ;
```

2.8 Compiling the MCT

WARNING This guide uses COAWST v 3.6.

Each new user must compile the MCT before compiling COAWST. The first step is to use the *setup_pgi.sh* file. This file was copied from the previous repository root, it is essential to change the directories contained therein. So open the file:

```
nedit setup_pgi.sh
```

If necessary, change the directories according to the name of your COAWST folder and execute the file to load the necessary modules:

```
source setup_pgi.sh
```

The libraries will be activated, as shown in Figure 2.2:

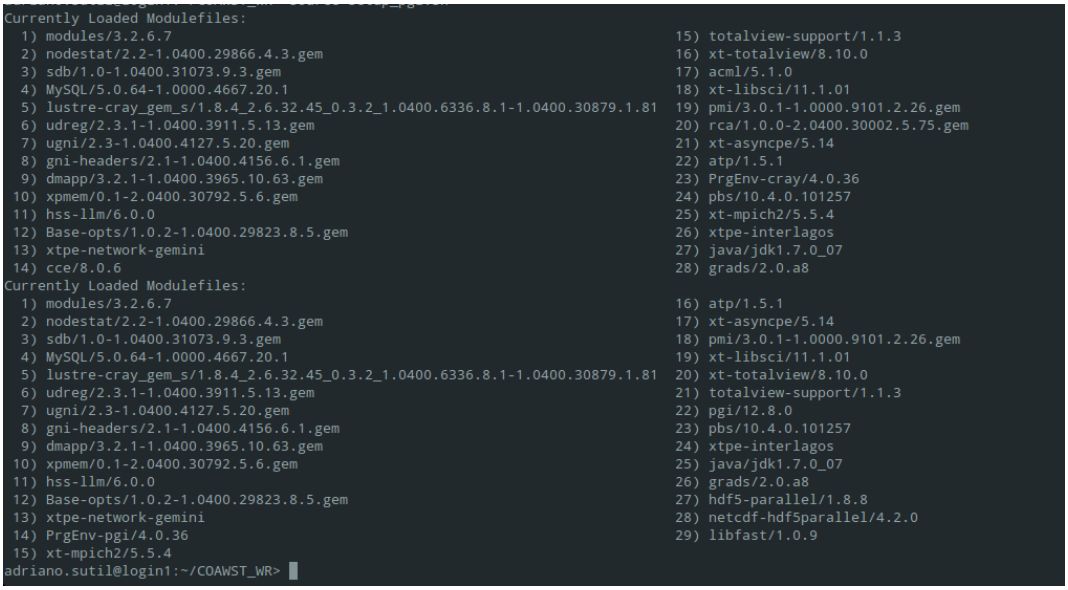


Figure 2.2. Modules activated in the cluster with the file *setup_pgi.sh* with user adriano.sutil.

Enter the MCT folder directory:

```
1 cd /home/name.surname/COAWST/Lib/MCT
```

Open the file *Makefile.conf*:

```
1 nedit Makefile.conf
```

And modify the file as follows:

WARNING Remember to change the *name.surname*!

```
FC              = ftn
FCFLAGS         = -O2
F90FLAGS        =
REAL8           = -r8
ENDIAN          = -Mbyteswapio
INCFLAG         = -I
INCPATH         =
MPILIBS         =
DEFS            = -DSYSLINUX -DCPRPGI
FPP             = cpp
FPPFLAGS        = -P -C -N -traditional
CC              = cc
ALLCFLAGS       = -DFORTRAN_UNDERSCORE_ -DSYSLINUX -DCPRPGI -O
COMPILER_ROOT   =
BABELROOT       =
PYTHON          =
PYTHONOPTS      =
FORT_SIZE       =
CRULE           = .c.o
90RULE          = .F90.o
F90RULECPP      = .F90RULECPP
INSTALL         = /home/name.surname/COAWST/Lib/MCT/install-sh -c
MKINSTALLDIRS   = /home/name.surname/COAWST/Lib/MCT/mkinstalldirs
abs_top_builddir= /home/name.surname/COAWST/Lib/MCT/
MCTPATH         = /home/name.surname/COAWST/Lib/MCT/mct
MPEUPATH        = /home/name.surname/COAWST/Lib/MCT/mpeu
EXAMPLEPATH     = /home/name.surname/COAWST/Lib/MCT/examples
MPISERPATH      =
libdir          = /home/name.surname/COAWST/Lib/MCT/pgi/lib
includedir      = /home/name.surname/COAWST/Lib/MCT/pgi/include
AR              = ar cq
RM              = rm -f
```

Install the MCT by typing the following commands:

```
make
make install
```

Observe the messages that appear in the terminal and look for errors. If not, the MCT was successfully compiled.

2.9 Compiling the Sandy test case

There are some test cases within COAWST to be compiled and worked on. In this case we'll compile Hurricane Sandy's project, which couples and nests WRF, ROMS and SWAN. First, it is necessary to know

the structure of COAWST files and folders.

The typical COAWST directory structure is exemplified in Figure 2.3. Mainly, we will use the folders *Projects* e *Work* to work on.

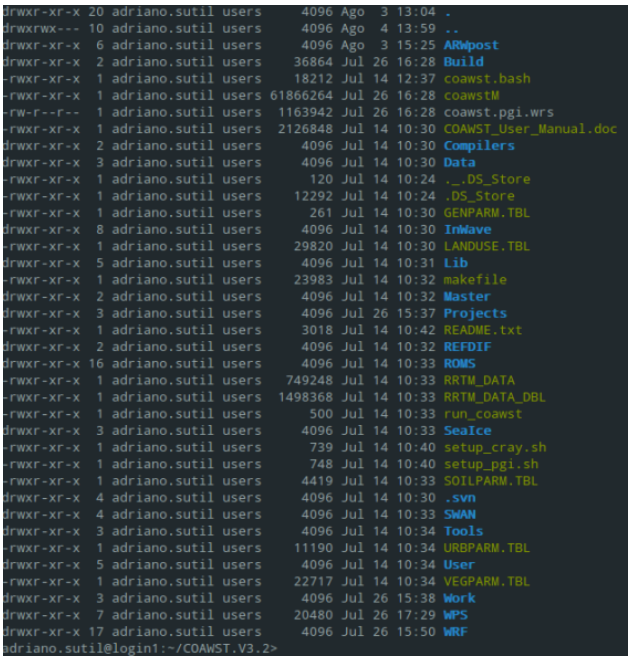


Figure 2.3. Representation of the main COAWST folder and subfolders.

2.9.1 Projects folder

To organize the projects, in the directory *COAWST/Projects/Sandy* are all the files used to simulate the Sandy case. The following files must be inside:

- Bound_spec_command
- coastline.mat
- coupling_sandy.in
- create_sandy_application.m
- hycom_info.mat
- ijcoast.mat
- multi_1.at_10m.dp.201210.grb2
- multi_1.at_10m.hs.201210.grb2
- multi_1.at_10m.tp.201210.grb2
- namelist.input
- ocean_sandy.in
- roms_master_climatology_sandy.m
- roms_narr_Oct2012.nc
- roms_narr_ref3_Oct2012.nc
- Rweigths.txt

- Sandy_bdy.nc
- Sandy_clm.nc
- Sandy_clm_ref3.nc
- sandy.h
- Sandy_ini.nc
- Sandy_ini_ref3.nc
- Sandy_init.hot
- Sandy_ref3_init.hot
- Sandy_roms_contact.nc
- Sandy_roms_grid.nc
- Sandy_roms_grid_ref3.nc
- Sandy_swan_bathy.bot
- Sandy_swan_bathy_ref3.bot
- Sandy_swan_coord.grd
- Sandy_swan_coord_ref3.grd
- scrip_sandy_moving.nc
- scrip_sandy_static.nc
- specpts.mat
- swan_narr_Oct2012.dat
- swan_narr_ref3_Oct2012.dat
- swan_sandy.in
- swan_sandy_ref3.in
- tide_forc_Sandy.nc
- TPAR10.txt
- TPAR11.txt
- TPAR12.txt
- TPAR13.txt
- TPAR14.txt
- TPAR15.txt
- TPAR16.txt
- TPAR17.txt
- TPAR18.txt
- TPAR1.txt
- TPAR2.txt
- TPAR3.txt
- TPAR4.txt
- TPAR5.txt
- TPAR6.txt
- TPAR7.txt
- TPAR8.txt
- TPAR9.txt
- USeast_bathy.mat
- Uweigths.txt
- Vweigths.txt
- wrfbdy_d01
- wrfinput_d01
- wrfinput_d02
- wrflowinp_d01

- wrflowinp_d02

2.9.2 Work folder

To make the management of simulations easier, it is suggested, for each new user, the creation of the *Work* folder in main COAWST directory, with each project inserted separately within it. It is in this folder that will be simulated the test case.

```
cd /scratch/name.surname/COAWST
mkdir Work
cd Work
mkdir Sandy
```

The folder */scratch/name.surname/COAWST/Work/Sandy* must contain the following files.

- run_sandy.sh
- limpa.sh
- link.sh

The file *run_sandy.sh* will be used to start the simulation, *link.sh* creates symbolic links in the folder *Work*, that will be used by the models, and *limpa.sh* is used to clear the folder if an error occurs and a new integration needs to be started. The files are inside the folder */repositorio/work_coawst*.

Therefore:

```
cd /scratch/name.surname/repositorio/work_coawst
cp limpa.sh link.sh run_sandy.sh /scratch/name.surname/COAWST/Work/Sandy
```

Go to the directory */scratch/name.surname/COAWST/Work/Sandy* and open the file*run_sandy.sh*:

```
cd /scratch/name.surname/COAWST/Work/Sandy
nedit run_sandy.sh
```

Search for the following commands and modify the directories according to your username:

```
PBS -o /scratch/name.surname/COAWST/Work/Sandy/rws_total.out
ROOTDIR=/scratch/name.surname/COAWST
```

2.9.3 Compiling the test case

Go to the Sandy project folde:

```
/home/name.surname/COAWST/Projects/Sandy
```

Open the following files to make changes to the next steps:

- coupling_sandy.in
- swan_sandy.in
- swan_sandy_ref3.in
- ocean_sandy.in

```
nedit coupling_sandy.in swan_sandy.in swan_sandy_ref3.in ocean_sandy.in
```

Look for the command line below in the file *coupling_sandy.in*:

```
OCN_name = Projects/Sandy/ocean_sandy.in
```

And replace with:

```
OCN_name = /scratch/name.surname/COAWST/Projects/Sandy/ocean_sandy.in
```

In the files *swan_sandy.in* and *swan_sandy_ref3.in* complete all directory paths below:

Modify:

```
READGRID COORDINATES 1 'Projects/Sandy/Sandy_swan_coord.grd' 4 0 0 FREE
READINP BOTTOM 1 'Projects/Sandy/Sandy_swan_bathy.bot' 4 0 FREE
&READINP WIND 1 'Projects/Sandy/swan_namnarr_30Sep10Nov2012.dat' 4 0 FREE
INITIAL HOTSTART SINGLE 'Projects/Sandy/Sandy_init.hot'
```

By:

```
READGRID COORDINATES 1 '/scratch/name.surname/COAWST/Projects/Sandy/Sandy_swan_coord.grd' 4 0 0 FREE
READINP BOTTOM 1 '/scratch/name.surname/COAWST/Projects/Sandy/Sandy_swan_bathy.bot' 4 0 FREE
&READINP WIND 1 '/scratch/name.surname/COAWST/Projects/Sandy/swan_namnarr_30Sep10Nov2012.dat' 4 0 FREE
INITIAL HOTSTART SINGLE '/scratch/name.surname/COAWST/Projects/Sandy/Sandy_init.hot'
```

In the file *ocean_sandy.in* look for the command lines like below:

```
MyAppCPP = SANDY
VARNAME  = ROMS/External/varinfo.dat
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
NGCNAME =  Projects/Sandy/Sandy_roms_contact.nc
BRYNAME == Projects/Sandy/Sandy_bdy.nc
FRCNAME == Projects/Sandy/roms_narr_Oct2012.nc \
           Projects/Sandy/roms_narr_ref3_Oct2012.nc
```

And replace with:

```
MyAppCPP = Sandy
VARNAME  = /scratch/name.surname/COAWST/ROMS/External/varinfo.dat
GRDNAME == /scratch/name.surname/COAWST/Projects/Sandy/Sandy_roms_grid.nc \
           /scratch/name.surname/COAWST/Projects/Sandy/Sandy_roms_grid_ref3.nc
ININAME == /scratch/name.surname/COAWST/Projects/Sandy/Sandy_ini.nc \
           /scratch/name.surname/COAWST/Projects/Sandy/Sandy_ini_ref3.nc
NGCNAME =  /scratch/name.surname/COAWST/Projects/Sandy/Sandy_roms_contact.nc
BRYNAME == /scratch/name.surname/COAWST/Projects/Sandy/Sandy_bdy.nc
FRCNAME == /scratch/name.surname/COAWST/Projects/Sandy/roms_narr_Oct2012.nc \
           /scratch/name.surname/COAWST/Projects/Sandy/roms_narr_ref3_Oct2012.nc
```

Go back to the main COAWST folder and open the file *coawst.bash*:

```
cd /scratch/name.surname/COAWST
nedit coawst.bash
```

Search for the following commands and modify, if necessary:

```
export   COAWST_APPLICATION=JOE_TC
export   MY_ROOT_DIR=${HOME}/COAWST
export   MY_HEADER_DIR=${MY_PROJECT_DIR}/Projects/JOE_TC
```

By:

```
export   COAWST_APPLICATION=Sandy
export   MY_ROOT_DIR=${HOME}/COAWST
export   MY_HEADER_DIR=${MY_PROJECT_DIR}/Projects/Sandy
```

Activate the modules again into the file *setup_pgi.sh*, and then compile the project with the command:

```
./coawst.bash -j 4  1> coawst.pgi.sandy 2>&1 &
```

This command will create the text file *coawst.pgi.sandy* where you can follow the progress of the compilation. Open the text file with the following command and look for the final message, as shown in Figure 2.4.

```
nedit coawst.pgi.sandy
```

```
IPA: no IPA optimizations for 5 source files
IPA: Recompiling ./Build/get_sparse_matrix.o: new IPA information
IPA: Recompiling ./Build/master.o: new IPA information
IPA: Recompiling ./Build/mct_coupler_utils.o: new IPA information
IPA: Recompiling ./Build/mod_coupler_iounits.o: new IPA information
IPA: Recompiling ./Build/ocean_control.o: new IPA information
IPA: Recompiling ./Build/ocean_coupler.o: new IPA information
IPA: Recompiling ./Build/read_coawst_par.o: new IPA information
IPA: Recompiling ./Build/read_model_inputs.o: new IPA information
IPA: Recompiling ./Build/roms_export.o: new IPA information
IPA: Recompiling ./Build/roms_import.o: new IPA information
```

Figure 2.4. Final message after compiling COAWST.

If you want to follow the progress of the compilation through the terminal, use the command:

```
tail -f coawst.pgi.sandy
```

WARNING The compilation process is long!

We finished! In the main COAWST directory, a file called *coawstM* will be created. In this file all the information about your project will be compiled. Now with COAWST compiled, we will start the test case.

2.10 Simulating the Sandy Test Case

To simulate the case, search for the *run_sandy.sh* file in *Work/Sandy*. Type it:

```
cd /scratch/name.surname/COAWST/Work/Sandy
nedit run_sandy.sh
```

WARNING The *Work* folder should contain the files *clean.sh*, *link.sh* and *run_sandy.sh*. They can be found in the folder used as a repository. See Section 2.9.2.

When opening the file, check if the directories are in accordance with your username and type the command below to start the integration.

```
qsub run_sandy.sh
```

WARNING The command *qsub* will submit you job. It will send the executed script to a batch of the cluster, reserving a part of the processors for your simulation.

The process will generate two files to follow the evolution of the simulation: *log.out* and *log.err*. To open, use:

```
nedit log.out log.err
```

Or look directly at the terminal with the command:

```
tail -f log.out
```

The outputs of the simulations will be stored in the *Work/Sandy* folder. If an error occurs, clean the desktop with the command:

```
./limpa.sh
```

WARNING The simulation of the may take several hours when integrating using 3 processors.

3. COAWST features

So far we have learned how to use the Kerana cluster and how to compile and run a simple COAWST test case. From now on we will enter into the specifics of the models, such as changing the number of processors and the rate of exchange of information between them.

3.1 Structural model files

As you may have noticed when simulating the Sandy test case, the models have files that assist the user about how to setup the project

ROMS uses the *sandy.h* file as a file containing the C pre-processing options that define the project. Access the website *https://www.myroms.org/wiki/cppdefs.h* to know more about the available options into this file. ROMS also uses the *ocean_sandy.in* as standard input for running the model. This file defines the spatial dimensions of the project and parameters that are not informed during compilation, such as time steps, coefficients and physical constants, configuration of vertical coordinates, flags for control the output frequency, among other factors. You can learn more about this file by accessing the site *https://www.myroms.org/wiki/ocean.in*.

WRF uses the file *namelist.input* to manage information about the project, as well as physical parameterization schemes that will be used. To learn about the file description, visit *https://esrl.noaa.gov/gsd/wrfportal/namelist_input_options.html*. To learn about the physical options of the model and the references of each one, visit *http://www2.mmm.ucar.edu/wrf/users/phys_references.html*.

SWAN uses the file *swan_sandy.in* as a manager. It describes several parameters, such as the project description, the input data, the grid and the initial and boundary conditions, the wave physics parameterizations, among others. To learn more about configuring the file, visit the *User Manual* tab in the website *http://swanmodel.sourceforge.net/* and look for the section *Description of commands*.

3.2 Modifying the number of processors

3.2.1 ROMS

The processors are located in the *ocean.in* file, which is located in the *Projects* folder, and are called *NtileI* and *NtileJ*. An example is in Figure 3.1. In this case, ROMS will reserve 320 processors to run, since 16 x 20 = 160.

```
! Domain decomposition parameters for serial, distributed-memory or
! shared-memory configurations used to determine tile horizontal range
! indices (Istr,Iend) and (Jstr,Jend), [1:Ngrids].

      NtileI == 16                              ! I-direction partition
      NtileJ == 20                              ! J-direction partition
```

Figure 3.1. Representation of the number of processors used in ROMS.

3.2.2 WRF

To change the number of WRF processors, it is necessary to modify the file *namelist.input*. You will find the variables *nproc_x* and *nproc_y*, as in Figure 3.2. In this case, 320 processors will be reserved for the atmospheric model.

```
nproc_x                                     = 16,
nproc_y                                     = 20,
```

Figure 3.2. Representation of the number of processors for the WRF.

3.2.3 SWAN

The SWAN *.in* file does not discriminate the numbers of processors used, so just change the number number of processors in *coupling.in*, as will be shown in the subsection below.

3.2.4 COAWST

Now that the number of processors have been modified, the coupler needs to be informed about the change. This information is communicated through the file *coupling.in*. It should contain the total number of processors to be used by the ROMS, WRF and SWAN models, as in the figure 3.3:

```
!
! Number of parallel nodes assigned to each model in the coupled system.
! Their sum must be equal to the total number of processors.

   NnodesATM =  320                  ! atmospheric model
   NnodesWAV =  64                   ! wave model
   NnodesOCN =  320                  ! ocean model
```

Figure 3.3. Representation of the number of processors for each COAWST module.

The *NnodesATM* refers to the total number of processors used by WRF, *NnodesWAV* is the total used by SWAN and *NnodesOCN* is total used by ROMS. Just change according to the total number of processors used in the previous steps.

Finally, it is necessary to modify the total number of processors in *run.sh*,used to submit the experiment. Add the total number of processors used by the three models, following the equation 3.1:

$$TotalProc = NnodesATM + NnodesWAV + NnodesOCN \tag{3.1}$$

Where*TotalProc* is the sum of all processors used in the models.

Now open the file *run.sh*, located inside the folder *Work*, and look for the following line:

```
PBS -l mppwidth= 3
```

And for the line:

```
aprun -n 36 coawstM ./coupling.in 1> log.out 2> log.err
```

Modify the number *3* by the total number of processors used.

3.3 Modifying the coupling time interval between models

To modify the information exchanged between models, open the file *coupling.in* and modify the variables *TI_ATM2WAV*, *TI_ATM2OCN*, *TI_WAV2ATM*, *TI_WAV2OCN*, *TI_OCN2WAV*, *TI_OCN2ATM*, as in Figure 3.4.

WARNING The information exchange rate is defined in seconds.

```
! Time interval (seconds) between coupling of models.

  TI_ATM2WAV =   900.0d0              ! atmosphere to wave coupling interval
  TI_ATM2OCN =   900.0d0              ! atmosphere to ocean coupling interval
  TI_WAV2ATM =   900.0d0              ! wave to atmosphere coupling interval
  TI_WAV2OCN =   900.0d0              ! wave to ocean coupling interval
  TI_OCN2WAV =   900.0d0              ! ocean to wave coupling interval
  TI_OCN2ATM =   900.0d0              ! ocean to atmosphere coupling interval
```

Figure 3.4. Information exchange interval between models used in COAWST. In this example, the exchange will occur every 900 seconds.

4. Building the WRF

Now that we have learned how to simulate a test case and change the coupling rate and number of processors, we will begin the process of creating a specific grid and the initial and boundary conditions for the WRF using the NCEP Climate Forecast System Reanalysis (CFSR; **Saha2006**).

4.1 Compiling WRF in Kerana cluster

It is recommended to copy the WRF folder inside COAWST to your work root area, in order to avoid conflicts if you choose to use the WRF without coupling to the COAWST model. Therefore:

```
cd /scratch/name.surname/COAWST
cp -r WRF /scratch/nome.dsobrenome
```

Activate the modules in the *setup_pgi.sh* file with the command:

```
source setup_pgi.sh
```

Enter the copy folder of the WRF (*/home/name.surname/WRF*) and run:

```
./configure
```

WARNING The next step can be automated. If you choose the automated compilation, see Section 2.7.

If you have automated the compilation process, as shown in Section 2.7, the files *real.exe*, *wrf.exe*, *tc.exe* and *ndown.exe* will be created. If you choose the manual option on the Kerana cluster, look down for the option *Cray XC CLE / Linux x86 _64, Cray compiler with gcc (dmpar)*.

In the next option, the following message will appear:

```
Compile for nesting? (1=basic, 2=preset moves, 3=vortex following) [default 1]:
```

Choose option 1.

WARNING End of the automated build process.

Type:

```
compile em_real
```

The *real.exe*, *wrf.exe*, *tc.exe* and *ndown.exe* will be generated in the */home/name.surname/WRF/test/em_real*.

4.2 WRF Preprocessing System (WPS)

To build the initial and boundary conditions of the WRF, we will use WRF Preprocessing System (WPS). The WPS is a set of three programs that prepare the input data for the simulations. It consists of the following modules:

- ***geogrid***: Defines the model domain and interpolates the geographic data for the grid;
- ***ungrib***: Extract weather fields from *.grib* files;
- ***metgrid***: Interpolates the weather fields extracted by ungrib to the model grid defined by metrid.

WPS is controlled by the *namelist.wps* file and is outlined in Figure 4.1.

For more information about WPS, visit: *http://www2.mmm.ucar.edu/wrf/OnLineTutorial/*.

After creating these files, the *real.exe* is used to interpolate the meteorological fields to ηlevel.

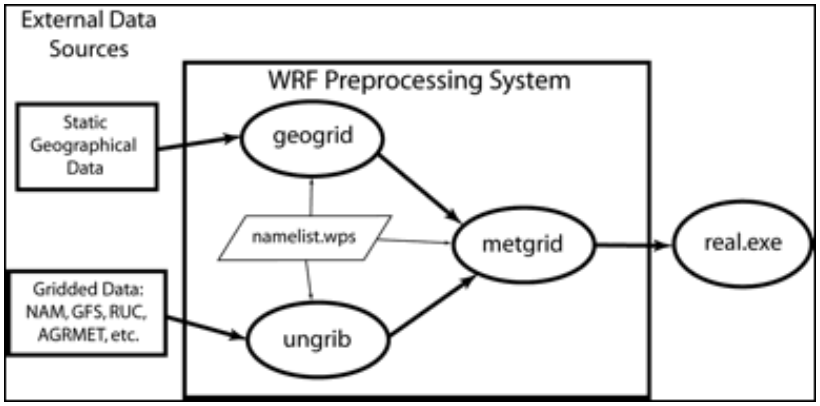


Figure 4.1. WPS schemeatic.
Author: **duda2006**.

4.2.1 Downloading WPS

WPS is included in the COAWST package, however WPS can be downloaded by visiting *http://www2.mmm.ucar.edu/wrf/users/download/get_source.html*. Download it and add the *tar.gz* file inside the cluster with *sftp*:

```
sftp -P2000 name.surname@acesso-hpc.cptec.inpe.br
cd COAWST
put WPSV3.9.0.1.tar.gz
```

To unzip, use *ssh* to enter the cluster and type:

```
ssh -Y name.surname@acesso-hpc.cptec.inpe.br -p 2000
cd COAWST
tar -xvzf WPSV3.9.0.1.tar.gz
```

4.2.2 Compiling WPS in the Kerana cluster

WARNING As stated earlier in Section 4.2.2, WPS is included with COAWST. We recommend to use this version.

To compile the WPS and generate the executables, it is necessary to activate the libraries with *setup_pgi.sh*, found in the directory */home/name.surname/repository*. Activate them with the command:

```
source setup_pgi.sh
```

Inside the WPS folder, enter the following command:

```
./configure
```

You will be asked which machine and compiler will be used. Choose the option *Cray XE / XC CLE / Linux x86 _64, PGI compiler (serial)*. In this case, a message will be generated that Fortran is not compatible with C and NetCDF, but ignore it.

Open *configure.wps* file:

```
nedit configure.wps
```

Modify:

```
WRF_DIR  = ../WRFV3
SFC      = ftn
SCC      = gcc
DM_CC    = cc
DM_FC    = ftn
FFLAGS   = -N255 -f free -h byteswapio
F77FLAGS = -N255 -f fixed -h byteswapio
```

By:

```
WRF_DIR  = /scratch/name.surname/WRF
SFC      = ftn
SCC      = gcc
DM_CC    = gcc
DM_FC    = ftn
FFLAGS   =
F77FLAGS =
```

Save the changes and start the compilation with the command:

```
./compile
```

You should generate the executables *metgrid.exe* and *geogrid.exe*. However, it is possible that *ungrib.exe* is not generated. In this case, repeat:

```
./configure
```

Choose the option *Linux x86_64, PGI compiler (dmpar)*.

Open *configure.wps* file again, modify:

```
WRF_DIR = ../WRFV3
SFC     = pgf90
SCC     = pgcc
DMFC    = mpif90
DMCC    = mpicc
```

By:

```
WRF_DIR = /home/name.surname/WRF
SFC     = ftn
SCC     = gcc
DMFC    = ftn
DMCC    = gcc
```

The installer will generate only the *ungrib.exe*.

4.3 Building your project using CFSR

A database as option to use for simulations is the NCEP Climate Forecast System Reanalysis (CFSR) (*http://rda.ucar.edu/datasets/ds093.0*). This database is a reanalysis, from January 1979 to December 2010, where a coupled model (atmosphere-ocean-land surface and sea ice) assimilates orbital data, research cruises, oceanographic buoys and weather stations. This database has vertical resolution with 38 levels, extending from the surface to 1 hPa, with a 6-hour temporal resolution and horizontal of 0.5°for pressure levels data and 0.312°for variables on the surface.

To access the data it is necessary to register on the site. After doing this, enter the database link, click on the*Data Acess* and then, *Get a subset*.

On the next page, as seen in Figure 4.2, write the temporal selection, and select in *Parameter presets* one of the three parameters of the WRF presets: (*Pressure*, *Surface* and *SST*).

WARNING	Only one parameter (*Pressure*, *Surface* and *SST*) can be downloaded at a time, therefore you need to repeat this step three times.

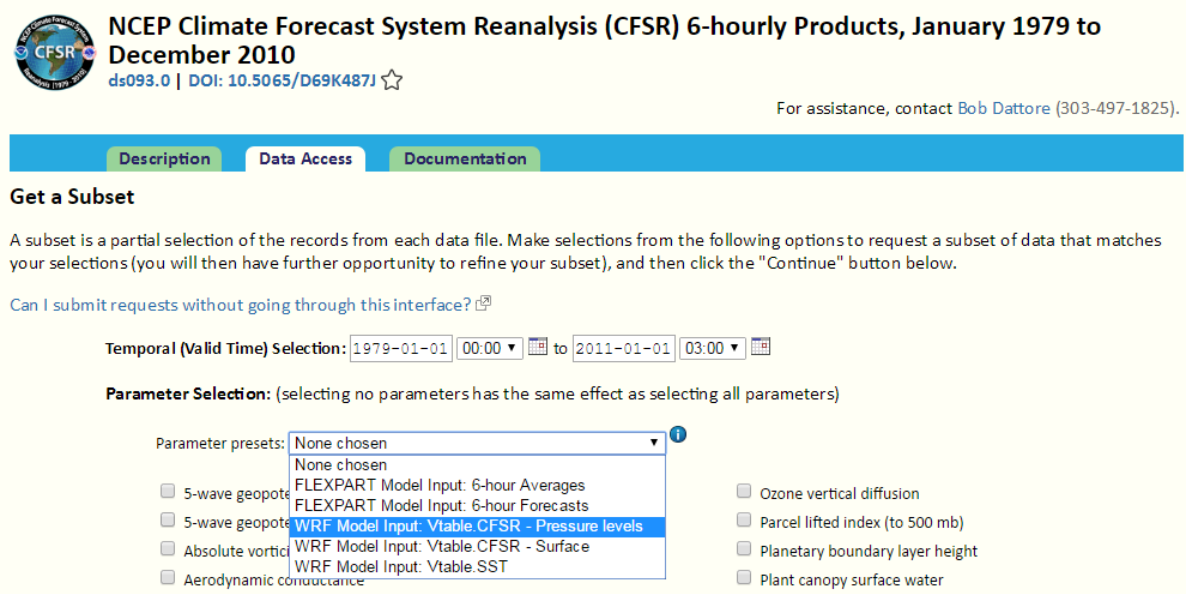


Figure 4.2. Screenshot of the CFSR data page.

Download the data (as *GRIB*) and separate it into three folders according to the parameters of each one: SST, Surface and Pressure. Compress them and put them in the cluster:

```
tar -cvzf SST.tar.gz SST/
tar -cvzf Pressure.tar.gz Pressure/
tar -cvzf Surface.tar.gz Surface/
ssh -Y name.surname@acesso-hpc.cptec.inpe.br -p 2000
mkdir Data_CFSR
exit
sftp -P2000 name.surname@acesso-hpc.cptec.inpe.br
cd Data_CFSR
put SST.tar.gz
put Pressure.tar.gz
put Surface.tar.gz
exit
ssh -Y name.surname@acesso-hpc.cptec.inpe.br -p 2000
cd Data_CFSR
tar -xvzf SST.tar.gz
tar -xvzf Pressure.tar.gz
tar -xvzf Surface.tar.gz
```

4.3.1 *geogrid*

As previously mentioned, textit geogrid generates the grid domain using geographic data. The data can be obtained from this website: *http://www2.mmm.ucar.edu/wrf/users/download/get_sources_wps_geog.html*, or directly from the guide repository:

```
/scratch/name.surname/repositorio/COAWST/WPS/geog
```

With the CFSR and *geogrid* data in hand, enter into the WPS directory and open *namelist.wps*. The file structure is shown in Figure 4.3:

```
nedit namelist.wps
```

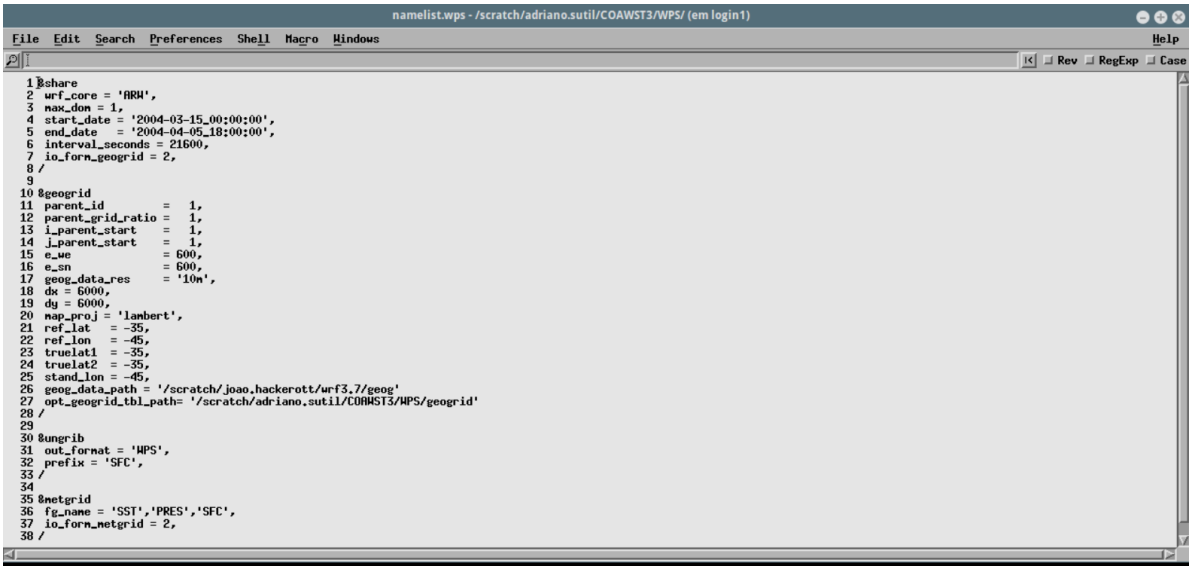


Figure 4.3. *namelist.wps* example.

Change the *geog_data_path* path to the directory where geographic data is located (*geog folder*), *geog_data_res* if you want to use other geographic data and *opt_geogrid_tbl_path*, which is the *geogrid* directory.

The WPS uses a ground zero point defined by the user as the latitude and longitude degrees in *ref_lat* and *ref_lon*. It will generate the grid from that point and through the chosen spatial resolution. Take as example the Figure 4.3: a simulation will be prepared with 6 km of spatial resolution (*dx* and *dy* = 6000), in Lambert projection, with 600 grid points. Using as reference latitude -35°longitude -45°, a grid will be created with 600 points in the west-east direction (*e_we*) and 600 points in the south-north direction (*e_sn*), with each point having a spacing of 6000 meters (6 km) between them.

To start *geogrid*, you need a file with a Shell extension (*.sh*), which submits program to be executed in the cluster. It can be found in the */repository/WPS_scripts* directory. Move the contents of the folder to the WPS directory with the commands:

```
cd /home/name.surname/repositorio/WPS_scripts
mv qsub_geogrid.sh qsub_ungrib.sh qsub_metgrid.sh /home/name.surname/COAWST/WPS
```

The structure of the *qsub_geogrid.sh* file is shown in Figure 4.4. Modify the directory according to your user.

```
#!/bin/sh
#PBS -l mppwidth=1
#PBS -N GEOGRID
#PBS -j oe
#PBS -o Joe_test.out
#PBS -l walltime=02:30:00
#PBS -q workq
  echo "Running GEOGRID on KERANA"
#
  export MPICH_ENV_DISPLAY=1
  export MPICH_ABORT_ON_ERROR=1
  export MPICH_RANK_REORDER_DISPLAY=1
  export MPICH_RANK_REORDER_METHOD=1
  export MALLOC_MMAP_MAX_=0
  export MALLOC_TRIM_THRESHOLD_=536870912
  export OMP_NUM_THREADS=1
#
  EXECDIR=/scratch/nome.sobrenome/COAWST/WPS

  cd $EXECDIR

  aprun -n 1 geogrid/src/geogrid.exe  1> log.out 2> log.err

  \rm geogrid.log.00*
```

Figure 4.4. *qsub_geodrid.sh* example.

To run *geogrid*, use *qsub* command:

```
1 qsub qsub_geodrid.sh
```

Two files will be generated: *log.err* e *log.out*. you can follow how the program is being executed and search for errors. The program completion message is showned in Figure 4.5, in the file *log.out*.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Successful completion of geogrid.        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Application 593294 resources: utime ~813s, stime ~1s
```

Figure 4.5. Example of completion of *geogrid*.

When completing the execution of *geogrid*, the file *geo_em_d01.nc* will be generated. You can view the NetCDF file with Ncview:

```
1 module load netcdf
2 ncview geo_em_d01.nc
```

You will always need to run *geogrid* until you find a domain that is suitable for your project, since WPS does not have a graphical interface. One solution to work around the problem is to use NCAR Command Language version 6.4 (NCL; **Ncl2017**) to plot the image of the chosen domain. This will be the subject of the next subsection.

4.3.2 Using the NCL to plot the grid domain

To install the NCL, move the *NCL* folder inside the repository:

```
mv /scratch/name.surname/repositorio/NCL /scratch/name.surname
```

Open the *.bashrc* file:

```
cd
nedit .bashrc
```

Add the following command lines, changing to your username:

```
export NCARG_ROOT=/scratch/name.surname/NCL
export PATH=$NCARG_ROOT/bin:$PATH
```

Save the file, reload the modules inside *.bashrc* and then execute NCL.

```
source .bashrc
ncl
```

Check if the NCL has been initialized as in Figure 4.6.

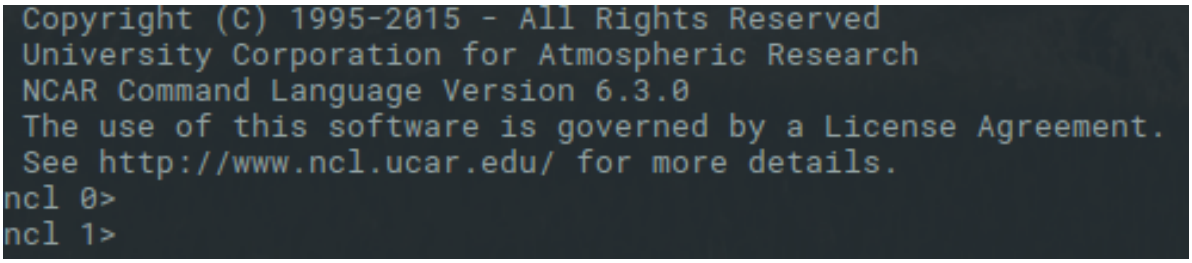


Figure 4.6. Example of NCL initialization.

Enter the repository and open the *plotgrids.ncl* file. This script generates the domain image from the data in the *namelist.wps* file.

```
cd /scratch/name.surname/repositorio
nedit plotgrids.ncl
```

Modify the *namelist.wps* directory according to your user area:

```
filename = "/home/name.surname/COAWST/WPS/namelist.wps"
```

Run the code with the command:

```
ncl plotgrids.ncl
```

The code will search for the grid points and the domain resolution through the file *namelist.wps* and show it on the screen, as shown in Figure 4.7. Repeat the process until you find a domain that suits your needs.

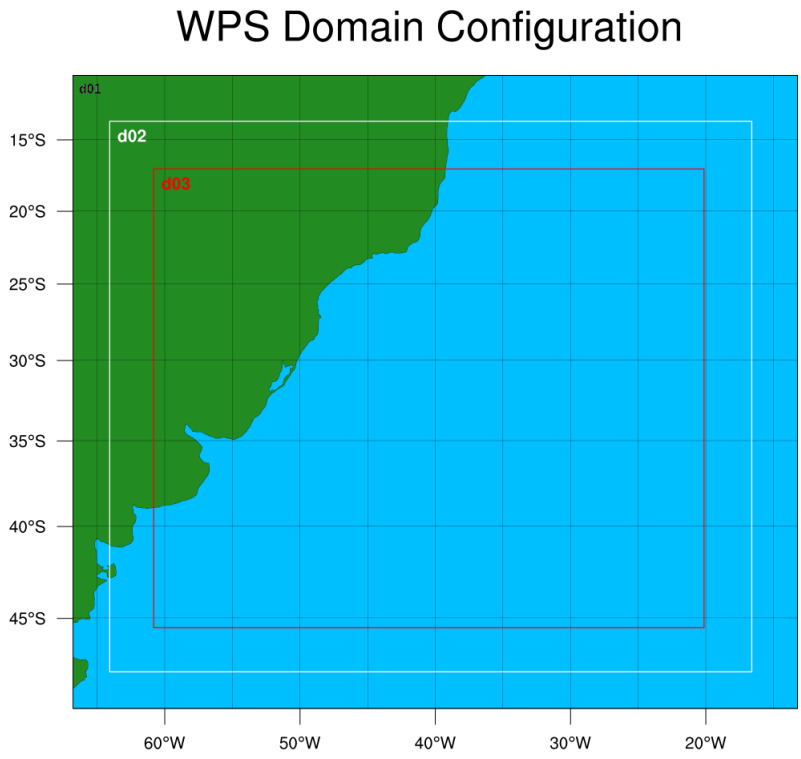


Figure 4.7. Figure generated by the *plotgrids.ncl* file. In this example, three domains were generated.

4.3.3 *ungrib*

With the domain prepared by *geogrid*, we can proceed to *ungrib*, which extracts the data from the *Grib* files. We will start creating the SST data. Open *namelist.wps* and change the *prefix* to *SST*, in the *& ungrib* section, as shown in Figure 4.8.

```
&ungrib
 out_format = 'WPS',
 prefix = 'SST',
/
```

Figure 4.8. *&ungrib* example.

Import the Variable Table for SST through a symbolic link. Type in the terminal:

```
ln -sf ungrib/Variable_Tables/Vtable.SST Vtable
```

WARNING The *Vtable* is used to read the files in *Grib* format. It is mandatory to use the corresponding *Vtable* to the chosen database. You can consult them at *WPS/ungrib/Variable_Tables*.

Create the symbolic links with the SST data through the *link_grib.csh* file:

```
./link_grib.csh /scratch/name.surname/Dados_CFSR/SST/*
```

Open the *qsub_ungrib.sh* file and modify the path according to your user and type:

```
qsub qsub_ungrib.sh
```

WARNING The *qsub_ungrib.sh* file is located in the */repository/WPS_scripts* folder.

Several files will be created with the initial *SST:* followed by the date chosen for simulation. To check if everything went well, look for the message in Figure 4.9 at the end of the file *log.out*.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Successful completion of ungrib.    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Application 593300 resources: utime ~18s, stime ~1s
```

Figure 4.9. Final message in the *log.out* file when running *ungrib*.

We will proceed to the CFSR surface data. Change the *prefix* in *namelist.wps* (Figure 4.8). Modify *SST* by *SFC* and save the modification.

Import the CFSR *Variable Table* with the symbolic link creation command:

```
ln -sf ungrib/Variable_Tables/Vtable.CFSR Vtable
```

Create the symbolic links to the CFSR surface data with the command below and run *ungrib* again:

```
./link_grib.csh /scratch/name.surname/Dados_CFSR/Surface/*
qsub qsub_ungrib.sh
```

Look for the final message as shown in Figure 4.9.

Change the *prefix* of *namelist.wps* (Figure 4.8), replace *SFC* with *PRES* and save the file.

Create the symbolic links to the pressure data and run *ungrib*:

```
./link_grib.csh /scratch/name.surname/Dados_CFSR/Pressure/*
qsub qsub_ungrib.sh
```

Finally, look for the final message as identified in Figure 4.9. At the end of these steps, there will be several files with the initials *SST*, *PRES*, *SFC* followed by the dates of the chosen period.

4.3.4 *metgrid*

To interpolate the data generated by *ungrib.exe*, open *namelist.wps* and change on the *fg_name* tab the names used in *ungrib*. In the previous case, *PRES*, *SFC* and *SST* were used, as in the example in Figure 4.10:

```
&metgrid
 fg_name = 'PRES','SFC','SST',
 io_form_metgrid = 2,
/
```

Figure 4.10. *&metgrid* example.

Open and then change the *qsub_metgrid.sh* file path and run:

```
qsub qsub_metgrid.sh
```

Move the files generated by *metgrid* (*met_em.d01**) to the real WRF case directory:

```
mv met_em.d01* /home/name.surname/WRF/test/em_real
```

> **WARNING** If you are an experienced user, you can create symbolic links pointing to the *met* files instead of moving them.

4.3.5 *real*

We will use the *real.exe* program to generate the files that will be used in WRF. Open the file *namelist.input*, which is in the WRF folder (*/home/name.surname/WRF/test/em_real*), and modify it according to your chosed domain and simulation time in *namelist.wps*.

For the WRF to simulate the SST, it is necessary to include the following variables at the end of the section *&time_control* (Figure 4.11):

```
io_form_auxinput4  = 2,
auxinput4_inname   = "wrflowinp_d<domain>",
auxinput4_interval = 360,
```

The *io_form_auxinput4* refers to the final format of the textitwrflowinp file that will be generated by the *real*. The *auxinput4_inname* is the name of the boundary condition file for SST and *auxinput4_interval* is the time interval, in minutes, of the boundary file.

```
&time_control
run_days                            = 5,
run_hours                           = 0,
run_minutes                         = 0,
run_seconds                         = 0,
start_year                          = 2008,
start_month                         = 11,
start_day                           = 21,
start_hour                          = 00,
start_minute                        = 00,
start_second                        = 00,
end_year                            = 2008,
end_month                           = 11,
end_day                             = 25,
end_hour                            = 18,
end_minute                          = 00,
end_second                          = 00,
interval_seconds                    = 21600,
input_from_file                     = .true.,
history_interval                    = 360,,
frames_per_outfile                  = 80,
restart                             = .false.,
restart_interval                    = 50000,
io_form_history                     = 2
io_form_restart                     = 2
io_form_input                       = 2
io_form_boundary                    = 2
debug_level                         = 0
io_form_auxinput4                   = 2
auxinput4_inname                    = "wrflowinp_d<domain>"
auxinput4_interval                  = 360,
/
```

Figure 4.11. *&time_control* example.

Also, in the *&physics* (Figure 4.12) in the *namelist.input* the option to update the SST:

```
1 sst_update = 1,
```

```
&physics
sst_update                          = 1,
mp_physics                          = 6,
ra_lw_physics                       = 4,
ra_sw_physics                       = 3
swint_opt                           = 1,
sf_sfclay_physics                   = 1,
sf_surface_physics                  = 2,
bl_pbl_physics                      = 1,
cu_physics                          = 1,
isfflx                              = 1,
ifsnow                              = 1,
icloud                              = 1,
surface_input_source                = 1,
num_soil_layers                     = 4,
sf_urban_physics                    = 0,
sst_skin                            = 0,
```

Figure 4.12. *&physics* example.

After completing the modifications, you must use the file *qsub_real.sh* to submit the job. The file is located in

the */repository/WRF_scripts* directory. Move the file to the *em_real* folder, change the directory according to your user and execute:

```
cd /home/name.surname/repositorio/WRF_scripts
mv qsub_real.sh /home/name.surname/WRF/test/em_real
qsub qsub_real.sh
```

WARNING The *qsub_real.sh* file is located in the */repository/WRF_scripts* folder.

To check if *real* succeeded, look at the end of the *log.out* file for the success message as shown in Figure 4.13.

```
d01 2007-02-18_00:00:00 real_em: SUCCESS COMPLETE REAL_EM INIT
d01 2007-02-18_00:00:00 Timing for loop #   41 =          2 s.
d01 2007-02-18_00:00:00 real_em: SUCCESS COMPLETE REAL_EM INIT
d01 2007-02-18_00:00:00 real_em: SUCCESS COMPLETE REAL_EM INIT
d01 2007-02-18_00:00:00 Timing for loop #   41 =          2 s.
d01 2007-02-18_00:00:00 real_em: SUCCESS COMPLETE REAL_EM INIT
d01 2007-02-18_00:00:00 real_em: SUCCESS COMPLETE REAL_EM INIT
d01 2007-02-18_00:00:00 real_em: SUCCESS COMPLETE REAL_EM INIT
d01 2007-02-18_00:00:00 real_em: SUCCESS COMPLETE REAL_EM INIT
Application 593290 resources: utime ~11862s, stime ~117s
```

Figure 4.13. Successful completion of *real*.

Upon completing *real*, three files will be generated: *wrfbdy_d01*, *wrfinput_d01* and *wrflowinp_d01*, if you have chosen only one domain. After that, the WRF conditions are ready to be copied to your project folder in COAWST:

```
cp wrfbdy_d01 wrfinput_d01 wrflowinp_d01 /home/name.surname/COAWST/Projects/project_name
```

4.4 Using the WRF

With the files ready, it is possible to start the simulation of your project using only the WRF.

To run the simulation on the cluster, it is necessary to use the file *qsub_wrf.sh*. It is located in the */repository/WRF_scripts* directory. Move the file to the WRF *in_real* folder, modify the directory according to your user and save:

```
cd /home/name.surname/repositorio/WRF_scripts
mv qsub_wrf.sh /home/name.surname/WRF/test/em_real
nedit qsub_wrf.sh
```

To start the simulation, type:

```
qsub qsub_wrf.sh
```

WARNING The *qsub_wrf.sh* file is located in the */repository/WRF_scripts* folder.

Follow the evolution of the job with the *log.out* and *log.err* files. If you want to have the information updated by the terminal, type:

```
tail -f log.out
```

5. Building the ROMS

5.1 Installing Python libraries on the personal computer

Before generating the ROMS files, it is necessary to install the libraries and modules. In this chapter, we address how to download them, as well as the commands in the terminal to install through *apt-get*.

There are two ways to install the required libraries: manually, and through Anaconda. Anaconda is a free and open source platform for Python and R and has a package manager called Conda, which makes easy to install the libraries. In this guide, I will present both options. For convenience, I suggest installing using Conda (Section 5.1.2), but beware of version conflicts, as the Conda repositories are constantly updated.

WARNING	ROMS files must be builded on your own computer and no longer on Kerana cluster, as previously done with WRF.

5.1.1 Installing manually

WARNING	Consider installing libraries with Anaconda will be faster than installing manually. See Section 5.1.2.

5.1.2 Installing with Conda

Anaconda is a distribution for Python and R that supports the various libraries and packages used in this guide, making installation easier. To download Anaconda go to *https://www.continuum.io/downloads*. To install, just enter the directory where the installation file is located, change the installation permissions of the file and follow the installer's instructions:

```
sudo chmod 770 Anaconda2.sh
 ./Anaconda2.sh
```

Next, enter the following commands on the terminal to install the libraries needed to build a project on ROMS using Python:

```
conda install -c conda-forge openmpi
conda install -c anaconda gcc
conda install -c mutirri szip
conda install -c anaconda zlib
conda install -c anaconda curl
conda install -c anaconda hdf5
conda install -c conda-forge netcdf-fortran
conda install -c scitools jasper
conda install -c anaconda libpng
conda install -c conda-forge udunits
conda install -c anaconda netcdf4
conda install -c anaconda cython
conda install -c anaconda numpy
conda install -c anaconda scipy
conda install -c anaconda mpi4py
conda install -c conda-forge esmf
conda install -c conda-forge lpsolve55
conda install -c conda-forge cftime
```

5.2 Installing Pyroms

With the libraries already installed, this section will show you how to install Pyroms. It is a collection of tools to create ROMS files. It was originally started by Rob Hetland as a project on GoogleCode and rewritten by Frederic Castruccio.

Pyroms can be easily installed using Pip, a command line tool that allows you to install software packages written in Python. To install Pip for Python 2, use the following:

```
sudo apt install python-pip
```

For Python 3 users, use:

```
sudo apt install python3-pip
```

Then, install the Pyrons by cloning the repository at Github:

```
git clone https://github.com/ESMG/pyroms.git
```

Install it using Pip:

```
pip install -e pyroms/pyroms
pip install -e pyroms/pyroms_toolbox
pip install -e pyroms/bathy_smoother
```

This will install both three Pyroms packages, but if you try to import any of then without Script installed, you will get a warning similar to that found in Figure 5.1:

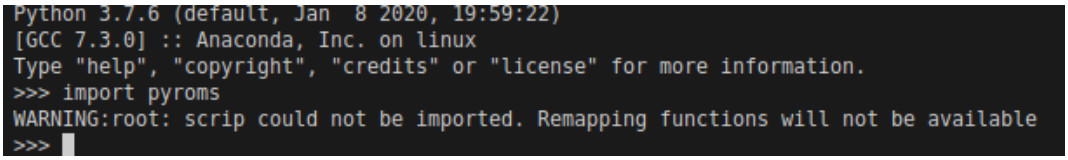


Figure 5.1. Scrip warning when Pyroms is imported without installing it.

As stated in the Pyroms repository, the scrip module is not available via Conda or any other package repository, but it can be built and installed from source as the following:

```
cd pyroms/pyroms/external/scrip/source/
```

Then, print the active Conda environment path. This will be used to find the netCDF library.

```
conda info | grep "active env location"
```

The output must be something like as the following:

```
active env location : /home/USER/anaconda3
```

Export the PREFIX variable with the location found in the previous step.

```
export PREFIX=/home/USER/anaconda3
```

Install the module with the make command.

```
sudo make DEVELOP=1 PREFIX=$PREFIX install
```

Move the Scrip files to the Pyroms folder, like the following:

```
mv -vf scrip*.so ../../../pyroms
```

5.2.1 Pyroms Palau_Hycom test case

> **WARNING** This section will present the script files used by Pyroms as example to create ROMS forcing files. If you want to used our model2roms toolbox, check Section 5.3.

This subsection was written by Dr. Jonas Takeo Carvalho.
Lattes CV: ***http://lattes.cnpq.br/8827254187143196***

You can check several examples inside the *examples* folder in Pyroms root directory. In this section we will show how to run the Palau_Hycom test case. Before running the scripts, check the paths contained within its structure. In some scripts it is necessary to change the indicated path to the path in which your files are. First, run the file *get_hycom_GLBa0.08_Palau_grid.py* to create the grid named *HYCOM_GLBa0.08_Palau_grid.nc*. This is a grid that represents Hycom data.

```
python get_hycom_GLBa0.08_Palau_grid.py
```

Next, download the temperature, salinity, *u* and *v* components of the currents and SSH variables through the scripts for this data. Each variable has a script. In the example for this tutorial, in addition to the paths within the scripts, the number of days to download was also modified. On lines 79 and 82 of each variable file (for example, **get_hycom_GLBa0.08_Palau_temp_2015.py** thee values *366* and 365 were changed to 1 (it means that it will be downloaded only one day). Before running, create a directory called *data*. The downloaded files will be placed in this directory.

Manipulating the files

After executing the grid generation and data download script files, you will have the following files:

- HYCOM_GLBa0.08_PALAU_grid.nc
- HYCOM_GLBa0.08_temp_2015_001.nc
- HYCOM_GLBa0.08_salt_2015_001.nc
- HYCOM_GLBa0.08_ssh_2015_001.nc
- HYCOM_GLBa0.08_u_2015_001.nc
- HYCOM_GLBa0.08_v_2015_001.nc

We must now create the initial and boundary conditions with the downloaded data, but first we must concatenate them into a single file. As they are files with different variables, without a common variable, we must first generate a variable to be shared between them, and still, be useful for reading the ROMS model. Then the variable *ocean_time* will br created, which is read by the model and common to all files. For the most recent data from Hycom, this variable already exists, but but for older Hycom data, this step is important. Use *NCO* to do this step:

- ncks -O −−mk_rec_dmn ocean_time HYCOM_GLBa0.08_ssh_2015_001.nc var1.nc
- ncks -O −−mk_rec_dmn ocean_time HYCOM_GLBa0.08_salt_2015_001.nc var2.nc
- ncks -O −−mk_rec_dmn ocean_time HYCOM_GLBa0.08_temp_2015_001.nc var3.nc
- ncks -O −−mk_rec_dmn ocean_time HYCOM_GLBa0.08_u_2015_001.nc var4.nc
- ncks -O −−mk_rec_dmn ocean_time HYCOM_GLBa0.08_v_2015_001.nc var5.nc

After including the variable *ocean_time*, concatenate the files into a single file with the commands:

- ncks -a -O var1.nc HYCOM_GLBa0.08_2015_001.nc
- ncks -a -A var2.nc HYCOM_GLBa0.08_2015_001.nc
- ncks -a -A var3.nc HYCOM_GLBa0.08_2015_001.nc
- ncks -a -A var4.nc HYCOM_GLBa0.08_2015_001.nc
- ncks -a -A var5.nc HYCOM_GLBa0.08_2015_001.nc

The file *HYCOM_GLBa0.08_2015_001.nc* contains all 5 variables necessary to generate boundary initial conditions. Now, change the paths in the scripts *make_bdry_file.py* and **make_ic_file.py** pointing to the local directories you are using. It is also necessary to do the same thing for interpolation script files. All files with the initial name *remap* must be updated with you path.

After all the paths have been corrected, first run the script *make_remap_weights_file.py*, which will generate the files used during the process of creating the initial and boundary condition files. After these files are generated, execute the scripts **make_bdry_file.py** and **make_ic_file.py** remembering only to add the year you chosed on the command line, for example:

```
1 python make_bdry_file.py 2015
```

Should be generated the files *HYCOM_GLBy0.08_2019_001_ic_PALAU.nc* and *HYCOM_GLBy0.08_2019_001_bdry_PALAU.nc*

Observations

Pyroms uses a file called *gridid.txt*, which contains information regarding the ROMS grid, such as the name of the grid, the number of vertical levels, among other parameters. This file can be found in Pyroms in the *pyroms/examples* directory. We should note that the information in the PALAU grid is already in the file, but it is necessary to change the directory path.

Another information that we must take into account is that this grid generated for PALAU is not a typical ROMS grid, it is only to be an example. The *gridid.txt* file must contain the information from the ROMS notes, which will also be used to make the model, so we must export this file to be viewed in any directory, as an example.

In addition to this command, it is recommended to export in your *.bashrc* file the paths of the Python environment created for Pyroms:

```
export PATH="$HOME/install/anaconda3/bin:\$PATH"
export PYTHONPATH=\$PYTHONPATH:$HOME/install/anaconda3/envs/pyroms37/lib
export PYTHONPATH=\$PYTHONPATH:$HOME/install/anaconda3/envs/pyroms37/lib/python3.7/site-packages
```

5.2.2 Pyroms LOA Antarctica test case

Based on the structure of the Palau test case, the scripts to generate initial and boundary conditions with Pyroms were replicated to the Antarctica region. We selectec the Antarctic Peninsula as its central point in the grid. Below are some steps to generate the desired files.

Replicating the scripts

In addition to replicating the scripts to generate the grid that represents and download Hycom data, we must generate the ROMS grid for the region of interest. We can use as an example the script *make_YELLOW_grd_v1.py* of the Yellow_Sea test case. In this script we must replace the name, longitude and latitude values, grid dimensions, and vertical coordinate values. For this case, the ROMS grid, and its information from the *gridid.txt* file can be found in the Kerana's repository.

In the script files for downloading the data (*get_hycom_GLBa0.08_variavel.py*), it is also necessary to change the path of the Hycom data to be downloaded. In the Palau example, the structure of the download data is from 2015. This structure is changed for periods called *exp*. These scripts are also found in the repository.

Correcting dimensions

A necessary change in the scripts is the grid size that represents Hycom and its downloaded data. After defining the limits for both latitude and longitude, we must choose an additional number of points outside the grid. If we don't define any extra points in the grid or in the data, the following error will be displayed:

```
ValueError: operands could not be broadcast together with shapes (873,1249) (875,1251)
```

The first set refers to the size of the grid that represents the Hycom and the second set represents the downloaded Hycom data. We must adjust these values, increasing the dimension of the latitude, longitude and variable within the Hycom data download scripts by 2 units compared to the script representing the Hycom grid. In the figure 5.2 and in the figure 5.3 we can see the location in the scripts where this addition is made.

```
#read grid and variable attributes from the first file
url='https://tds.hycom.org/thredds/dodsC/GLBy0.08/expt_93.0/ts3z'

#url='http://tds.hycom.org/thredds/dodsC/datasets/GLBa0.08/expt_91.1/2015/temp/archv.2015_001_00_3zt.nc'
#url='https://tds.hycom.org/thredds/catalogs/GLBy0.08/expt_93.0/ts3z/archv.2019_001_00_3zt.nc'
dataset = netCDF4.Dataset(url)
lon = dataset.variables['lon'][lllon-5:urlon+5]
lat = dataset.variables['lat'][lllat:urlat+5]
z = dataset.variables['depth'][:]
#spval = dataset.variables[invarname]._FillValue
var = dataset.variables[invarname][0,:,lllat:urlat+5,lllon-5:urlon+5]
spval = var.get_fill_value()
units = dataset.variables[invarname].units
long_name = dataset.variables[invarname].long_name
dataset.close()
```

Figure 5.2. get_hycom_GLBa0.08_antartic_grid.py script

```
#-----------------------------------------------------
#Dimensoes do hycom exp 93.0 aprox 9 km lon x 5 km lat

lllat=0            # -80
urlat=875          # -45
lllon=3000         # 240 (-100)
urlon=4251         # 340 (-20)


#Download Hycom files for Antartic area

ano=2019
mes = '10'

invarname = 'water_temp'
outvarname = 'temp'

#Definindo as datas da leitura dos dados

for dia in range(2,3):
  data=str(ano)+str(mes)+str(dia)
  if dia < 10:
    data=str(ano)+str(mes)+'0'+str(dia)
  print ('Baixando arquivo do dia '+str(data))

#nome do arquivo de entrada
  url='ftp://ftp.hycom.org/datasets/GLBy0.08/expt_93.0/data/hindcasts/2019/hycom_glby_930_'+str(data)+'12_t000_ts3z.nc'

#Baixando o arquivo com WGET
  os.system('/usr/bin/wget -c ' +str(url))

#Lendo a grade e os atributos da variavel para o primeiro arquivo
  arquivo='hycom_glby_930_'+str(data)+'12_t000_ts3z.nc'
  dataset = netCDF4.Dataset(arquivo)
  var = dataset.variables[invarname][0,:,lllat+1:urlat+4,lllon-4:urlon+4]
  lon = dataset.variables['lon'][lllon-4:urlon+4]
  lat = dataset.variables['lat'][lllat+1:urlat+4]
  z = dataset.variables['depth'][:]
  spval = var.get_fill_value()
  units = dataset.variables[invarname].units
  long_name = dataset.variables[invarname].long_name
  dataset.close()
```

Figure 5.3. get_hycom_GLBa0.08_antartic_temp.py script

Note that in the grid script, the minimum and maximum longitude receive +5 and -5 extra values. In the download data script file, the minimum and maximum longitude receive -4 and +4 extra value, solving the ValueError. These values worked well in the Antartic grid. This exercise dows not have a specific rule, it is necessary to start from a value of your desire and observe the obtained results.

When checking the initial and countour files, it should be noted that the edges are with the correct values. If any of the edges have extreme or zero values where they should have valid values, you must redo the files by testng other extra value for longitude and latitude. The initial condition file, if it is wrong, will also present extreme values, and an unusual pattern for the region.

When generating a long-term series, check if there is any missing day from the sries, ai it will generate zero values for this day and when running the model, it wil l propably interrupt the simulation. The initial and boundary condition files should look similar to the figures 5.4 and 5.5.

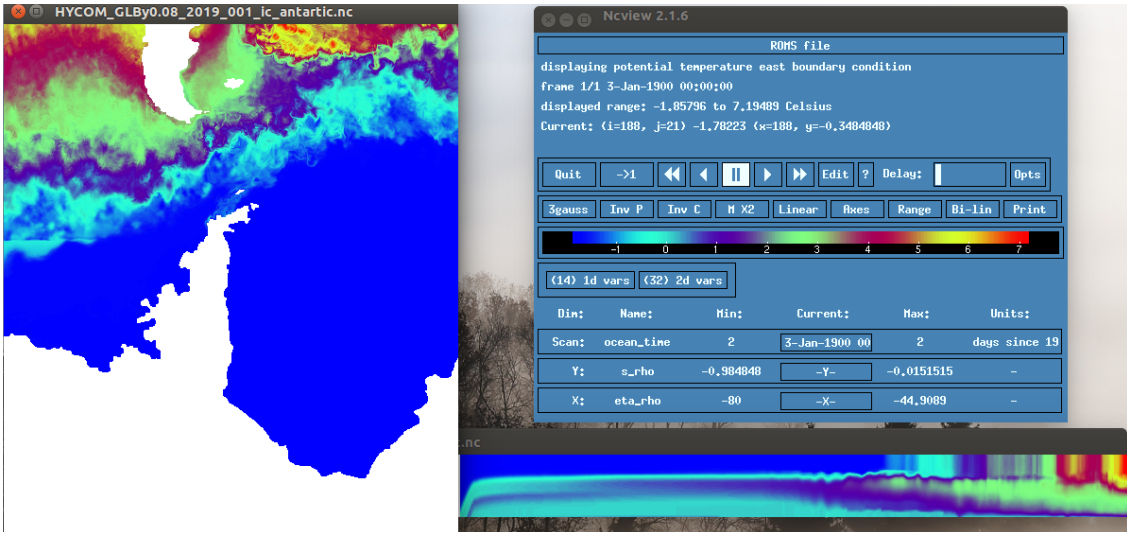


Figure 5.4. Initial condition and east temperature boundary.

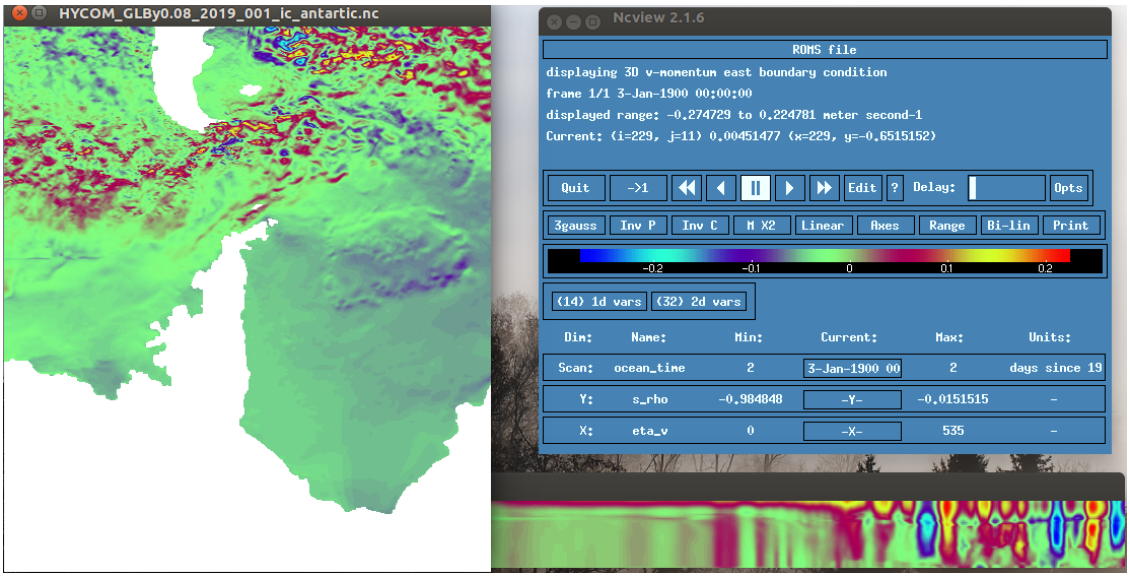


Figure 5.5. Initial condition and east boundary of the current *v* component.

5.3 The model2roms toolbox

In this section, you will be introduced to install model2roms, the toolbox used to generate ROMS forcing files and its grid. It was created and is being updated by Trond Kristiansen (**Trond2019**) and we adapted to the needs of LOA-INPE.

To download *model2roms*, enter *home* on your computer and download using Github:

```
cd $HOME
git clone https://github.com/usutil/model2roms
```

Change the folder name from *model2roms-master* to *model2roms*:

```
mv model2roms-master model2roms
```

5.3.1 Introduction to the ROMS grid

To generate the ROMS grid, we will use the code *make_roms_grid.py*, located in the *model2roms/grid* directory. The code supports the SRTM30_plus data (**Becker2009**), however, for practical purposes, we will exemplify only the data from ETOPO1 (**Amante2009**), which will be discussed in the next subsection.

Install basemap to use the code, if you have not previously installed

```
conda install -c anaconda basemap
```

5.3.2 ETOPO1 1 Arc-Minute Global Relief Model

The ETOPO1 (**Amante2009**; Figure 5.6) is produced by the National Geophysical Data Center and provides two layers of relief information. The layers include bathymetry over the oceans and some of the main lakes on Earth. Terrestrial topography and ocean bathymetry are based on SRTM30 topography and through various bathymetric cruises

Visit the site *https://www.ngdc.noaa.gov/mgg/global/global.html* and subset your area of interes on the ETOPO1 website map and download the data in NetCDF format.

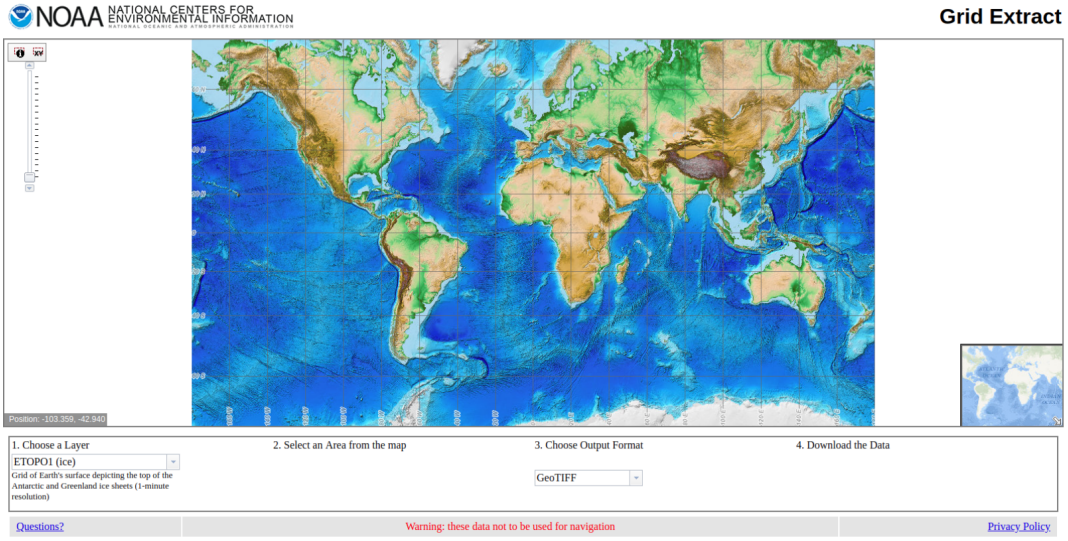


Figure 5.6. ETOPO1 webpage.

> **WARNING** The script *make_roms_grid.py* does not allow the selection of the latitude and longitude limits of the grid, therefore, be as accurate as possible when subseting of your area.

Creating the ROMS grid

Enter the *model2roms* directory and look for the *grid* subfolder. Open the code *make_roms_grid.py*.

```
cd $HOME/model2roms/grid
gedit make_roms_grid.py
```

As Figure 5.7 shown, the variables that will be modified are:

- **grd_name**: Grid name;
- **grd_final**: The NetCDF file name of the grid;
- **etopo1_dir**: The directory where the ETOPO1 NetCDF file is located;
- **srtm_dir**: The directory where the SRTM_30_PLUS NetCDF file is located;
- **hmin**: Value of *h* parameter;
- **theta_b**: Control parameter for coordinates close to the ocean floor;
- **theta_s**: Control parameter for coordinates close to the surface;
- **Tcline**: Critical depth, in meters, controlling the elongation of the coordinates following the terrain. It can be interpreted as the width of the surface where vertical resolution is best;
- **N**: Number of Sigma layers;
- **rmax**: Topography smoothing factor;
- **intrp_method**: Grid interpolation method: Linear (*linear*) or Near Neighbor (*nn*);
- **grid_resolution**: Grid resolution;
- **max_depth**: Maximum grid depth.

> **WARNING** In *grid_resolution*, it is necessary to place the numerator to do the ETOPO1 calculation. For example: as ETOPO1 has 1/60° of spatial resolution, if the chosen spatial resolution is 1/12°, the value of *grid_resolution* will be 5, because 5/60° is the same as 1/12° .

> **WARNING** As we do not use the data from SRTM_30_PLUS, the variable *srtm_dir* does not need to be modified.

```
"""
File name:      make_roms_grid.py
Author:         Ueslei Adriano Sutil
Email:          uesleisutil@gmail.com
Created:        17 June 2016
Last modified: 04 July 2019
Version:        6.1

Variable options:
    - grd_name          As mentioned in gridid.txt Pyroms file;
    - intrp_method      'linear' or 'nn' for nearest neighborhood;
    - grid_resolution   The ETOPO1 spatial resolution is 1/60°. If you desire a 1/12° spatial resolution, enter 5 (5/60 = 1/12°);
                        The SRTM30+ spatial resolution is 1/120°. If you desire a 1/12° spatial resolution, enter 10 (10/120 = 1/12°).
    - max_depth         The grid max depth (m).

Download ETOPO1 here:
    https://rda.ucar.edu/datasets/ds759.4/index.html#!description
Reference:
    Amante, C. and B. W. Eakins, 2009: ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis. 24, NOAA Technical Memorandum NESDIS NGDC, 19 pp.

Download SRTM30_plus here:
    https://topex.ucsd.edu/WWW_html/srtm30_plus.html
Reference:
    Becker, J. J., D. T. Sandwell, W. H. F. Smith, J. Braud, B. Binder, J. Depner, D. Fabre, J. Factor, S. Ingalls, S-H. Kim, R. Ladner, K. Marks, S. Nelson, A. Pharaoh, R. Trimmer, J. Von Rosenberg, G. Wallace, P. Weatherall., Global Bathymetry
"""

from   mpl_toolkits.basemap import Basemap
from   bathy_smoother       import bathy_tools, bathy_smoothing
import mpl_toolkits.basemap as mp
import numpy                as np
import netCDF4
import pyroms
import pyroms_toolbox

# Outputs names.
grd_name            = 'antartic'
grd_final           = grd_name+'_grd.nc'
etopo1_dir          = '/home/usuario/model2roms/grid/etopo1.nc'
srtm_dir            = '/home/usuario/model2roms/grid/topo30.nc'

# Grid settings.
hmin                = 25
theta_b             = 2.0
theta_s             = 7.0
Tcline              = 200
N                   = 40
rmax                = 0.25
intrp_method        = 'linear'
grid_resolution     = 5
max_depth           = -5000


# NOTE: Do not need to change above.

# Open bathymetry file.
print("Choose the Digital Elevation Model: (1) ETOPO1 or (2) SRTM30+, then press the [ENTER] buttom:")
grid_choice = input()
if grid_choice=="1":
    data = netCDF4.Dataset(etopo1_dir, 'r')
    lons = data.variables[u'x'][:]
    lats = data.variables[u'y'][:]
    cota = data.variables[u'z'][:]
    cota = np.array(cota, dtype='float32')
elif grid_choice=="2":
```

Figure 5.7. Screenshot of the *make_grid.py* file, which is located in the *model2roms/grid* folder.

After making the changes, just run the code. Type it:

```
ipython make_roms_grid.py --pylab
```

5.3.3 Introduction to ROMS forcing files

We will use *model2roms* to generate the ROMS conditions. The toolbox supports data from GLORYS12V1 (**Fernandez2018**) and SODA3 (**Carton2018**), however, we will only use GLORYS12V1 as an example.

5.3.4 Global Ocean Physics Reanalysis (GLORYS)

GLORYS is a global reanalysis of the ocean with a spatial resolution of 1/12° with daily or monthly temporal resolution and 50 vertical levels. It is based on the current CMEMS global weather forecasting system and is forced by ECMWF's ERA-Interim from the NEMO model. A Kalman filter with reduced order is used to assimilate the sea altimetry, sea surface temperature, sea ice concentration data obtained from remote sensing and the *in situ* data of vertical temperature and salinity profiles . In addition, a 3D-VAR scheme provides a correction for temperature and salinity deviations.

GLORYS can be downloaded from the website: *http://marine.copernicus.eu/services-portfolio/access-to-products/?option=com_csw&view=details&product_id=GLOBAL_REANALYSIS_PHY_001_030*.

When you try to download, the site will warn you that you will need to create a Copernicus account. Register and choose the GLORYS daily data set, as shown in Figure 5.8.

Figure 5.8. Screenshot from GLORYS webpage.

Click on *Dowload product* and, on the next page (Figure 5.9), select the area according to the latitude and longitude limits of the chosen grid. Also choose the period of data to be downloaded and if you want to use the sea ice model in ROMS, select all variables, with the exception of:

- *bottomT - Sea floor potential temperature*;
- *mlotst - Density ocean mixed layer thickness*.

If you choose to not use the sea ice model, do not select the variables:

- *siconc - Ice concentration*;
- *sithick - Sea ice thickness*;
- *usi - Sea ice eastward velocity*;
- *vsi - Sea ice northward velocity*.

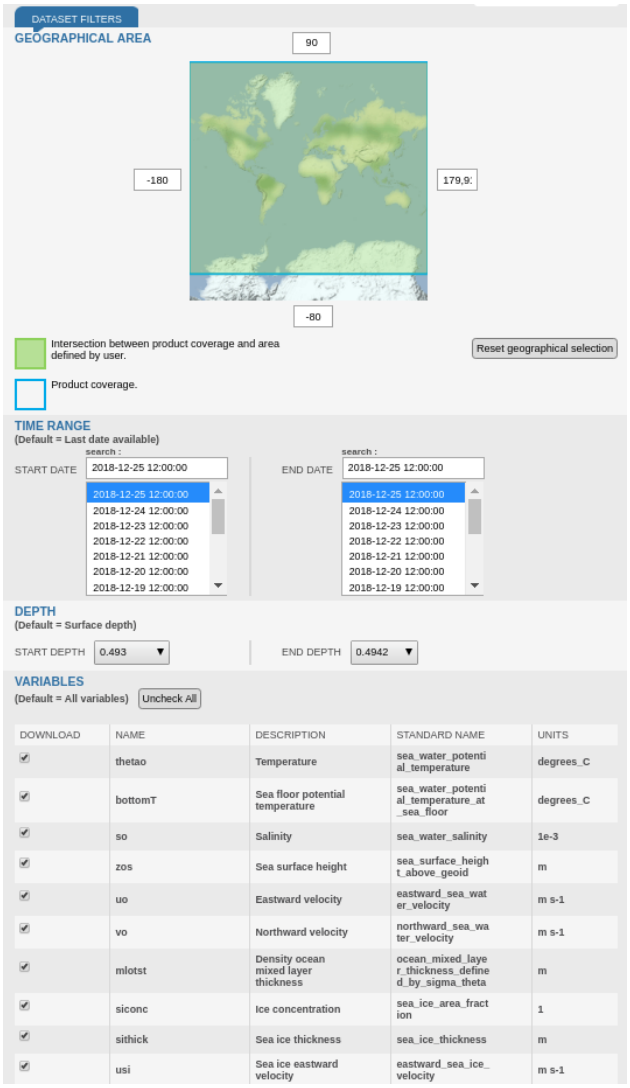


Figure 5.9. Another screenshot from the GLORYS webpage.

The Copernicus server allow files for download with a maximum of 1024 mb. If the selected period has a larger file size, as for example in Figure 5.10, it will be necessary to return to the previous step and partition the files into smaller pieces or download the complete file by clicking on *FTP ACCESS* (Figure 5.10).

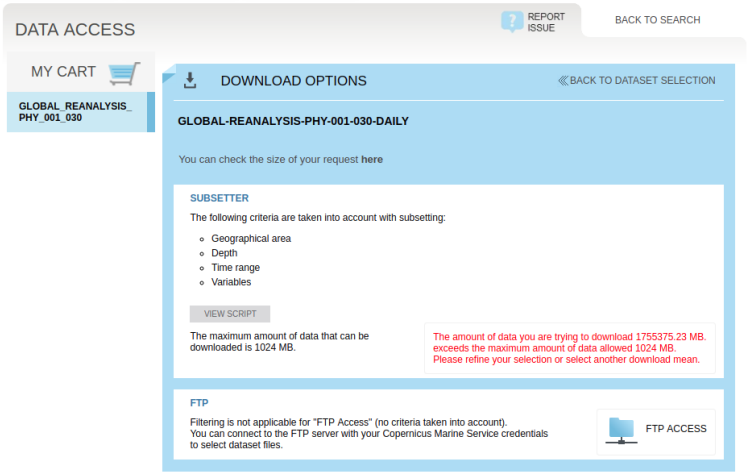


Figure 5.10. Error message when trying to download a text larger than 1024 mb.

If you choose to partition the files or download the files by *FTP*, you will need to create a new file with all the concatenated chosen dates. In this case, it is necessary to use *Climate Data Operators* (CDO).

There are two ways to install CDO: by *Conda* or by *apt-get*.

If you choose to use *Conda*, type:

```
conda install -c conda-forge cdo
```

If you choose to use *apt-get*, type:

```
sudo apt-get install cdo
```

After installation, enter the directory where all GLORYS files are located and type:

```
cdo cat glorys* glorys.nc
```

Where:

- *cdo* means the program command;
- *cat* means the concatenation command of all files;
- *glorys** means that all files called *glorys* will be concatenated within the folder;
- *glorys.nc* means the final file name created by concatenating with *CDO*.

WARNING Para facilitar os próximos passos, coloque o arquivo do GLORYS dentro do diretório *$HOME/model2roms/input*

5.3.5 Creating ROMS forcing files

When opening the *model2roms* folder, it is possible to observe several script files. We will start with the file *compile.py*, which will compile several files in Fortran90.

Compile the files with the command:

```
ipython compile.py --pylab
```

If your *GLORYS* file name is different from *glorys.nc*, open the code *forcingFilenames.py* (Figure 5.11) and in row 15, change *'glorys.nc'* to the name of the created file.

```
cd $HOME/model2roms
gedit forcingFilenames.py
```

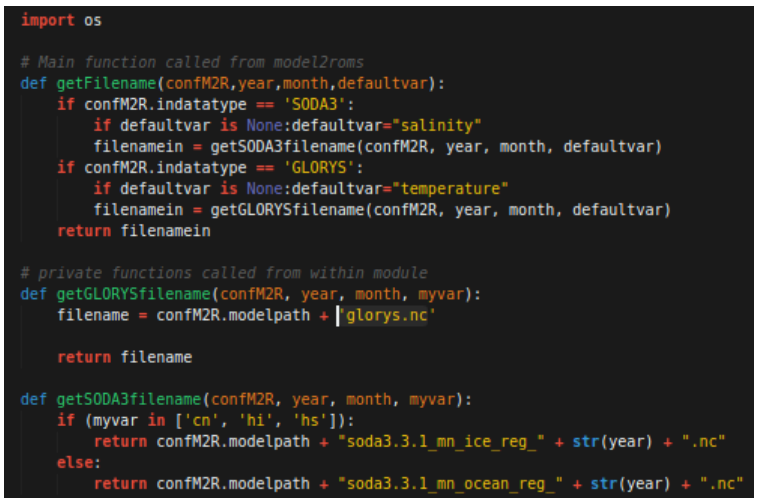


Figure 5.11. Screenshot of the *forcingFilename.py* file.

Open the file *configM2R.py* and, in row 16 (Figure 5.12), change the abbreviation *SuaAbreviaçãoAqui* of the *defineabbreviation* to the name of your choice.

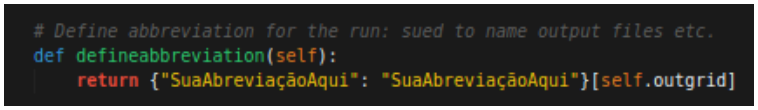


Figure 5.12. Screenshot of the *defineabbreviation* in the code *configM2R.py*.

In row 63 (Figure 5.13), change the grid directory *Minha_Grade.nc* to your grid directory and the abbreviation *SuaAbreviaçãoAqui* by the name chosen in the previous step.

```
def defineromsgridpath(self):
    return {'SuaAbreviaçãoAqui': '/home/usuario/model2roms/grid/Minha_Grade.nc'}[self.outgrid]
```

Figure 5.13. Screenshot of the *configM2R.py* code.

On row 67 (Figure 5.14), change the *GLORYS* file directory to your chosen path.

```
def defineforcingdatapath(self):
    return {'SODA3': "/home/usuario/Documentos/model2roms/input/",
            'GLORYS': "/home/usuario/Documentos/model2roms/input/"}[self.indatatype]
```

Figure 5.14. Screenshot of the *configM2R.py* code.

From row 76 (Figure 5.15), modify according to your project:

- **self.compileall**: Set as *True* if you want the Fortran files to be recompiled each time *model2roms* is executed;
- **self.createoceanforcing**: Set as *True* to create the hydrodynamic variables;
- **self.createatmosforcing**: Set as *True* to create atmospheric forces. **Currently this function is in the testing phase and is not entirely available for use**;
- **self.writeice**: Set as *True* to create the sea ice variables;
- **self.set2DvarsToZero**: Creates the ice and sea level files with zero values. Since GLORYS has these values, it is recommended to leave it as *False*;
- **self.useesmf**: Set as *True* to use *ESMF* to interpolate the GLORYS data to the ROMS grid;
- **self.usefilter**: Set as *True* to apply a filter to smooth out 2D fields.
- **self.myformat**: The extension for writing ROMS input files. It is, by default, *NETCDF*;
- **self.timefrequencyofinputdata**: The time frequency of the input data. If using GLORYS, write *'day'*;
- **self.indatatype**: The name of the data used to generate the forcing files. *'GLORYS* or *SODA3*;
- **self.authorname**: The name of the user using *model2roms*;
- **self.authoremail**: The email of the user using *model2roms*;
- **self.ingridtype**: It will interpolate the GLORYS grid, which is in *'ZLEVEL'* coordinate;
- **self.grdtype**: The GLORYS grid type, which is *'regular'*;
- **self.lonname**: The name of the GLORYS longitude variable, which is *'longitude'*;
- **self.latname**: The name of the GLORYS latitude variable, which is *'latitude'*;
- **self.depthname**: The name of the GLORYS depth variable, which is *'depth'*;
- **self.lonname_u**: The name of the GLORYS U-longitude variable, which is *'longitude'*;
- **self.latname_u**: The name of the GLORYS U-latitude variable, which is *'latitude'*;
- **self.lonname_v**: The name of the GLORYS V-longitude variable, which is *'longitude'*;
- **self.latname_v**: The name of the GLORYS U-latitude variable, which is *'latitude'*;
- **self.timename**: The name of the GLORYS time variable, which is *'time'*;
- **self.realm**: The realm in which *model2roms* is being executed, in this case, *'ocean'*;
- **self.fillvaluein**: The *fillvalue* value of the NetCDF file. By convention, *-1.e20*;
- **self.outgrid**: The name of the abbreviation used in the definition *defineabbreviation*;
- **self.outgridtype**: If *"ROMS"*, it will create the output in the standard ROMS format;
- **self.nlevels**: The number of vertical levels in the grid. It must be the same as the one chosen in the code *make_roms_grid.py*;
- **self.vstretching**: It must be the same as the one chosen in the code *make_roms_grid.py*;

- **self.vtransform**: It must be the same as the one chosen in the code *make_roms_grid.py*;
- **self.theta_s**: It must be the same as the one chosen in the code *make_roms_grid.py*;
- **self.theta_b**: It must be the same as the one chosen in the code *make_roms_grid.py*;
- **self.tcline**: It must be the same as the one chosen in the code *make_roms_grid.py*;
- **self.hc**: It must be the same as the one chosen in the code *make_roms_grid.py*;
- **self.start_year**: The initial year of GLORYS data;
- **self.end_year**: The final year of GLORYS data;
- **self.start_month**: The initial month of GLORYS data;
- **self.end_month**: The final month of GLORYS data;
- **self.start_day**: The initial day of GLORYS data;
- **self.end_day**: The final day of GLORYS data;

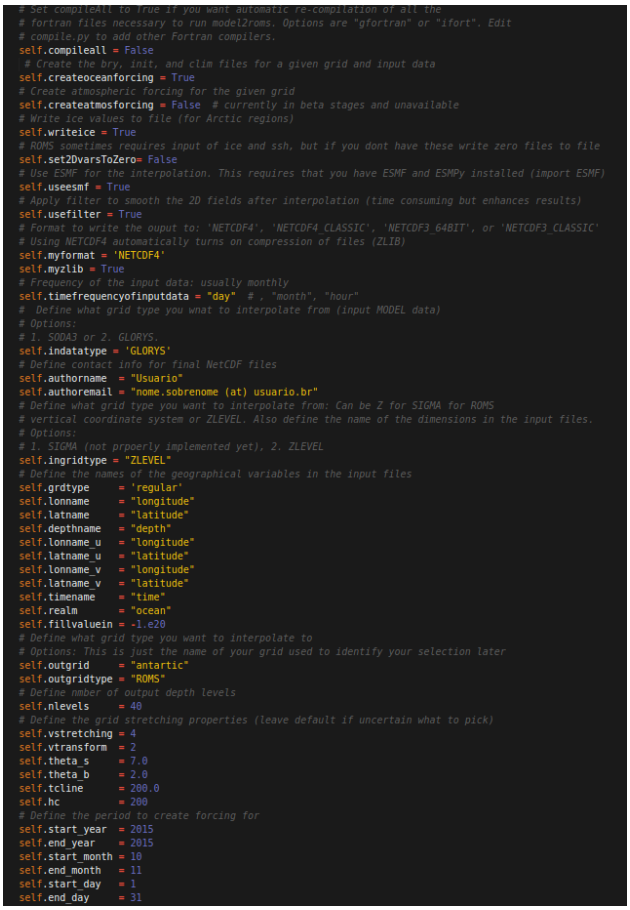


Figure 5.15. Screenshot of the *configM2R.py* code.

Run *model2roms* with the command:

```
ipython runM2R.py
```

At the end, the initial, boundary and climatological files will be created, which should be added to your project folder, in the Kerana cluster.

6. Building the SWAN

SWAN uses the *SWAN.EDT* file to read the grid (*swan_coord.grd*) and bathymetry (*swan_bathy.bot*) files. It is also possible to define a numerical grid within the *SWAN.EDT* and indicate the bathymetry file that will be read and associated with the defined grid.

As an example, SWAN files will be created from the ROMS grid, but without the introduction of initial data, as the implementation of a grid inside *SWAN.EDT* is not trivial.

For SWAN boundary conditions, the wind fields will come from the WRF, as soon as the waves generated are associated with the atmospheric systems represented in these simulations. Without the information the boundary files information, the simulated waves will be generated only within the boundary of the domain, disregarding the energy that is leaving or entering the grid.

We will use the MATLAB script, *make_swan.m*, to generate the files. The script is located at:

```
1 /home/name.surname/repositorio/SWAN_scripts
```

As shown in Figure 6.1, the script has the following construction:

```
clear all
ncfile   = '../roms_grid.nc'
x_rho    = ncread(ncfile,'lon_rho');
y_rho    = ncread(ncfile,'lat_rho');
h        = ncread(ncfile,'h');
mask_rho = ncread(ncfile,'mask_rho');

%Replace the land positions with the flag for land (defined in the SWAN
%input file)
land_values = find(mask_rho == 0);
h(land_values) = 9999;

%Print the depths to the bathy file
[n,m] = size(h);

fid = fopen('swan_bathy.bot','w');
for index1 = 1:m;
    for index2 = 1:n;
        fprintf(fid,'   ');
       fprintf(fid,'%12.8f',h(index2,index1));
    end
    fprintf(fid,'\n');
end

%Print the grid coordinates to the grid file
fid = fopen('swan_coord.grd','w');
fprintf(fid,'%12.6f\n',x_rho);
fprintf(fid,'%12.6f\n',y_rho);
```

Figure 6.1. *make_swan.m* script.

To generate the two SWAN files, look in the script for the variable *ncfile* and change the directory to the path where your ROMS grid is.

Run the script and the two files will be created: *swan_coord.grd* and *swan_bathy.bot*. The two files must be placed inside your project folder.

7. Building the Budgell's Sea Ice Model

The sea ice model is fully coupled to ROMS, so that, after generating the conditions of the ROMS with ice data (see Section 5.3), to activate the sea ice model just modify the ROMS *.h* file in your project. According to **hedstrom2018**, you can add the following options into the *.h* file, as shown in Figure 7.1:

- **ICE_MODEL**: defines the sea ice model;
- **ANA_ICE**: defines initial analytical conditions for sea ice;
- **ICE_THERMO**: defines the ice thermodynamics;
- **ICE_MK**: defines the **Mellor1989** ice thermodynamics. Currently, is the only choice;
- **ICE_MOMENTUM**: defines the momentum component of the ice;
- **ICE_MOM_BULK**: defines the alternate ice-water stress computation;
- **ICE_EVP**: defines the elastic-viscous-plastic rheology from **Hunke1997** and **Hunke2001**;
- **ICE_QUAD_STRENGTH**: defines the quadratic ice strength from **Overland1988**;
- **ICE_ADVECT**: defines the advection of ice tracers;
- **ICE_SMOLAR**: defines the MPDATA use for ice tracers. Currently, it is the only option;
- **ICE_UPWIND**: defines the upwind advection;
- **ICE_BULK_FLUXES**: define the ice part of bulk flux computation;
- **ICE_DIAGS**: defines the diagnosis of sea ice;
- **ICE_SHOREFAST** : defines the simple shorefast-ice algorithm from **Budgell2005**;
- **ICE_I_O**: defines to allow light into the ice as heat;
- **ICE_CONVSNOW**: defines the conversion of flooded snow to ice.

```
#ifdef SOLVE3D
# define ICE_MODEL
# ifdef ICE_MODEL
#  define ANA_ICE
#  define ICE_THERMO
#  define ICE_MK
#  define ICE_MOMENTUM
#  define ICE_MOM_BULK
#  define ICE_EVP
#  define ICE_STRENGTH_QUAD
#  define ICE_ADVECT
#  define ICE_SMOLAR
#  define ICE_UPWIND
#  define ICE_BULK_FLUXES
#  define ICE_I_O
#  define ICE_DIAGS
# endif
#endif
```

Figure 7.1. Screenshot of ROMS *.h* file

For more information about the model, it is recommended to read **hedstrom2018**.

Get the *.in* file for the sea ice model. Copy the file *ice.in* into the Kerana repository and add it to your project.

```
/home/name.surname/repositorio/ICE_scripts
```

Modify the ROMS *.in* file, pointing to the *ice.in* into the variable *IPARNAM*:

```
nedit ocean.in
IPARNAM =  ice.in
```

8. Building the weights between grids with SCRIP

8.1 Building the weights

As seen in the section 1.6, SCRIP is used to interpolate the weights between two or more grids of different models. In COAWST, the package was modified to generate only one NetCDF file that will be used during integrations.

The SCRIP directory is located at:

```
1 /home/name.surname/COAWST/Lib/SCRIP
```

Inside the folder, look for the file with the extension.*in*. As in the example in Figure 8.1:

```
$INPUTS
!
!  Input file for scrip_coawst.
!  The $INPUTS line is required at the top of this file.
!  Edit this file to enter the correct information below.
!  Then run this program as "scrip_coawst scrip_coawst_sandy.in"
!
! 1) Enter name of output netcdf4 file
!
OUTPUT_NCFILE='scrip_sandy_moving.nc'
!OUTPUT_NCFILE='scrip_sandy_static.nc'

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

! 3) Enter name of the ROMS grid file(s):
!
ROMS_GRIDS(1)='/home/nome.sobrenome/COAWST/Projects/Sandy/Sandy_roms_grid.nc',
ROMS_GRIDS(2)='/home/nome.sobrenome/COAWST/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, and
!    -if the swan grids are Spherical(set cartesian=0) or
!                           Cartesian(set cartesian=1).
!
SWAN_COORD(1)='/home/nome.sobrenome/COAWST/Projects/Sandy/Sandy_swan_coord.grd',
SWAN_COORD(2)='/home/nome.sobrenome/COAWST/Projects/Sandy/Sandy_swan_coord_ref3.grd',
SWAN_BATH(1)='/home/nome.sobrenome/COAWST/Projects/Sandy/Sandy_swan_bathy.bot',
SWAN_BATH(2)='/home/nome.sobrenome/COAWST/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 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)='/home/nome.sobrenome/COAWST/Projects/Sandy/wrfinput_d01',
!WRF_GRIDS(2)='/home/nome.sobrenome/COAWST/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

!
!  The $END statement below is required
```

Figure 8.1. SCRIP *.in* file for the Sandy project.

In *OUTPUT_NCFILE*, change the name of the NetCDF file that will be generated if necessary.

In section 2 of the file, change the variables *NGRIDS_ROMS*, *NGRIDS_SWAN* and *NGRIDS_WRF* according to the number of grids, existing in your project, in ROMS, SWAN and WRF, respectively.

In the third section of the file, renew the ROMS grid directories according to the names in your project.

For SWAN, in the fourth section, in addition to changing the directories of the SWAN grids (*SWAN_COORD* and *SWAN_BATH*), change the number of existing grid points, according to your project , in the variables *SWAN_NUMX* and *SWAN_NUMY*.

Finally, in the fifth section, change the WRF grid directories (*WRF_GRIDS*). In *PARENT_GRID_RATIO*, if your project includes nesting between the WRF grids, change to the relationship used between the grids used in your project. In *PARENT_ID*, add the grid ID.

Then, save the changes.

To run SCRIP, search the repository for the file *qsub_scrip.sh*:

```
1 /home/name.surname/repositorio/qsub_scrip.sh
```

Move the file to the SCRIP directory:

```
1 mv /home/name.surname/repositorio/qsub_scrip.sh /home/name.surname/COAWST/Lib/SCRIP
```

Open the file *qsub_scrip.sh*:

```
nedit qsub_scrip.sh
```

Change and save the file *.sh*, as in the example in Figure 8.2:

```
#!/bin/sh
#PBS -l mppwidth=1
#PBS -N SCRIP
#PBS -j oe
#PBS -o Joe_test.out
#PBS -l walltime=35600:00:00
#PBS -q workq
  echo "Running GEOGRID on KERANA"
#
  export MPICH_ENV_DISPLAY=1
  export MPICH_ABORT_ON_ERROR=1
  export MPICH_RANK_REORDER_DISPLAY=1
  export MPICH_RANK_REORDER_METHOD=1
  export MALLOC_MMAP_MAX_=0
  export MALLOC_TRIM_THRESHOLD_=536870912
  export OMP_NUM_THREADS=1
#
  EXECDIR=/scratch/nome.sobrenome/COAWST/Lib/SCRIP
  chmod 755 *
# export  ATP_ENABLED=1
# ulimit -a
# ulimit -c unlimited
# ulimit -s unlimited
# ulimit -m unlimited
# ulimit -a


 cd $EXECDIR

 aprun -n 1 scrip_coawst /scratch/nome.sobrenome/COAWST/Lib/SCRIP/scrip_coawst_sandy.in 1> log.out 2> log.err
```

Figure 8.2. *qsub_scrip.sh* file used to run SCRIP.

To start SCRIP, type:

```
qsub qsub_scrip.sh
```

At the end, the file *scrip_static.nc* will be created. Now put them in your project folder and you're done! COAWST is ready to run.

8.2 Running your simulation

Now, with everything ready, your project is ready to be started. Visit the section 2.10 to remember how to execute the project.

9. Papers of the Ocean and Atmosphere Studies Laboratory (LOA/INPE)

Ocean-Atmosphere Interactions in an Extratropical Cyclone in the Southwest Atlantic

U. A. Sutil, L. P. Pezzi, R. C. M. Alves and A. B. Nunes

Abstract

This work shows an investigation of the behavior of heat fluxes in the processes of ocean-atmosphere interaction during the passage of an Extra-tropical Cyclone (EC) in the Southwest Atlantic in September 2006 using a coupled regional model's system. A brief evaluation of the simulated data is done by comparison with air and sea surface temperature (SST) data, wind speed, sea level pressure. This comparison showed that both model simulations present some differences (mainly, the wind), nevertheless the simulated variables show quite satisfactory results, therefore allowing a good analysis of the ocean-atmosphere interaction processes. The simulated thermal gradient increases the ocean's heat fluxes into the atmosphere in the cold sector of the cyclone and through the convergence of low level winds the humidity is transported to higher levels producing precipitation. The coupled system showed a greater ability to simulate the intensity and trajectory of the cyclone, compared to the simulation of the atmospheric model.

Sutil2019

Avaliable at: *http://www.anuario.igeo.ufrj.br/2019_01/2019_1_525_535.pdf*

Low connectivity compromises the conservation of reef fishes by marine protected areas in the tropical South Atlantic

C. A. K. Endo, D. F. M. Gherardi, L. P. Pezzi and L. N. Lima

Abstract

The total spatial coverage of Marine Protected Areas (MPAs) within the Brazilian Economic Exclusive Zone (EEZ) has recently achieved the quantitative requirement of the Aichii Biodiversity Target 11. However, the distribution of MPAs in the Brazilian EEZ is still unbalanced regarding the proportion of protected ecosystems, protection goals and management types. Moreover, the demographic connectivity between these MPAs and their effectiveness regarding the maintenance of biodiversity are still not comprehensively understood. An individual-based modeling scheme coupled with a regional hydrodynamic model of the ocean is used to determine the demographic connectivity of reef fishes based on the widespread genus Sparisoma found in the oceanic islands and on the Brazilian continental shelf between 10° N and 23° S. Model results indicate that MPAs are highly isolated due to extremely low demographic connectivity. Consequently, low connectivity and the long distances separating MPAs contribute to their isolation. Therefore, the current MPA design falls short of its goal of maintaining the demographic connectivity of Sparisoma populations living within these areas. In an extreme scenario in which the MPAs rely solely on protected populations for recruits, it is unlikely that they will be able to effectively contribute to the resilience of these populations or other reef fish species sharing the same dispersal abilities. Results also show that recruitment occurs elsewhere along the continental shelf indicating that the protection of areas larger than the current MPAs would enhance the network, maintain connectivity and contribute to the conservation of reef fishes.

Endo2019

Avaliable at: *https://www.nature.com/articles/s41598-019-45042-0*

An Investigation of Ocean Model Uncertainties Through Ensemble Forecast Experiments in the Southwest Atlantic Ocean

L. N. Lima, L. P. Pezzi, S. G. Penny and C. A. S. Tanajura

Abstract

Ocean general circulation models even with realistic behavior still incorporate large uncertainties from external forcing. This study involves the realization of ensemble experiments using a regional model configured for the Southwest Atlantic Ocean to investigate uncertainties derived from the external forcing such as the atmosphere and bathymetry. The investigation is based on perturbing atmospheric surface fluxes and bathymetry through a series of ensemble experiments. The results showed a strong influence of the South Atlantic Convergence Zone on the underlying ocean, 7 days after initialization. In this ocean region, precipitation and radiation flux perturbations notably impacted the sea surface salinity and sea surface temperature, by producing values of ensemble spread that exceeded 0.08 and 0.2 °C, respectively. Wind perturbations extended the impact on currents at surface, with the spread exceeding 0.1 m/s. The ocean responded faster to the bathymetric perturbations especially in shallow waters, where the dynamics are largely dominated by barotropic processes. Ensemble spread was the largest within the thermocline layer and in ocean frontal regions after a few months, but by this time, the impact on the modeled ocean obtained from either atmospheric or bathymetric perturbations was quite similar, with the internal dynamics dominating over time. In the vertical, the sea surface temperature exhibited high correlation with the subsurface temperature of the shallowest model levels within the mixed layer. Horizontal error correlations exhibited strong flow dependence at specific points on the Brazil and Malvinas Currents. This analysis will be the basis for future experiments using ensemble-based data assimilation in the Southwest Atlantic Ocean.

Lima2019

Avaliable at: *hhttps://agupubs.onlinelibrary.wiley.com/doi/epdf/10.1029/2018JC013919*

Coupled ocean-atmosphere forecasting at short and medium time scales

J. Pullen, R. Allard, H. Seo, A. J. Miller, S. Chen, L. P. Pezzi, T. Smith, P. Chu, J. Alves and R. Caldeira

Abstract

Recent technological advances over the past few decades have enabled the development of fully coupled atmosphere-ocean modeling prediction systems which are used today to support short-term (days to weeks) and medium-term (10-21 days) needs for both the operational and research communities. Utilizing several coupled modeling systems we overview the coupling framework, including model components and grid resolution considerations, as well as the coupling physics by examining heat fluxes between atmosphere and ocean, momentum transfer, and freshwater fluxes. These modeling systems can be run as fully coupled atmosphere-ocean and atmosphere-ocean-wave configurations. Examples of several modeling systems applied to complex coastal regions including Madeira Island, Adriatic Sea, Coastal California, Gulf of Mexico, Brazil, and the Maritime Continent are presented. In many of these studies, a variety of field campaigns have contributed to a better understanding of the underlying physics associated with the atmosphere-ocean feedbacks. Examples of improvements in predictive skill when run in coupled mode versus standalone are shown. Coupled model challenges such as model initialization, data assimilation, and earth system prediction are discussed.

Pullen2017

Avaliable at: *http://meteora.ucsd.edu/∼miller/papers/TheSea_Chapter23.html*

Regional modeling of the water masses and circulation annual variability at the Southern Brazilian Continental Shelf

L. F. Mendonça, R. B. Souza, C. R. C. Aseff, L. P. Pezzi, O. O. Möller and R. C. M. Alves

Abstract

The Southern Brazilian Continental Shelf (SBCS) is one of the more productive areas for fisheries in Brazilian waters. The water masses and the dynamical processes of the region present a very seasonal behavior that imprint strong effects in the ecosystem and the weather of the area and its vicinity. This paper makes use of the Regional Ocean Modeling System (ROMS) for studying the water mass distribution and circulation variability in the SBCS during the year of 2012. Model outputs were compared to in situ, historical observations and to satellite data. The model was able to reproduce the main thermohaline characteristics of the waters dominating the SBCS and the adjacent region. The mixing between the Subantarctic Shelf Water and the Subtropical Shelf Water, known as the Subtropical Shelf Front (STSF), presented a clear seasonal change in volume. As a consequence of the mixing and of the seasonal oscillation of the STSF position, the stability of the water column inside the SBCS also changes seasonally. Current velocities and associated transports estimated for the Brazil Current (BC) and for the Brazilian Coastal Current (BCC) agree with previous measurements and estimates, stressing the fact that the opposite flow of the BCC occurring during winter in the study region is about 2 orders of magnitude smaller than that of the BC. Seasonal maps of simulated Mean Kinetic Energy and Eddy Kinetic Energy demonstrate the known behavior of the BC and stressed the importance of the mean coastal flow off Argentina throughout the year.

Mendonca2017

Avaliable at: *https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2016JC011780*

The Influence of Sea Ice Dynamics on the Climate Sensitivity and Memory to Increased Antarctic Sea Ice

C. K. Parise, L. P. Pezzi, K. I. Hodges and F. Justino

Abstract

The study analyzes the sensitivity and memory of the Southern Hemisphere coupled climate system to increased Antarctic sea ice (ASI), taking into account the persistence of the sea ice maxima in the current climate. The mechanisms involved in restoring the climate balance under two sets of experiments, which differ in regard to their sea ice models, are discussed. The experiments are perturbed with extremes of ASI and integrated for 10 yr in a large 30-member ensemble. The results show that an ASI maximum is able to persist for 4 yr in the current climate, followed by a negative sea ice phase. The sea ice insulating effect during the positive phase reduces heat fluxes south of 60°S, while at the same time these are intensified at the sea ice edge. The increased air stability over the sea ice field strengthens the polar cell while the baroclinicity increases at midlatitudes. The mean sea level pressure is reduced (increased) over high latitudes (midlatitudes), typical of the southern annular mode (SAM) positive phase. The Southern Ocean (SO) becomes colder and fresher as the sea ice melts mainly through sea ice lateral melting, the consequence of which is an increase in the ocean stability by buoyancy and mixing changes. The climate sensitivity is triggered by the sea ice insulating process and the resulting freshwater pulse (fast response), while the climate equilibrium is restored by the heat stored in the SO subsurface layers (long response). It is concluded that the time needed for the ASI anomaly to be dissipated and/or melted is shortened by the sea ice dynamical processes.

Parise2015

Avaliable at: *https://journals.ametsoc.org/doi/10.1175/JCLI-D-14-00748.1*

Modeling the spawning strategies and larval survival of the Brazilian sardine (*Sardinella brasiliensis*)

D. F. Dias, L. P. Pezzi, D. F. M. Gherardi and R. Camargo

Abstract

An Individual Based Model (IBM), coupled with a hydrodynamic model (ROMS), was used to investigate the spawning strategies and larval survival of the Brazilian Sardine in the South Brazil Bight (SBB). ROMS solutions were compared with satellite and field data to assess their representation of the physical environment. Two spawning experiments were performed for the summer along six years, coincident with ichthyoplankton survey cruises. In the first one, eggs were released in spawning habitats inferred from a spatial model. The second experiment simulated a random spawning to test the null hypothesis that there are no preferred spawning sites. Releasing eggs in the predefined spawning habitats increases larval survival, suggesting that the central-southern part of the SBB is more suitable for larvae development because of its thermodynamic characteristics. The Brazilian sardine is also capable of exploring suitable areas for spawning, according to the interannual variability of the SBB. The influence of water temperature, the presence of Cape Frio upwelling, and surface circulation on the spawning process was tested. The Cape Frio upwelling plays an important role in the modulation of Brazilian sardine spawning zones over SBB because of its lower than average water temperature. This has a direct influence on larval survival and on the interannual variability of the Brazilian sardine spawning process. The hydrodynamic condition is crucial in determining the central-southern part of SBB as the most suitable place for spawning because it enhances simulated coastal retention of larvae.

Dias2014

Avaliable at: *http://www.iag.usp.br/pos/meteorologia/biblio/modeling-spawning-strategies-and-larval-survival-brazilian-sardine-sardinella-br*

Sea surface temperature anomalies driven by oceanic local forcing in the Brazil-Malvinas Confluence

I. P. da Silveira and L. P. Pezzi

Abstract

Sea surface temperature (SST) anomaly events in the Brazil-Malvinas Confluence (BMC) were investigated through wavelet analysis and numerical modeling. Wavelet analysis was applied to recognize the main spectral signals of SST anomaly events in the BMC and in the Drake Passage as a first attempt to link middle and high latitudes. The numerical modeling approach was used to clarify the local oceanic dynamics that drive these anomalies. Wavelet analysis pointed to the 8–12-year band as the most energetic band representing remote forcing between high to middle latitudes. Other frequencies observed in the BMC wavelet analysis indicate that part of its variability could also be forced by low-latitude events, such as El Niño. Numerical experiments carried out for the years of 1964 and 1992 (cold and warm El Niño-Southern Oscillation (ENSO) phases) revealed two distinct behaviors that produced negative and positive sea surface temperature anomalies on the BMC region. The first behavior is caused by northward cold flow, Río de la Plata runoff, and upwelling processes. The second behavior is driven by a southward excursion of the Brazil Current (BC) front, alterations in Río de la Plata discharge rates, and most likely by air-sea interactions. Both episodes are characterized by uncoupled behavior between the surface and deeper layers.

Silveira2014

Avaliable at: *https://link.springer.com/article/10.1007%2Fs10236-014-0699-4*

Acknowledgments

To CNPq for the Institutional Training Scholarship of the National Institute for Space Research, **process 301110/2017-4**, provided to U. A. Sutil, the Research Productivity program grant awarded to L. P. Pezzi (CNPq 304009/2016-4). Also for the Antarctic Modeling and Observational System (ATMOS) project grant for both authors from the process CNPq/MCTIC/CAPES/FNDCT Nº 21/2018 (443013/2018-7).

To CAPES for promoting the Advanced Studies in Medium and High Latitudes Oceanography project (23038.004304/2014-28).

To Trond Kristiansen (*https://github.com/trondkr/model2roms*) for creating and sharing the *model2roms* toolbox.

To Kate Hedström for help and support to learn about the sea ice model.

To Jonas Takeo Carvalho for writing the Pyroms test case subsection.
To João Hackerott for his help in compiling the WPS in the Kerana cluster and for answering questions with the WRF.

To Matheus Fagundes for his help in Python.

To the entire modeling team that developed ROMS and distributed the source codes openly, especially to Hernan G. Arango.

To John Warner and all the creators and collaborators of COAWST for developing and making the codes available free of charge.

To Mathias Legrand, Vel and Andrea Hidalgo for the template on LaTeX.