



Guia prático para utilização do COAWST 3.4

Ueslei Adriano Sutil
Luciano Ponzi Pezzi



Laboratório de Estudos do Oceano e da Atmosfera
OBT - INPE

UESLEI ADRIANO SUTIL
uesleisutil1@gmail.com

LUCIANO PONZI PEZZI
luciano.pezzi@inpe.br

Produzido através do Programa de Capacitação Interna do MCTIC/CNPq (Processo 301110/2017-4).

Conteúdo licenciado sob *Creative Commons Attribution-NonCommercial 3.0 Unported* (CC BY-NC 3.0). Para obter uma cópia da licença, acesse: <http://creativecommons.org/licenses/by-nc/3.0>. A figura A Grande Onda de Kanagawa, que compõe o cabeçalho de cada capítulo, e de autoria de Katsushika Hokusai, está disponível para domínio público.

Template Legrand Orange Book produzido por Mathias Legrand (legrand.mathias@gmail.com) e modificado por Vel (vel@latextemplates.com) e Andrea Hidalgo. Licenciado sob *Creative Commons Attribution-NonCommercial 3.0 Unported* (CC BY-NC 3.0).

Versão 2.0
Julho 2019





Sumário

Nota dos autores 7

1	Introdução	9
1.1	O Regional Ocean Modeling System	9
1.2	O Weather Research & Forecasting Model	10
1.3	O Simulating Waves Nearshore	11
1.4	O Model Coupling Toolkit	11
1.5	O Spherical Coordinate Remapping Interpolation Package	11
1.6	O Coupled-Ocean-Atmosphere-Wave-Sediment Transport Modeling System	12
1.7	O Python	13
1.8	Materiais necessários para o uso do guia	13
2	O COAWST no cluster Kerana	15
2.1	Sobre o Kerana	15
2.2	Registrando uma conta de usuário	15
2.3	Acessando o cluster Kerana	16
2.4	Repositório de arquivos	17
2.5	O ambiente do Kerana	17
2.6	Baixando o COAWST	18
2.7	Automatizando a compilação do COAWST no Kerana	18
2.8	Compilando o MCT	19

2.9	Compilando o caso-teste Sandy	20
2.9.1	Pasta Projects	21
2.9.2	Pasta Work	23
2.9.3	Compilando o caso teste	24
2.10	Simulando o caso teste Sandy	26
3	Características do COAWST	29
3.1	Os arquivos estruturais dos modelos	29
3.2	Modificando o número de processadores	30
3.2.1	Modificando os processadores no ROMS	30
3.2.2	Modificando os processadores no WRF	30
3.2.3	Modificando os processadores no SWAN	30
3.2.4	Modificando os processadores no COAWST	30
3.3	Modificando o intervalo de tempo do acoplamento entre os modelos	31
4	Construindo o seu projeto no WRF	33
4.1	Compilando o WRF no Kerana	33
4.2	O WRF Preprocessing System (WPS)	34
4.2.1	Download do WPS	35
4.2.2	Compilando o WPS no Kerana	35
4.3	Construindo o seu projeto utilizando o CFSR	37
4.3.1	O geogrid	39
4.3.2	Utilizando o NCL para plotar o domínio da grade	41
4.3.3	O ungrid	43
4.3.4	O metgrid	45
4.3.5	O real	45
4.4	Usando o WRF	47
5	Construindo o ROMS	49
5.1	Instalando as bibliotecas do Python no computador pessoal	49
5.1.1	Instalação manual	49
5.1.2	Utilizando o Conda	60
5.2	Instalando o Pyroms	60
5.3	O model2roms	63
5.3.1	O ETOPO1 1 Arc-Minute Global Relief Model	63
5.3.2	Gerar a grade do ROMS	63
5.3.3	O Simple Ocean Data Assimilation (SODA)	65
5.3.4	Gerar as condições do ROMS	65
5.4	Compilando o ROMS com o modelo de gelo marinho	70

6	Construindo o SWAN	71
7	Construindo os pesos entre grades com o SCRIP	73
7.1	Construindo os pesos	73
7.2	Executando seu projeto no COAWST	75
8	Trabalhos do LOA	77
	Agradecimentos	83
	Referências bibliográficas	85



Nota dos autores

Este guia foi desenvolvido para auxiliar usuários novos com a familiarização e utilização do sistema de modelagem regional acoplada (COAWST). A principal idéia é ensinar o leitor as etapas necessárias para utilizar o COAWST, desde a sua instalação, simulação de um caso teste e a configuração de um projeto, escolhido pelo próprio usuário. Para atingir este objetivo, utilizamos diversas linguagens de programação, como Fortran, Python e MATLAB. Futuramente pretendemos adaptar todas as rotinas (scripts) para linguagem computacionais que sejam livres (gratuitas).

Quando começamos a escrever nosso guia, sentimos a necessidade de expandi-lo em seu escopo e incluir o passo a passo para a utilização do Coupled Ocean-Atmosphere-Wave-Sediment Transport Modeling System. Queríamos passar para o leitor a nossa experiência em utilizar o sistema de modelagem numérica considerado o estado da arte na área, de modo que fosse possível, ao longo da leitura, entender o funcionamento e a aplicação dele, unindo a teoria com a prática.

Porém uma grande dificuldade neste processo é a geração das condições de contorno e inicial do modelo regional oceanico, o Regional Ocean Modeling System, que depende de softwares pagos. Para contornar este problemas escolhemos trabalhar com o pacote *model2roms*. Este conjunto de rotinas foi desenvolvido em linguagem Python por Trond Kristiansen (<http://www.trondkristiansen.com/>).

Cabe também destacar, que em alguns capítulos o leitor irá encontrará como utilizar o COAWST em um cluster com arquitetura que esta disponível para o uso do Laboratório de Estudos do Oceano e da Atmosfera (LOA) da Coordenação Geral de Observação da Terra (OBT) do Instituto Nacional de Pesquisas Espaciais (INPE). Este é um sistema de computação de alta performance que permite o paralelismo de operações numéricas. Neste caso, o guia poderá servir somente como inspiração e valerá o conhecimento prévio do leitor em aplicar o COAWST no seu próprio sistema computacional. Isto implicará em conhecimento prévio do usuário em relação a sistema operacional (UNIX), compiladores fortran e C, além de bibliotecas de leitura de dados (por exemplo NetCDF), que deverão estar instaladas *a priori* no sistema computacional a ser utilizado.

Desejamos uma boa leitura e sucesso na sua pesquisa.

Os autores.



1. Introdução

1.1 O Regional Ocean Modeling System

O Regional Ocean Modeling System (ROMS; [Shchepetkin & McWilliams, 2005](#)) é um modelo oceânico tridimensional de superfície livre, coordenada vertical sigma (que segue o terreno) e resolve equações primitivas. Este modelo utiliza a média de Reynolds e o método de diferenças finitas para resolver as equações de Navier-Stokes assumindo aproximações hidrostáticas e de Boussinesq ([Haidvogel et al., 2008](#)).

As equações hidrostáticas de momentum utilizam um esquema de passo de tempo *split-explicit*, aonde os modos barotrópico e baroclínico são resolvidos separadamente em distintos números finitos de passos de tempo para resolver as equações de superfície livre e momentum integrado na vertical. A estrutura de passos de tempo separados mantém a conservação de volume e a preservação de consistência que são necessárias para as equações de traçadores ([Haidvogel et al., 2008](#); [Shchepetkin & McWilliams, 2005](#)).

A partir da grade, o modelo resolve as equações na horizontal através de coordenadas curvilíneas ortogonais do tipo Arakawa-C ([Arakawa & Lamb, 1977](#)). Na vertical, as coordenadas seguem as feições do terreno e permitem ajustar a resolução ao longo da coluna d'água. Para garantir a conservação de momentum, a grade utiliza diferenças finitas de segunda ordem ([Haidvogel et al., 2008](#)).

Segundo [Gouveia \(2015\)](#), o ROMS está estruturado por diversos modulos escritos em linguagem em Fortran (.F). Além deles também tem os arquivos de entrada (.in) que contém as parametrizações numéricas do modelo, arquivos de cabeçalho (.h) que fornecem informações de pré-processamento, arquivos com definições de variáveis (varinfo.dat) e pelo makefile usado para compilar os executáveis.

O ROMS é um modelo que possui códigos livre e seu desenvolvimento conta com a contribuição da comunidade de usuários. Atualmente, a versão utilizado no COAWST é gerenciado pelo Dr Hernan Arango da Rutgers University. Para acessar ao código do modelo, é necessário fazer o cadastro no site do ROMS (<https://www.myroms.org/>) para ter acesso ao código fonte do modelo. É necessário apresentar uma justificativa para se cadastrar no site. O site também conta com um fórum extremamente útil e bastante ativo (<https://www.myroms.org/forum/>) para perguntas e sugestões sobre problemas na utilização deste modelo.

1.2 O Weather Research & Forecasting Model

O Weather Research and Forecasting (WRF; [Skamarock et al., 2008](#)) é um modelo desenvolvido pelo National Centers for Environmental Prediction (NCEP), National Center for Atmospheric Research (NCAR) e grupos de pesquisa de diferentes universidades (<https://www.mmm.ucar.edu/weather-research-and-forecasting-model>).

Para integrar no tempo as equações governantes, o ARW utiliza modos de baixa freqüência que são integrados utilizando o esquema de Runge-Kutta de terceira ordem, e os modos acústicos (de alta frequência) integrados com passo de tempo menor, para manter a estabilidade numérica, através de um esquema “forward-backward” para os modos acústicos que se propagam horizontalmente, e de um esquema implícito para modos acústicos de propagação vertical e oscilações de empuxo ([Skamarock et al., 2008](#)).

O modelo WRF utiliza uma grade do tipo Arakawa-C ([Arakawa & Lamb, 1977](#)). Nesta grade as velocidades normais estão escalonadas a meio comprimento da grade das variáveis termodinâmicas, conforme visualizado na Figura 1.1.

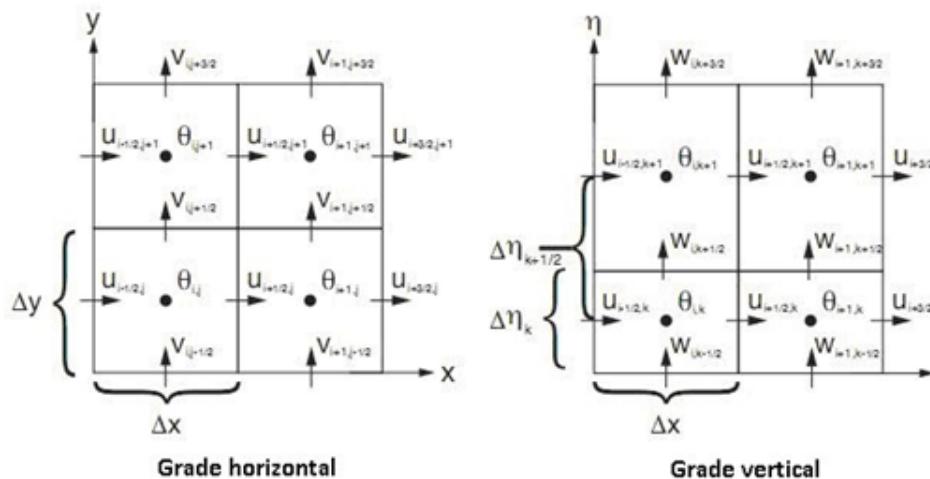


Figura 1.1. Grade horizontal e vertical do Weather Research and Forecast (WRF) em Arakawa-C.

Fonte: [Skamarock et al. \(2008\)](#).

É importante ressaltar que o WRF, sem o acoplamento com outros modelos, simula a rugosidade da superfície baseada na Relação de rugosidade com o estresse do vento proposta por [Charnock \(1955\)](#), como exemplificado na Equação 1.1:

$$Z_0 = Z_{ch} \frac{u_*^2}{g} \quad (1.1)$$

Onde Z_0 é a rugosidade, Z_{ch} é o parâmetro de Charnock (um valor adimensional de 0,018), U_* a velocidade de fricção (m/s) e g a aceleração da gravidade ($9,81 \text{ m/s}^2$).

1.3 O Simulating Waves Nearshore

O Simulating Waves Nearshore (SWAN; Booij et al., 1996, 1999) é um modelo de terceira geração, concebido para simular ondas em águas rasas. Ele é fundamentado na equação espectral discreta do balanço da ação da onda. É amplamente utilizado na previsão numérica do espectro de ondas em regiões costeiras, estuários, canais e outros, podendo utilizar campos de vento, batimetria e correntes fornecidos por outros modelos (Booij et al., 1996, 1999).

Silva (2013) e Booij et al. (1996; Booij et al., 1999) elencam as principais características do SWAN:

- Refração de onda com profundidade variável;
- Empinamento induzido pela profundidade e corrente;
- Geração de ondas pelo vento;
- dissipação por *whitecapping*;
- Dissipação pela quebra de ondas induzida pela profundidade;
- Dissipação devido à fricção com o fundo;
- Interações não lineares de 3 e 4 ondas;
- Difração.

1.4 O Model Coupling Toolkit

O Model Coupling Toolkit (MCT; Jacob et al., 2005; Larson et al., 2005; Warner et al., 2008) é um conjunto de rotinas livres, escritas basicamente em Fortran 90 que são utilizadas no acoplamento de modelos. Ele é uma ferramenta muito poderosa pois permite que os modelos sejam acoplados também de forma paralela.

Segundo o site do MCT (<http://www.mcs.anl.gov/research/projects/mct/>), ele fornece os seguintes serviços de acoplamento de núcleo:

- Um registro das componentes dos modelos;
- Descritores de decomposição do domínio;
- Ferramentas paralelizadas para interpolação *intergrid*;
- Ferramentas para mesclar dados de entre vários componentes;
- Um modelo de programação semelhante ao MPI (Message Passing Interface).

1.5 O Spherical Coordinate Remapping Interpolation Package

O Spherical Coordinate Remapping Interpolation Package (SCRIP; Jones, 1998, 1999) está disponível gratuitamente em <http://oceans11.lanl.gov/trac/SCRIP> e é distribuído junto com o COAWST. Este pacote é usado para projetos que utilizam mais de um modelo e que possuem grades horizontais diferentes, ou seja com diferentes resoluções espaciais. O SCRIP gerará os pesos de interpolação que são usados para remapear os dados entre as distintas grades dos distintos modelos.

No COAWST, o SCRIP foi modificado para gerar um único arquivo (no formato NetCDF) que contém o resultado do cálculo dos pesos baseado nas grades dos modelos.

1.6 O Coupled-Ocean-Atmosphere-Wave-Sediment Transport Modeling System

O Coupled Ocean-Atmosphere-Wave-Sediment Transport Modeling System (COAWST; [Warner et al., 2010, 2008](#)), é composto pelo modelo atmosférico WRF, o modelo oceânico ROMS, o modelo de ondas SWAN e o modelo transporte de sedimentos da Community Sediment Transport Modeling Project (CSTM; [Warner et al., 2008](#)). A troca das informações entre os modelos é realizada através do MCT ([Warner et al., 2010, 2008](#)). A frequência com que estas informações são trocadas é ajustada pelo usuário.

O acoplamento entre os modelos permite a possibilidade de resultados acurados nas simulações de fenômenos de interação entre os meios oceânico e atmosférico, permitindo identificar a evolução dos processos que afetam as regiões costeiras e como alteram as condições na costa ([Miller et al., 2018; Pullen et al., 2018](#)).

ATENÇÃO

Este guia não usa o CSTM. Em caso de interesse, há um estudo prévio sobre a transferência de sedimentos durante o furacão Isabel (2003) realizado por [Warner et al. \(2010\)](#).

Conforme a Figura 1.2, as informações trocadas entre modelos são:

- WRF -> ROMS: Estresse de superfície e fluxo de calor líquido (calculado no ROMS a partir das componentes dos fluxos de calor latente e sensível e radiação de ondas curta e longa);
- ROMS -> WRF: Temperatura da superfície do mar;
- SWAN -> ROMS: Direção da onda em superfície e no fundo, altura, comprimento, período, quebra percentual, dissipação de energia e velocidade orbital inferior;
- ROMS -> SWAN: Batimetria, elevação da superfície, altura da superfície do mar e correntes médias em profundidade;
- SWAN -> WRF: Rugosidade da superfície do mar (calculado no WRF a partir da altura significativa da onda, comprimento e período);
- WRF -> SWAN: Vento a 10m.

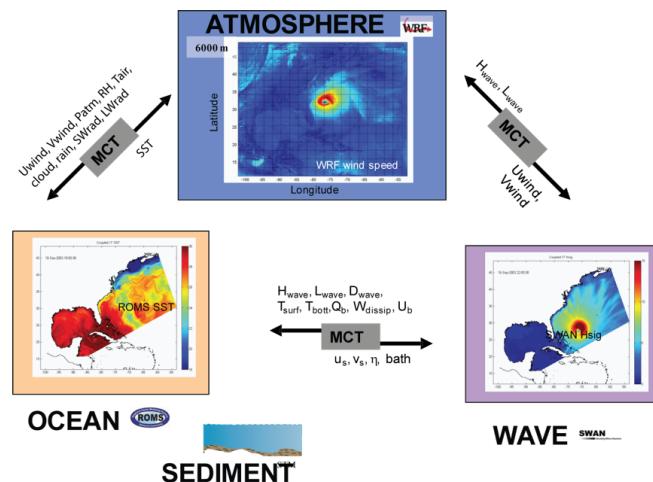


Figura 1.2. Esquema que detalha a troca de informações entre os modelos que compõem o COAWST.

Fonte: [Warner \(2018\)](#).

A página do Woods Hole Coastal and Marine Science Center fornece, em caráter experimental, uma apresentação em tempo real da integração do COAWST (Temperatura da Superfície do Mar, Altura da Superfície do Mar, Altura Significativa de Ondas, Vetores de Corrente e Vento e dispersão de Sedimentos) para o leste dos Estados Unidos e Golfo do México. É possível acessar o conteúdo em <https://woodshole.er.usgs.gov/project-pages/cccp/public/COAWST.htm>.

1.7 O Python

O Python é uma linguagem de programação poderosa, projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Prioriza a legibilidade do código sobre a velocidade ou expressividade.

A linguagem preza pela simplicidade e eficácia e possui uma ampla comunidade, o que facilita muito ao usuário em caso de dúvidas, pois encontra-se facilmente através da internet um extenso acervo de bibliotecas e documentação.

O Python Brasil (<http://python.org.br>) oferece grande auxílio para iniciantes na linguagem, disponibilizando introduções (<http://python.org.br/introducao/>) e também um guia (<http://python.org.br/cientifico>) para usar o Python no meio científico.

1.8 Materiais necessários para o uso do guia

Este guia foi construído utilizando o elementary OS 0.4 Loki, baseado no Ubuntu 16.04 LTS. Esta versão conta com o gcc 5.4.0. Isso é importante pois versões diferentes podem gerar conflitos ao utilizar o pacote model2roms no próprio computador.

Este guia utiliza o COAWST v. 3.2. É importante manter a versão utilizada, pois versões diferentes podem apresentar problemas na compilação e integração.



2. O COAWST no cluster Kerana

2.1 Sobre o Kerana

A máquina CRAY XE, também chamada de Kerana, é um cluster com arquitetura massivamente paralela, contando com 84 nós de processamento e 2688 cores e está localizada nas dependências do CPTEC/INPE, em Cachoeira Paulista, São Paulo. Por contar com a habilidade de parallelizar operações através da interface Message Passing Interface (MPI), o cluster é ideal para usar modelos numéricos com alta resolução espacial e temporal.

2.2 Registrando uma conta de usuário

Para iniciar o processo de requerimento de uma conta de usuário no Kerana, é necessário que o computador nas dependências do INPE possua IP fixo. É requerido do seu orientador ou supervisor o envio de um email para o Suporte do INPE (suporte@dsr.inpe.br) informando o endereço MAC, *hostname* do computador e motivo da requisição. Caso o computador já possua um IP fixo atribuído, também informe para que ocorra a troca.

Com o IP fixado, é necessário que o seu supervisor ou orientador entre em contato com o *Helpdesk* do INPE (helpdesk@cptec.inpe.br) requisitando a abertura da conta no cluster Kerana. Será necessário completar o preenchimento de um formulário via email, como o exemplo a seguir:

- 1. Nome do solicitante:**
- 2. Local de trabalho:**
- 3. Endereço de IP do equipamento de origem:**
- 4. Nome e endereço de IP do equipamento de destino:** acesso-hpc.cptec.inpe.br
- 5. Serviço/Porta:** ssh/2000
- 6. Volume diário de transferência de dados:**
- 7. Período de uso:**
- 8. Propósito do uso:**

9. Autorização das chefias dos departamentos:
10. Ramal:

2.3 Acessando o cluster Kerana

O acesso será feito inteiramente pelo terminal do computador. Serão necessários dois comandos primários: um para acesso e manipulação de arquivos e pastas dentro do Kerana e outro para fazer *download* e *upload* de arquivos.

Para acessar e manipular arquivos e pastas, digite no terminal, substituindo *nome.sobrenome* pelo usuário fornecido pelo *Helpdesk*:

ATENÇÃO

A partir de agora, sempre que o guia mostrar o usuário *nome.sobrenome*, altere para o seu nome de usuário fornecido pelo *Helpdesk*.

```
1 ssh -Y nome.sobrenome@acesso-hpc.cptec.inpe.br -p 2000
```

Para fazer *download* e *uploads*, digite:

```
1 sftp -P2000 nome.sobrenome@acesso-hpc.cptec.inpe.br
```

ATENÇÃO

Não é possível fazer *download* e *upload* de vários arquivos ao mesmo tempo usando o *sftp*. Uma dica é compactar em um único arquivo *tar.gz* e depois descompactá-los.

Para adicionar arquivos do seu computador para o Kerana:

```
1 put arquivo.tar.gz
```

Para extrair arquivos do Kerana para o seu computador:

```
1 get arquivo.tar.gz
```

2.4 Repertório de arquivos

São necessários certos arquivos na área de cada usuário para facilitar a utilização do cluster . Você encontrará eles no diretório:

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

Para copiar os arquivos para sua área, digite:

```
1 cp -r /scratch/adriano.sutil/repositorio /scratch/nome.sobrenome
```

ATENÇÃO

A partir de agora este guia utilizará os arquivos que estão dentro deste repositório, portanto é essencial que eles estejam em sua área.

2.5 O ambiente do Kerana

É necessário ativar alguns módulos no cluster para compilar o COAWST. Neste caso, abra o arquivo chamado *.bashrc* que se encontra na raiz do seu usuário.

```
1 vim .bashrc
```

Adicione os seguintes campos no final do arquivo, alterando somente o *nome.sobrenome* para o seu nome de usuário:

```
1 module load java
2 module load netcdf
3
4 export PATH=/scratch/nome.sobrenome/repositorio/Softs/nedit/5.5:$PATH
5 export PATH=/scratch/nome.sobrenome/repositorio/Softs/bin:$PATH
6
7 export PHDF5=${HDF_DIR}
8 export WRFIO_NCD_LARGE_FILE_SUPPORT=1
9
10 export PATH=/home/luciano.pezzi/local/bin:$PATH
11 export JASPERINC=/home/luciano.pezzi/local/include
12 export JASPERLIB=/home/luciano.pezzi/local/lib
13 export LD_LIBRARY_PATH=/home/luciano.pezzi/local/lib:$LD_LIBRARY_PATH
```

Salve e digite no terminal:

```
1 source .bashrc
```

2.6 Baixando o COAWST

ATENÇÃO

O COAWST versão 3.2 já está dentro do repositório discutido na Seção 2.4.

Para baixar o COAWST, envie um email para o Dr. John Warner (jcwarner@usgs.gov), um dos idealizadores do sistema de modelagem regional acoplada.

Após ter o acesso liberado, com as credenciais de usuário e senha disponibilizadas pelo Dr. John Warner, digite no terminal o comando abaixo, alterando o *myusername* para o seu nome de usuário.

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

Adicione a pasta do COAWST na sua área de trabalho do Kerana através de *sftp*, conforme ensinado na Seção 2.3.

2.7 Automatizando a compilação do COAWST no Kerana

ATENÇÃO

Este guia utiliza o COAWST versão 3.2.

Para agilizar o processo, é possível automatizar alguns passos da compilação. Portanto, entre no seguinte diretório:

```
1 cd /scratch/nome.sobrenome/COAWST/WRF/arch
```

Abra o arquivo *Config_new.pl*

```
1 nedit Config_new.pl
```

Procure pelas linhas:

```
1 printf "\nEnter selection [%d-%d] : ",1,$opt ;
2 $response = <STDIN> ;
```

E substitua o <STDIN> por 42, como no exemplo abaixo:

```
1 printf "\nEnter selection [%d-%d] : ",1,$opt ;
2 $response = 42 ;
```

Esta modificação selecionará, na compilação, as configurações da Kerana (*CRAY CCE (ftn/gcc): Cray XE and XC (dmpar)*), entre as várias disponíveis para usar o COAWST, como na Figura 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

Figura 2.1. Opções computacionais disponíveis para seleção na compilação do COAWST.

Agora, no mesmo arquivo, procure pelas seguintes linhas:

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

E modifique o <STDIN> para o modo básico de aninhamento do modelo atmosférico WRF, como o exemplo a seguir:

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

2.8 Compilando o MCT

ATENÇÃO Este guia utiliza o COAWST v 3.2.

Deve-se compilar o MCT antes de compilar o COAWST. Neste caso, é necessário utilizar o arquivo *setup_pgi.sh*. Este arquivo foi copiado do repositório anterior, sendo indispensável alterar os diretórios contidos nele. Portanto:

```
1 nedit setup_pgi.sh
```

Caso necessário, altere os diretórios de acordo com o nome da sua pasta do COAWST e ative o arquivo para carregar os módulos necessários:

```
1 source setup_pgi.sh
```

Serão carregadas na tela as bibliotecas, conforme apresentado na Figura 2.2:

```
Currently Loaded Modulefiles:
 1) modules/3.2.6.7
 2) nodestat/2.2-1.0400.29866.4.3.gem
 3) sdb/1.0-1.0400.31073.9.3.gem
 4) MySQL/5.0.64-1.0000.4667.20.1
 5) lustre-crash_gem_s/1.8.4_2.6.32.45_0.3.2_1.0400.6336.8.1-1.0400.30879.1.81
 6) udreg/2.3.1-1.0400.3911.5.13.gem
 7) ugni/2.3-1.0400.4127.5.20.gem
 8) gni-headers/2.1-1.0400.4156.6.1.gem
 9) dmapp/3.2.1-1.0400.3965.10.63.gem
10) xpmem/0.1-2.0400.30792.5.6.gem
11) hss-llm/6.0.0
12) Base-opts/1.0.2-1.0400.29823.8.5.gem
13) xtpe-network-gemini
14) cce/8.0.6
15) totalview-support/1.1.3
16) xt-totalview/8.10.0
17) acml/5.1.0
18) xt-libsci/11.1.01
19) pm1/3.0.1-1.0000.9101.2.26.gem
20) rca/1.0.0-2.0400.30002.5.75.gem
21) xt-asyncpe/5.14
22) atp/1.5.1
23) PrgEnv-cray/4.0.36
24) pbs/10.4.0.101257
25) xt-mpich2/5.5.4
26) xtpe-interlagos
27) java/jdk1.7.0_07
28) grads/2.0.a8

Currently Loaded Modulefiles:
 1) modules/3.2.6.7
 2) nodestat/2.2-1.0400.29866.4.3.gem
 3) sdb/1.0-1.0400.31073.9.3.gem
 4) MySQL/5.0.64-1.0000.4667.20.1
 5) lustre-crash_gem_s/1.8.4_2.6.32.45_0.3.2_1.0400.6336.8.1-1.0400.30879.1.81
 6) udreg/2.3.1-1.0400.3911.5.13.gem
 7) ugni/2.3-1.0400.4127.5.20.gem
 8) gni-headers/2.1-1.0400.4156.6.1.gem
 9) dmapp/3.2.1-1.0400.3965.10.63.gem
10) xpmem/0.1-2.0400.30792.5.6.gem
11) hss-llm/6.0.0
12) Base-opts/1.0.2-1.0400.29823.8.5.gem
13) xtpe-network-gemini
14) PrgEnv-pgi/4.0.36
15) xt-mpich2/5.5.4
16) atp/1.5.1
17) xt-asyncpe/5.14
18) pm1/3.0.1-1.0000.9101.2.26.gem
19) xt-libsci/11.1.01
20) xt-totalview/8.10.0
21) totalview-support/1.1.3
22) pgi/12.8.0
23) pbs/10.4.0.101257
24) xtpe-interlagos
25) java/jdk1.7.0_07
26) grads/2.0.a8
27) hdf5-parallel/1.8.8
28) netcdf-hdf5parallel/4.2.0
29) libfast/1.0.9

adriano.sutil@login1:~/COAWST_WR> ■
```

Figura 2.2. Módulos ativados no cluster com o arquivo *setup_pgi.sh* para o usuário adriano.sutil.

Vá até a pasta do MCT:

```
1 /home/nome.sobrenome/COAWST/Lib/MCT
```

Compile o MCT com os seguintes comandos:

```
1 ./configure --prefix=/scratch/nome.sobrenome/COAWST/Lib/MCT/pgi
2 make
3 make install
```

Observe as mensagens que aparecem no terminal e busque por erros. Caso negativo, o MCT foi compilado com sucesso.

2.9 Compilando o caso-teste Sandy

Existem dentro do COAWST alguns casos-teste para serem compilados e trabalhados. Neste caso compilaremos o projeto do furacão Sandy, que acopla e aninha o WRF, ROMS e SWAN. Primeiramente, é necessário conhecer a estrutura das pastas e arquivos do COAWST.

A estrutura típica de diretórios do COAWST está exemplificada na Figura 2.3:

```

drwxr-xr-x 20 adriano.sutil users   4096 Ago  3 13:04 .
drwxrwx--- 10 adriano.sutil users  4096 Ago  4 13:59 ..
drwxr-xr-x  6 adriano.sutil users  4096 Ago  3 15:25 ARkpost
drwxr-xr-x  2 adriano.sutil users 36864 Jul 26 16:28 Build
-rwrxr-xr-x  1 adriano.sutil users 18212 Jul 14 12:37 coawst.bash
-rwrxr-xr-x  1 adriano.sutil users 61866264 Jul 26 16:28 coawstM
-rwrxr-xr-x  1 adriano.sutil users 1163942 Jul 26 16:28 coawst.pgi.wrs
-rwrxr-xr-x  1 adriano.sutil users 2126848 Jul 14 10:30 COAWST_User_Manual.doc
drwxr-xr-x  2 adriano.sutil users  4096 Jul 14 10:30 Compilers
drwxr-xr-x  3 adriano.sutil users  4096 Jul 14 10:30 Data
-rwrxr-xr-x  1 adriano.sutil users 120 Jul 14 10:24 .._DS_Store
drwxr-xr-x  1 adriano.sutil users 12292 Jul 14 10:24 .DS_Store
-rwrxr-xr-x  1 adriano.sutil users 261 Jul 14 10:30 GENPARM.TBL
drwxr-xr-x  8 adriano.sutil users  4096 Jul 14 10:30 InWave
-rwrxr-xr-x  1 adriano.sutil users 29820 Jul 14 10:30 LANDUSE.TBL
drwxr-xr-x  5 adriano.sutil users  4096 Jul 14 10:31 Lib
-rwrxr-xr-x  1 adriano.sutil users 23983 Jul 14 10:32 makefile
drwxr-xr-x  2 adriano.sutil users  4096 Jul 14 10:32 Master
drwxr-xr-x  3 adriano.sutil users  4096 Jul 26 15:37 Projects
-rwrxr-xr-x  1 adriano.sutil users 3018 Jul 14 10:42 README.txt
drwxr-xr-x  2 adriano.sutil users  4096 Jul 14 10:32 REFDIF
drwxr-xr-x 16 adriano.sutil users  4096 Jul 14 10:33 ROMS
-rwrxr-xr-x  1 adriano.sutil users 749248 Jul 14 10:33 RRTM_DATA
drwxr-xr-x  1 adriano.sutil users 1498368 Jul 14 10:33 RRTM_DATA_DBL
-rwrxr-xr-x  1 adriano.sutil users 500 Jul 14 10:33 run_coawst
drwxr-xr-x  3 adriano.sutil users  4096 Jul 14 10:33 SeaIce
-rwrxr-xr-x  1 adriano.sutil users 739 Jul 14 10:40 setup_cray.sh
-rwrxr-xr-x  1 adriano.sutil users 748 Jul 14 10:40 setup_pgi.sh
-rwrxr-xr-x  1 adriano.sutil users 4419 Jul 14 10:33 SOILPARM.TBL
drwxr-xr-x  4 adriano.sutil users  4096 Jul 14 10:30 .svn
drwxr-xr-x  4 adriano.sutil users  4096 Jul 14 10:33 SWAN
drwxr-xr-x  3 adriano.sutil users  4096 Jul 14 10:34 Tools
-rwrxr-xr-x  1 adriano.sutil users 11190 Jul 14 10:34 URBPARM.TBL
drwxr-xr-x  5 adriano.sutil users  4096 Jul 14 10:34 User
-rwrxr-xr-x  1 adriano.sutil users 22717 Jul 14 10:34 VEGPARM.TBL
drwxr-xr-x  3 adriano.sutil users  4096 Jul 26 15:38 Work
drwxr-xr-x  7 adriano.sutil users 20480 Jul 26 17:29 WPS
drwxr-xr-x 17 adriano.sutil users  4096 Jul 26 15:50 WRF
adriano.sutil@adriano-login1:/COAWST_V3.2>

```

Figura 2.3. Representação da pasta principal do COAWST e as subpastas.

Serão utilizadas principalmente as pastas *Projects* e *Work*.

2.9.1 Pasta Projects

Na pasta *COAWST/Projects/Sandy* deverão constar os seguintes arquivos:

- 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
- Rweights.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 Pasta Work

Para facilitar o gerenciamento das simulações, é sugerida a criação da pasta *Work* no diretório principal do COAWST, com cada projeto inserido separadamente dentro dele. É nesta pasta que será simulado caso teste.

```
1 cd /scratch/nome.sobrenome/COAWST  
2 mkdir Work  
3 cd Work  
4 mkdir Sandy
```

Dentro da pasta */scratch/nome.sobrenome/COAWST/Work/Sandy* deverão conter os seguintes arquivos:

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

O arquivo *run_sandy.sh* é usado para iniciar a simulação, *link.sh* gera links simbólicos na pasta *Work*, que serão utilizados pelos modelos, e o *limpa.sh* é utilizado para limpar a pasta caso ocorra um erro e seja necessário iniciar uma nova integração. Os arquivos se encontram dentro da pasta */repositorio/work_coawst*.

Portanto:

```
1 cd /scratch/nome.sobrenome/repositorio/work_coawst  
2 cp limpa.sh link.sh run_sandy.sh /scratch/nome.sobrenome/COAWST/Work/Sandy
```

Vá até o diretório */scratch/nome.sobrenome/COAWST/Work/Sandy* e abra o arquivo *run_sandy.sh*:

```
1 cd /scratch/nome.sobrenome/COAWST/Work/Sandy  
2 nedit run_sandy.sh
```

Procure os comandos a seguir e modifique os diretórios de acordo com o seu nome de usuário:

```
1 PBS -o /scratch/nome.sobrenome/COAWST/Work/Sandy/rws_total.out  
2 ROOTDIR=/scratch/nome.sobrenome/COAWST
```

2.9.3 Compilando o caso teste

Vá para a pasta do projeto Sandy:

```
1 /home/nome.sobrenome/COAWST/Projects/Sandy
```

Abra os seguintes arquivos para realizar modificações nos próximos passos:

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

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

No arquivo *coupling_sandy.in* procure pela linha de comando abaixo e substitua:

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

Por:

```
1 OCN_name = /scratch/nome.sobrenome/COAWST/Projects/Sandy/ocean_sandy.in
```

Nos arquivos *swan_sandy.in* e *swan_sandy_ref3.in* complete todos os caminhos dos diretórios abaixo:

Modifique:

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

Por:

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

E no arquivo *ocean_sandy.in* procure pelas linhas de comando abaixo e substitua:

```

1 MyAppCPP = SANDY
2 VARNAME = ROMS/External/varinfo.dat
3 GRDNAME == Projects/Sandy/Sandy_roms_grid.nc \
    Projects/Sandy/Sandy_roms_grid_ref3.nc
4 ININAME == Projects/Sandy/Sandy_ini.nc \
    Projects/Sandy/Sandy_ini_ref3.nc
5 NGCNAME = Projects/Sandy/Sandy_roms_contact.nc
6 BRYNAME == Projects/Sandy/Sandy_bdy.nc
7 FRCNAME == Projects/Sandy/roms_narr_Oct2012.nc \
    Projects/Sandy/roms_narr_ref3_Oct2012.nc
10

```

Por:

```

1 MyAppCPP = Sandy
2 VARNAME = /scratch/nome.sobrenome/COAWST/ROMS/External/varinfo.dat
3 GRDNAME == /scratch/nome.sobrenome/COAWST/Projects/Sandy/Sandy_roms_grid.nc \
    /scratch/nome.sobrenome/COAWST/Projects/Sandy/Sandy_roms_grid_ref3.nc
4 ININAME == /scratch/nome.sobrenome/COAWST/Projects/Sandy/Sandy_ini.nc \
    /scratch/nome.sobrenome/COAWST/Projects/Sandy/Sandy_ini_ref3.nc
5 NGCNAME = /scratch/nome.sobrenome/COAWST/Projects/Sandy/Sandy_roms_contact.nc
6 BRYNAME == /scratch/nome.sobrenome/COAWST/Projects/Sandy/Sandy_bdy.nc
7 FRCNAME == /scratch/nome.sobrenome/COAWST/Projects/Sandy/roms_narr_Oct2012.nc \
    /scratch/nome.sobrenome/COAWST/Projects/Sandy/roms_narr_ref3_Oct2012.nc
10

```

Volte para a pasta principal do COAWST e abra o arquivo *coawst.bash*:

```

1 cd /scratch/nome.sobrenome/COAWST
2 nedit coawst.bash

```

Procure os seguintes comandos e modifique, se necessário:

```

1 export COAWST_APPLICATION=JOE_TC
2 export MY_ROOT_DIR=${HOME}/COAWST
3 export MY_HEADER_DIR=${MY_PROJECT_DIR}/Projects/JOE_TC

```

Por:

```

1 export COAWST_APPLICATION=Sandy
2 export MY_ROOT_DIR=${HOME}/COAWST
3 export MY_HEADER_DIR=${MY_PROJECT_DIR}/Projects/Sandy

```

Ative novamente os módulos no arquivo *setup_pgi.sh*, e depois compile o projeto com o comando:

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

Este comando criará o arquivo de texto *coawst.pgi.sandy* onde será possível acompanhar a evolução da compilação. Abra o arquivo de texto com o comando a seguir e procure pela mensagem final, como na Figura 2.4.

```
1 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/nct_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/runs_export.o: new IPA information
IPA: Recompiling ./Build/runs_import.o: new IPA information
```

Figura 2.4. Mensagem final após compilar o COAWST.

Caso queira acompanhar pelo terminal a evolução da compilação, utilize o comando:

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

ATENÇÃO O processo de compilação é longo!

Pronto! No diretório principal do COAWST, será criado um arquivo chamado *coawstM*. Neste arquivo estarão compiladas todas as informações do seu projeto. Agora com o COAWST compilado, iniciaremos o caso teste.

2.10 Simulando o caso teste Sandy

Para simular o caso, busque pela pasta *Work/Sandy*. Digite:

```
1 cd /scratch/nome.sobrenome/COAWST/Work/Sandy
2 nedit run_sandy.sh
```

ATENÇÃO A pasta *Work* deverá conter os arquivos *limpa.sh*, *link.sh* e *run_sandy.sh*. Eles poderão ser encontrados na pasta usada como repositório. Relembre a Seção 2.9.2.

Ao abrir o arquivo, verifique se os diretórios estão de acordo com o seu nome de usuário e digite o comando abaixo para iniciar a integração.

```
1 qsub run_sandy.sh
```

ATENÇÃO

O comando *qsub* submeterá o seu *job*. Ele enviará o script executado para um lote do cluster, reservando uma parte dos processadores para a sua simulação.

O processo gerará dois arquivos para acompanhar a evolução da simulação: o *log.out* e *log.err*. Para abrir, utilize:

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

Ou acompanhe diretamente no terminal com o comando:

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

As saídas das simulações serão armazenadas na pasta *Work/Sandy*. Caso algum erro ocorra, limpe a área de trabalho com o comando:

```
1 ./limpa.sh
```

ATENÇÃO

A simulação do caso teste Sandy poderá demorar 5 horas ao integrar usando 3 processadores.



3. Características do COAWST

Até aqui aprendemos como utilizar o cluster Kerana e, de maneira tangencial, como compilar e rodar um simples caso teste do COAWST. A partir de agora entraremos nas especificidades dos modelos, como por exemplo, alterar o número de processadores utilizados e a taxa de troca de informações entre eles.

3.1 Os arquivos estruturais dos modelos

Como você pode ter notado ao simular o caso Sandy, os modelos possuem arquivos gerenciais que auxiliam o usuário na definição do projeto:

O ROMS utiliza o arquivo *sandy.h* como arquivo que contém as opções de pré-processamento de C que definem o projeto. Acesse o site <https://www.myroms.org/wiki/cppdefs.h> para conhecer as definições disponíveis para o arquivo. O ROMS também utiliza o *ocean_sandy.in* como entrada padrão para executar o modelo. Este arquivo define as dimensões espaciais do projeto e parâmetros que não são informados durante a compilação, como por exemplo o passo de tempo, coeficientes e constantes físicas, configuração de coordenadas verticais, sinalizadores para controlar a frequência de saída, entre outros fatores. É possível aprender mais sobre este arquivo acessando o site <https://www.myroms.org/wiki/ocean.in>.

O WRF utiliza o arquivo *namelist.input* como arquivo para gerenciar as informações sobre o projeto, bem como os esquemas de parametrizações serão utilizados. Para aprender sobre a descrição do arquivo, visite https://esrl.noaa.gov/gsd/wrfportal/namelist_input_options.html. Para aprender sobre as opções físicas do modelo e as referências de cada um deles, acesse http://www2.mmm.ucar.edu/wrf/users/phys_references.html.

O SWAN utiliza o arquivo *swan_sandy.in* como gerenciador. Nele estão descritos diversos parâmetros, como a descrição do projeto, os dados de entrada, de grade e de condições de fronteira e iniciais, as parametrizações físicas de ondas, entre outros. Para saber mais sobre como configurar o arquivo, visite a guia *User Manual* no site <http://swanmodel.sourceforge.net/> e procure pela seção *Description of commands*.

3.2 Modificando o número de processadores

3.2.1 Modificando os processadores no ROMS

No ROMS, os processadores estão localizados no arquivo *ocean.in*, que fica dentro da pasta *Projects*, e se chamam *NtileI* e *NtileJ*. Um exemplo está na Figura 3.1. Neste caso o ROMS reservará 160 processadores para ser executado, pois $16 \times 10 = 160$.

```
! 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 == 10                                ! I-direction partition
NtileJ == 16                                ! J-direction partition
```

Figura 3.1. Representação do número de processadores usados no ROMS.

3.2.2 Modificando os processadores no WRF

Para alterar o número de processadores do WRF é necessário modificar o arquivo *namelist.input*. Nele existem as variáveis *nproc_x* e *nproc_y*, como na Figura 3.2. Neste caso serão reservados 320 processadores para o modelo atmosférico.

nproc_x	= 16,
nproc_y	= 20,

Figura 3.2. Representação do número de processadores para o WRF.

3.2.3 Modificando os processadores no SWAN

O arquivo *.in* do SWAN não discrimina os números de processadores usados, portanto basta alterar o número de processadores dele no *coupling.in*, como será mostrado na subseção a seguir.

3.2.4 Modificando os processadores no COAWST

Agora que foram modificados, individualmente, o número de processadores que serão utilizados pelos modelos, o acoplador precisa ser informado sobre esta quantidade de processadores. Essa informação é passada no arquivo *coupling.in*. Dentro dele deverão constar o número total de processadores a serem utilizados pelos modelos ROMS, WRF e SWAN, como na Figura 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

```

Figura 3.3. Representação do número de processadores para cada módulo do COAWST.

O $NnodesATM$ é referente ao total de processadores utilizados pelo WRF, o $NnodesWAV$ é o total do SWAN e o $NnodesOCN$ o do ROMS. Basta mudar de acordo com o total de processadores utilizados nos passos anteriores.

Por fim, é preciso modificar o total de processadores no *run.sh*, usado para submeter o experimento. Some o total de processadores usados pelos três modelos, seguindo a Equação 3.1:

$$TotalProc = NnodesATM + NnodesWAV + NnodesOCN \quad (3.1)$$

Onde $NnodesATM$ é o número de processadores utilizados no WRF, $NnodesWAV$ o número de processadores no SWAN, $NnodesOCN$ o total de processadores no ROMS e $TotalProc$ a soma de todos os processadores usados nos modelos.

Agora abra o arquivo *run.sh*, localizado dentro da pasta *Work*, e procure pelas linhas a seguir:

```

1 PBS -l mppwidth=3
2 aprun -n 3 coawstM ./coupling.in 1> log.out 2> log.err

```

Modifique o número 3 pelo número total de processadores utilizados.

3.3 Modificando o intervalo de tempo do acoplamento entre os modelos

Para modificar o intervalo de troca de informações entre os modelos, abra o arquivo *coupling.in* e modifique as variáveis $TI_ATM2WAV$, $TI_ATM2OCN$, $TI_WAV2ATM$, $TI_WAV2OCN$, $TI_OCN2WAV$, $TI_OCN2ATM$, como na Figura 3.4.

ATENÇÃO

A taxa de troca de informações deverá ser escolhida em segundos.

```
| 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
```

Figura 3.4. Intervalo de troca de informações entre os modelos utilizados no COAWST. Neste exemplo, a troca ocorrerá a cada 900 segundos.



4. Construindo o seu projeto no WRF

Agora que aprendemos como simular um caso teste e mudar a taxa de acoplamento e o número de processadores, iniciaremos o processo de criação de uma grade específica e das condições iniciais e de contorno (lateral) do WRF usando o NCEP Climate Forecast System Reanalysis (CFSR; [Saha et al., 2010](#)).

4.1 Compilando o WRF no Kerana

É recomendado copiar a pasta do WRF que já vem dentro do COAWST para a sua área de trabalho, a fim de evitar conflitos nas simulações, caso opte por integrar o WRF sem acoplamento e para utilizar o WRF Preprocessing System (WPS). Portanto:

```
1 cd /scratch/nome.sobrenome/COAWST  
2 cp -r WRF /scratch/nome.sobrenome
```

Ative os módulos do arquivo *setup_pgi.sh* com o comando:

```
1 source setup_pgi.sh
```

Entre na pasta cópia do WRF (*/home/nome.sobrenome/WRF*) e execute:

```
1 ./configure
```

ATENÇÃO

O próximo passo pode ser automatizado. Caso opte pela compilação automatizada, veja a Seção [2.7](#).

Caso tenha automatizado o processo de compilação, conforme apresentado na Seção 2.7, serão gerados os arquivos *real.exe*, *wrf.exe*, *tc.exe* e *ndown.exe*. Caso escolha a opção manual no cluster Kerana, procure pela opção *Cray XC CLE/Linux x86_64, Cray compiler with gcc (dmpar)*.

Na opção seguinte, aparecerá a seguinte mensagem:

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

Escolha a opção 1.

ATENÇÃO Fim do processo automatizado de compilação.

Digite:

```
1 compile em_real
```

Será gerado o *real.exe*, *wrf.exe*, *tc.exe* e *ndown.exe* no diretório */home/nome.sobrenome/WRF/test/em_real*.

4.2 O WRF Preprocessing System (WPS)

Para construir as condições iniciais e de contorno do WRF, usaremos WRF Preprocessing System (WPS). O WPS é um conjunto de três programas que preparam os dados de entrada para as simulações. Ele é composto pelos seguintes módulos:

- **geogrid**: Define o domínio do modelo e interpola os dados geográficos para a grade;
- **ungrib**: Extrai os campos meteorológicos de arquivos *.grib*;
- **metgrid**: Interpola os campos meteorológicos extraídos pelo ungrid para a grade do modelo definida pelo meogrid.

O WPS é controlado pelo arquivo de texto *namelist.wps* e é melhor esquematizado na Figura 4.1.

Para mais informações sobre o WPS, acesse: <http://www2.mmm.ucar.edu/wrf/OnLineTutorial/>.

Após criar estes arquivos, é utilizado o *real* para interpolar os campos meteorológicos para os níveis eta do WRF.

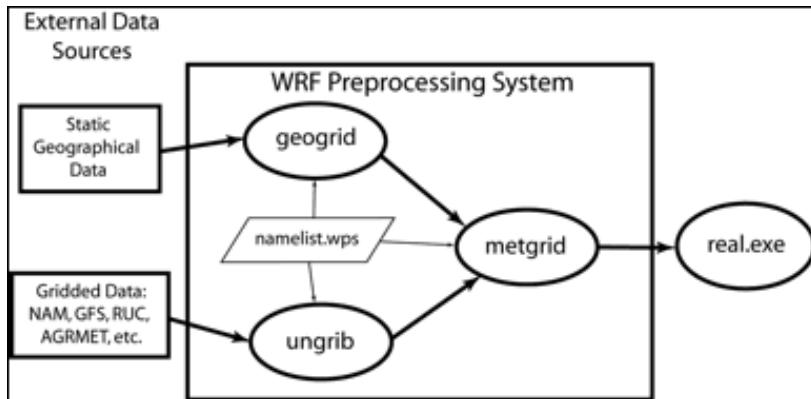


Figura 4.1. Construção de uma simulação real usando o WPS.

Fonte: Duda (2008)

4.2.1 Download do WPS

O WPS está incluído no pacote do COAWST, entretanto o *download* do WPS, pode ser feito na página do WRF (http://www2.mmm.ucar.edu/wrf/users/download/get_source.html). Baixe-o e adicione o arquivo *tar.gz* no cluster (de preferência dentro da pasta do COAWST) com o *sftp*:

```

1 sftp -P2000 nome.sobrenome@acesso-hpc.cptec.inpe.br
2 cd COAWST
3 put WPSV3.9.0.1.tar.gz

```

Para descompactar, saia do *sftp*, entre no cluster e digite:

```

1 ssh -Y nome.sobrenome@acesso-hpc.cptec.inpe.br -p 2000
2 cd COAWST
3 tar -xvzf WPSV3.9.0.1.tar.gz

```

4.2.2 Compilando o WPS no Kerana

ATENÇÃO

Como dito anteriormente, na Seção 4.2.2, o WPS já vem incluído junto com o COAWST. Sugere-se usar esta versão para gerar os arquivos de entrada do WRF.

Para compilar o WPS e gerar os executáveis do programa, é necessário ativar as bibliotecas com o *setup_pgi.sh*, que se encontra dentro do diretório */home/nome.sobrenome/repositorio*. Ative-os com o comando:

```
1 source setup_pgi.sh
```

Dentro da pasta do WPS, digite o seguinte comando:

```
1 ./configure
```

Será perguntado qual a máquina e o compilador que será utilizado. Escolha a opção *Cray XE/XC CLE/Linux x86_64, PGI compiler (serial)*. Neste caso, será gerada uma mensagem que o Fortran não é compatível com o C e com o NetCDF, porém ignore.

Abra o arquivo *configure.wps*:

```
1 nedit configure.wps
```

Modifique:

```
1 WRF_DIR  = ../WRFV3
2 SFC       = ftn
3 SCC       = gcc
4 DM_CC    = cc
5 DM_FC    = ftn
6 FFLAGS   = -N255 -f free -h byteswapi
7 F77FLAGS = -N255 -f fixed -h byteswapi
```

Por:

```
1 WRF_DIR  = /scratch/nome.sobrenome/WRF
2 SFC       = ftn
3 SCC       = gcc
4 DM_CC    = gcc
5 DM_FC    = ftn
6 FFLAGS   =
7 F77FLAGS =
```

Salve as modificações e inicie a compilação com o comando:

```
1 ./compile
```

Deverá gerar os executáveis *metgrid.exe* e *geogrid.exe*. Porém, é possível que o *ungrib.exe* não seja gerado. Neste caso, repita:

```
1 ./configure
```

Escolha a opção *Linux x86_64, PGI compiler (dmpar)*.

Abra novamente o *configure.wps* e modifique:

```
1 WRF_DIR = ../WRFV3
2 SFC      = pgf90
3 SCC      = pgcc
4 DMFC     = mpif90
5 DMCC     = mpicc
```

Por:

```
1 WRF_DIR = /home/nome.sobrenome/WRF
2 SFC      = ftn
3 SCC      = gcc
4 DMFC     = ftn
5 DMCC     = gcc
```

O instalador gerará somente o *ungrib.exe*

4.3 Construindo o seu projeto utilizando o CFSR

Uma opção de banco de dados para alimentar as simulações do modelo regional atmosférico é utilizar os dados do NCEP Climate Forecast System Reanalysis (CFSR) (<http://rda.ucar.edu/datasets/ds093.0>). Esse banco de dados é uma reanálise, de janeiro de 1979 a dezembro de 2010, alimentado através de dados orbitais, de cruzeiros de pesquisa, boias oceanográficas e estações meteorológicas e também através de modelagem numérica. Esta base de dados possui resolução vertical com 38 níveis, que se estendem desde a superfície até 1 hPa, com resolução temporal de 6 horas e horizontal de 0,5° para níveis de pressão e de 0,312° para variáveis na superfície.

Para acessar aos dados é necessário se cadastrar no site. Após feito isto, entre no link do banco de dados, clique na guia *Data Access* e clique em *Get a subset*.

Na próxima página, como observado na Figura 4.2, escolha as datas de início e fim dos dados e selecione em *Parameter presets* os três parâmetros do WRF (*Pressure*, *Surface* e *SST*). Só poderá ser selecionado um parâmetro por vez, logo, terás que baixar três vezes para ter os três parâmetros de entrada do modelo.

Figura 4.2. Exemplo da página de dados do CFSR.

Pegue os dados (em formato *grib*) e os separe em três pastas de acordo com os parâmetros de cada um: SST, Surface e Pressure. Compacte-os e coloque no cluster:

```

1 tar -cvzf SST.tar.gz SST/
2 tar -cvzf Pressure.tar.gz Pressure/
3 tar -cvzf Surface.tar.gz Surface/
4 ssh -Y nome.sobrenome@acesso-hpc.cptec.inpe.br -p 2000
5 mkdir Dados_CFSR
6 exit
7 sftp -P2000 nome.sobrenome@acesso-hpc.cptec.inpe.br
8 cd Dados_CFSR
9 put SST.tar.gz
10 put Pressure.tar.gz
11 put Surface.tar.gz
12 exit
13 ssh -Y nome.sobrenome@acesso-hpc.cptec.inpe.br -p 2000
14 cd Dados_CFSR
15 tar -xvzf SST.tar.gz
16 tar -xvzf Pressure.tar.gz
17 tar -xvzf Surface.tar.gz

```

ATENÇÃO

Você pode criar uma pasta de dados do CFSR na sua área. Não é preciso colocar os dados em uma pasta específica. No caso acima, os dados estão dentro da pasta Dados_CFSR.

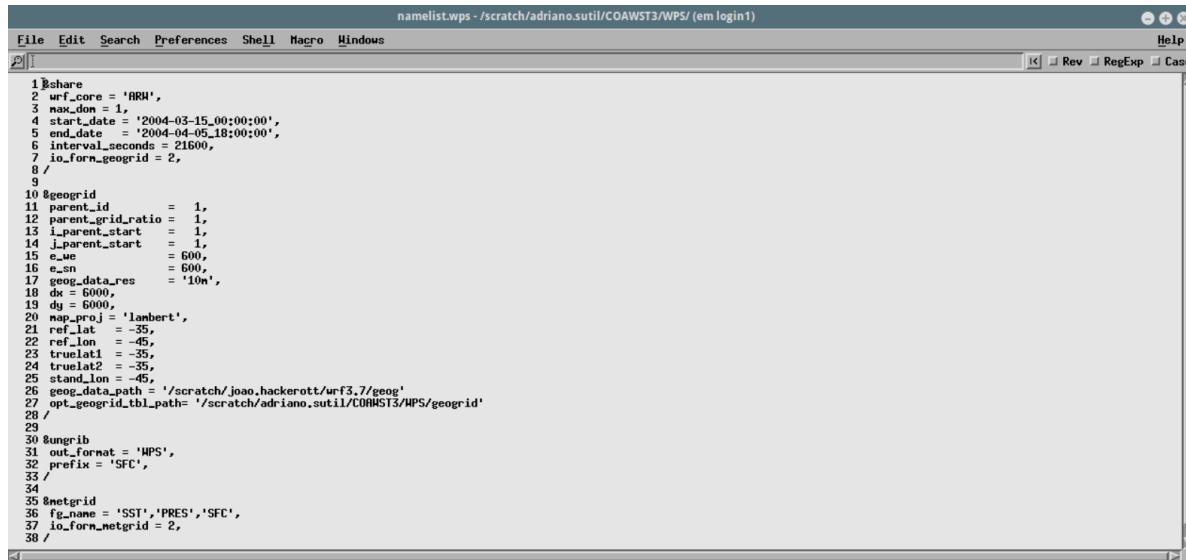
4.3.1 O geogrid

Como dito anteriormente, o geogrid gera o domínio da grade através de dados geográficos. Os dados podem ser obtidos nesse site: http://www2.mmm.ucar.edu/wrf/users/download/get_sources_wps_geog.html, ou então no repositório do guia a partir do diretório:

```
1 /scratch/nome.sobrenome/repositorio/COAWST/WPS/geog
```

Com os dados do CFSR e do geogrid em mãos, entre no diretório do WPS e abra o *namelist.wps*. A estrutura do arquivo está exemplificada na Figura 4.3:

```
1 nedit namelist.wps
```



```

namelist.wps - /scratch/adriano.sutil/COAWST3/WPS/ (em login1)
File Edit Search Preferences Shell Macro Windows Help
namelist.wps
1 &share
2 wrf_core = 'ARW',
3 max_dom = 1,
4 start_date = '2004-03-15_00:00:00',
5 end_date = '2004-04-05_18:00:00',
6 interval_seconds = 21600,
7 io_form_geogrid = 2,
8 /
9
10 &geogrid
11 parent_id = 1,
12 parent_grid_ratio = 1,
13 parent_start = 1,
14 j_parent_start = 1,
15 e_we = 600,
16 e_sn = 600,
17 geog_data_res = '10n',
18 dx = 6000,
19 dy = 6000,
20 map_proj = 'lambert',
21 nppt_lat = -35,
22 ref_lon = -45,
23 truelat1 = -35,
24 truelat2 = -35,
25 stand_lon = -45,
26 geog_data_path = '/scratch/joao.hackerott/wrf3.7/geog',
27 opt_geogrid_tbl_path= '/scratch/adriano.sutil/COAWST3/WPS/geogrid'
28 /
29
30 &ungrib
31 out_format = 'WPS',
32 prefix = 'SFC',
33 /
34
35 &metgrid
36 fg_name = 'SST','PRES','SFC',
37 io_form_metgrid = 2,
38 /

```

Figura 4.3. Exemplo do *namelist.wps*.

Altere os diretórios *geog_data_path*, para o diretório onde estão os dados geográficos do geog, o *geog_data_res* caso queira utilizar outros dados geográficos e o *opt_geogrid_tbl_path*, que é o diretório do próprio geogrid.

O WPS utiliza um ponto central e conta um ponto de grade de acordo com a resolução escolhida. No caso da Figura 4.3, será preparada uma simulação iniciando no dia 15 de março de 2004 até 05 de abril de 2004, com 6 km de resolução (*dx* e *dy* = 6000), em projeção lambertiana, com 600 pontos de grade.

Para iniciar o geogrid, é necessário ter o arquivo *.sh* submete a operação. Ele se encontra no diretório */repositorio/WPS_scripts*. Mova o conteúdo da pasta para o diretório do WPS com os comandos:

```
1 cd /home/nome.sobrenome/repositorio/WPS_scripts
2 mv qsub_geogrid.sh qsub_ungrib.sh qsub_metgrid.sh /home/nome.sobrenome/COAWST/WPS
```

A estrutura do arquivo *qsub_geogrid.sh* está demonstrada na Figura 4.4. Modifique o diretório de acordo com seu usuário.

```
#!/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*
```

Figura 4.4. Exemplo da estrutura do arquivo *qsub_geodrid.sh*.

Para executar o geogrid, utilize o comando *qsub*:

```
1 qsub qsub_geodrid.sh
```

Serão gerados dois arquivos para acompanhar a evolução do programa: *log.err* e *log.out*. Procure pela mensagem final de conclusão do programa, conforme a Figura 4.5, no arquivo *log.out*.

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

Figura 4.5. Exemplo de conclusão do geogrid.

Ao concluir a execução do geogrid, será gerado o arquivo *geo_em_d01.nc*. É possível visualizar o arquivo NetCDF com o Ncview:

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

Será sempre necessário executar o geogrid até encontrar um domínio que seja adequado para o seu projeto, pois o WPS não possui interface gráfica. Uma solução para contornar o problema é utilizar o NCAR Command Language versão 6.4 (NCL; [UCAR, 2017](#)) para plotar o domínio escolhido no *namelist.wps*. Este será o tema da próxima subseção.

4.3.2 Utilizando o NCL para plotar o domínio da grade

Para instalar o NCL na sua área, mova a pasta *NCL* que se encontra dentro do repositório:

```
1 mv /scratch/nome.sobrenome/repositorio/NCL /scratch/nome.sobrenome
```

Abra o *.bashrc*:

```
1 cd  
2 nedit .bashrc
```

Adicione as seguintes linhas de comando, alterando para o seu nome de usuário:

```
1 export NCARG_ROOT=/scratch/nome.sobrenome/NCL  
2 export PATH=$NCARG_ROOT/bin:$PATH
```

Salve e execute os comandos *source* e de inicialização do NCL.

```
1 source .bashrc  
2 ncl
```

Verifique se o NCL foi inicializado como na Figura 4.6.

```
Copyright (C) 1995-2015 - All Rights Reserved  
University Corporation for Atmospheric Research  
NCAR Command Language Version 6.3.0  
The use of this software is governed by a License Agreement.  
See http://www.ncl.ucar.edu/ for more details.  
ncl 0>  
ncl 1>
```

Figura 4.6. Exemplo de inicialização do NCL.

Entre no repositório e abra o arquivo *plotgrids.ncl*. O script gera a imagem do domínio a partir dos dados do arquivo *namelist.wps* do WPS.

```
1 cd /scratch/nome.sobrenome/repositorio  
2 nedit plotgrids.ncl
```

Modifique o diretório do *namelist.wps* de acordo com a sua área de usuário:

```
1 filename = "/home/nome.sobrenome/COAWST/WPS/namelist.wps"
```

Execute o script com o comando:

```
1 ncl plotgrids.ncl
```

O script buscará os pontos de grade e a resolução do domínio através do arquivo *namelist.wps* e mostrará na tela, como apresentado na Figura 4.7. Repita o processo até encontrar uma domínio de sua escolha.

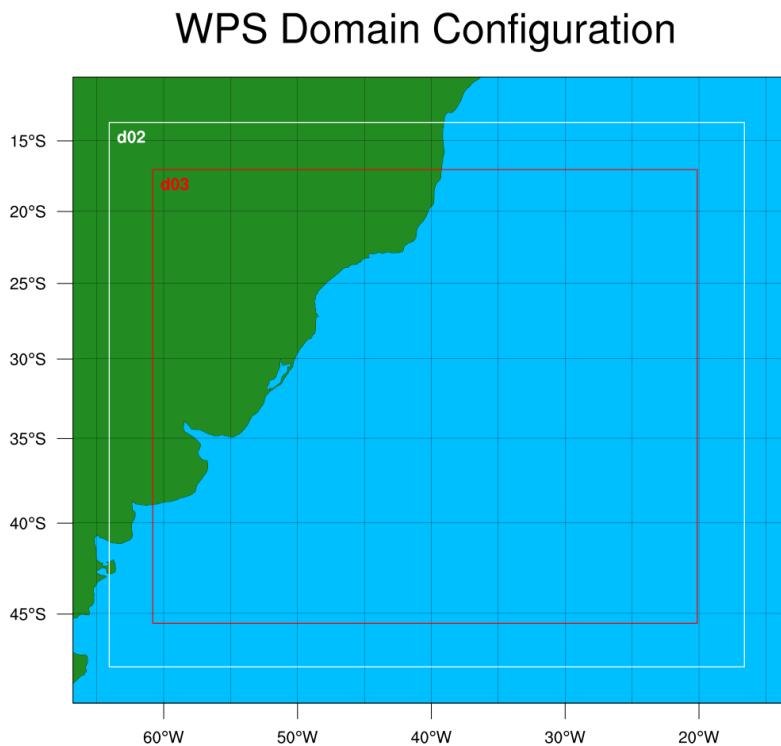


Figura 4.7. Exemplo da figura gerada pelo script *plotgrids.ncl*. Neste exemplo foram gerados três domínios.

Conforme a seção 4.3.1, para executar o geogrid, é necessário utilizar o comando *qsub*. Abra o arquivo *qsub_geogrid.sh* e mude os diretórios para o seu nome de usuário e salve. Digite o comando para executar o programa:

```
1 qsub qsub_geogrid.sh
```

Com a execução do geogrid, serão gerados dois arquivos para acompanhar a evolução do programa: *log.err* e *log.out*:

```
1 nedit log.err log.out &
```

Ao executar o geogrid com sucesso, uma mensagem final aparecerá no arquivo *log.out*, conforme a Figura 4.5. Será gerado o arquivo *geo_em_d01.nc*.

4.3.3 O ungrid

Com o domínio preparado pelo geogrid, podemos avançar para o ungrid. Abra o *namelist.wps* e na seção *&ungrid*, altere o *prefix* para *SST*, conforme a Figura 4.8 e salve.

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

Figura 4.8. Exemplo da seção *&ungrid* do *namelist.wps*.

Digite no terminal:

```
1 ln -sf ungrid/Variable_Tables/Vtable.SST Vtable
```

ATENÇÃO

A *Vtable* é utilizada para ler os arquivos em formato *grib*. É importante utilizar a *Vtable* correspondente para o banco de dados escolhido. Você pode consultá-las em *WPS/ungrid/Variable_Tables*

Copie os dados de SST do CFSR usando o arquivo *link_grib.csh* do WPS:

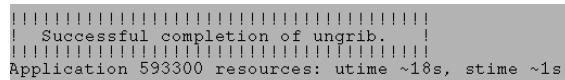
```
1 ./link_grib.csh /scratch/nome.sobrenome/Dados_CFSR/SST/*
```

Abra o arquivo *qsub_ungrib.sh* e modifique o diretório de acordo com o seu usuário e digite:

```
1 qsub qsub_ungrib.sh
```

ATENÇÃO O arquivo *qsub_ungrib.sh* se encontra dentro da pasta */repositorio/WPS_scripts*.

Serão criados diversos arquivos com inicial *SST*: seguido da data escolhida para a integração. Para checar o sucesso ao executar o *ungrib*, procure pela mensagem da Figura 4.9 ao final do arquivo *log.out*.



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

Figura 4.9. Mensagem final no arquivo *log.out* ao executar o *ungrib*.

Novamente altere o *prefix* no *namelist.wps* (Figura 4.8). Modifique *SST* por *SFC*. Salve a substituição e escreva no terminal:

```
1 ln -sf ungrib/Variable_Tables/Vtable.CFSR_sfc_flx06 Vtable
```

Crie os links simbólicos dos dados de superfície do CFSR com o comando abaixo e execute o *ungrib* mais uma vez:

```
1 ./link_grib.csh /scratch/nome.sobrenome/Dados_CFSR/Surface/*  
2 qsub qsub_ungrib.sh
```

Procure pela mensagem final conforme identificada na Figura 4.9.

Altere o *prefix* do *namelist.wps* (Figura 4.8). Substitua *SFC* por *PRES*, salve o arquivo e digite:

```
1 ln -sf ungrib/Variable_Tables/Vtable.CFSR_press_pgbh06 Vtable
```

Crie os links simbólicos para os dados de pressão e execute o *ungrib*:

```
1 ./link_grib.csh /scratch/nome.sobrenome/Dados_CFSR/Pressure/*  
2 qsub qsub_ungrib.sh
```

Por fim, procure pela mensagem final conforme identificada na Figura 4.9. Ao finalizar estes passos, constarão diversos arquivos com as iniciais *SST*, *PRES*, *SFC* seguidos com as datas do período de integração escolhidas.

4.3.4 O metgrid

Para interpolar os dados gerados pelo *ungrid.exe*, abra o *namelist.wps* e altere na guia *fg_name* os nomes utilizados no *ungrid*. No caso anterior, foram utilizados *PRES*, *SFC* e *SST*, como no exemplo da Figura 4.10:

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

Figura 4.10. Exemplo da guia do metgrid no *namelist.wps*.

Abra e mude o diretório do arquivo *qsub_metgrid.sh* e execute:

```
1 qsub qsub_metgrid.sh
```

Mova os arquivos gerados pelo *metgrid* (*met_em.d01**) para o diretório dos casos reais do WRF:

```
1 mv met_em.d01* /home/nome.sobrenome/WRF/test/em_real
```

4.3.5 O real

Usaremos o programa *real.exe* para gerar os arquivos finais para utilizar o WRF. Abra o arquivo *namelist.input*, que se encontra na pasta do WRF (*/home/nome.sobrenome/WRF/test/em_real*) e modifique de acordo com o seu domínio e tempo de simulação escolhidos no *namelist.wps*.

Para que o WRF faça a integração da TSM, é necessário incluir as seguintes variáveis ao final da seção *&time_control* (Figura 4.11):

```
1 io_form_auxinput4 = 2,
2 auxinput4_inname = "wrfflowinp_d<domain>",
3 auxinput4_interval = 360,
```

io_form_auxinput4 se refere ao formato final do arquivo *wrfflowinp* que será gerado pelo *real*, *auxinput4_inname* é o nome do arquivo de condição de fronteira para a TSM e *auxinput4_interval* é o intervalo de tempo, em minutos, do arquivo de fronteira.

```

&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                 = "wrfflowinp_d<domain>"
auxinput4_interval               = 360,
/

```

Figura 4.11. Exemplo da guia *&time_control* no *namelist.input*.

Inclua também, na seção *&physics*) (Figura 4.12 no *namelist.input* a opção para atualizar a TSM:

```

! 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,

```

Figura 4.12. Exemplo da guia *&physics* no *namelist.input* com *sst_update = 1* para atualizar a TSM ao longo do tempo.

Após concluir as modificações, é preciso utilizar o arquivo *qsub_real.sh* para submeter a operação. O arquivo está localizado no diretório */repositorio/WRF_scripts*. Mova o arquivo para a pasta *em_real* do WRF, mude o diretório de acordo com o seu usuário e execute:

```
1 cd /home/nome.sobrenome/repositorio/WRF_scripts
2 mv qsub_real.sh /home/nome.sobrenome/WRF/test/em_real
3 qsub qsub_real.sh
```

ATENÇÃO

O arquivo *qsub_real.sh* se encontra dentro da pasta */repositorio/WRF_scripts*.

Para atestar o sucesso em executar o real, procure no final do arquivo *log.out* pela mensagem de sucesso conforme exemplificada na Figura 4.13.

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

Figura 4.13. Exemplo da finalização com sucesso do real.

Ao completar o real, serão gerados três arquivos: *wrfbdy_d01*, *wrfinput_d01* e *wrflowinp_d01* caso tenha escolhido somente um domínio, isto é, sem aninhamento. Após isso, as condições do WRF estão prontas para serem copiadas para a sua pasta de projetos no COAWST:

```
1 cp wrfbdy_d01 wrfinput_d01 wrflowinp_d01 /home/nome.sobrenome/COAWST/Projects/nome_do_projeto
```

4.4 Usando o WRF

Com as condições do WRF prontas, é possível iniciar a simulação do seu projeto utilizando somente o WRF.

Para executar a simulação no cluster, é necessário utilizar o arquivo *qsub_wrf.sh*. Ele está localizado no diretório */repositorio/WRF_scripts*. Mova o arquivo para a pasta *em_real* do WRF, modifique o diretório de acordo com o seu usuário e salve:

```
1 cd /home/nome.sobrenome/repositorio/WRF_scripts
2 mv qsub_wrf.sh /home/nome.sobrenome/WRF/test/em_real
3 nedit qsub_wrf.sh
```

Para iniciar a simulação, execute:

```
1 qsub qsub_wrf.sh
```

ATENÇÃO

O arquivo *qsub_wrf.sh* se encontra dentro da pasta */repositorio/WRF_scripts*.

Acompanhe a evolução da simulação a partir dos arquivos *log.out* e *log.err*. Caso opte pela informação ser atualizada pelo terminal, digite:

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



5. Construindo o ROMS

5.1 Instalando as bibliotecas do Python no computador pessoal

Antes de gerar os arquivos de condições e a grade para o ROMS, é necessário instalar as bibliotecas e dependências. Nesta seção serão disponibilizados os endereços para baixá-las, bem como os comandos no terminal via *apt-get*.

Existem duas maneiras de instalar as bibliotecas necessárias: de modo manual, e através do Anaconda. Este último é uma plataforma gratuita e de código aberto para Python e R e possui um gerenciador de pacotes chamado Conda, que facilita a instalação de bibliotecas. Neste guia, apresentarei as duas opções. Pela praticidade, sugiro instalar usando o Conda (Seção 5.1.2), porém cuidado para conflito de versões, poise os repositórios do Conda são atualizados constantemente.

ATENÇÃO

A construção do ROMS deverá ser feita no seu próprio computador e não mais no Kerana, como feito anteriormente com o WRF.

5.1.1 Instalação manual

Abra o terminal, e crie uma pasta na sua *home* chamada *libsmodels*. Nela serão adicionados os arquivos baixados e as bibliotecas.

```
1 sudo mkdir libsmodels
```

Caso não tenha o *gfortran* e o *g++* instalados, instale via *apt-get*.

```
1 sudo apt-get install gfortran  
2 sudo apt-get install g++
```

A seguir, baixe o OpenMPI 2.0.1 (<https://www.open-mpi.org/software/ompi/v2.0/>), extraia o arquivo dentro da pasta *libsmodels* e digite no terminal:

```
1 export F77=gfortran
2 export FC=gfortran
3 export FC=gfortran
4 export CC=gcc
5 export CXX=g++
6 export FFLAGS="-m64 -fdefault-integer-8"
7 export FCFLAGS="-m64 -fdefault-integer-8"
8 export CFLAGS=-m64
9 export CXXFLAGS=-m64
```

Entre na pasta do OpenMPI e digite:

ATENÇÃO Altere o nome do computador abaixo (*usuario*) pelo nome do seu computador.

```
1 ./configure --prefix=/home/usuario/libsmodels
2 sudo make
3 sudo make check
4 sudo make install
```

Volte até a sua */home/usuario* e abra o arquivo oculto *.bashrc* através do comando *gedit .bashrc* ou com o editor de texto de sua preferência e nas linhas finais do arquivo escreva:

```
1 export PATH=/home/usuario/libsmodels/bin:$PATH
2 export LD_LIBRARY_PATH=/home/usuario/libsmodels/lib:$LD_LIBRARY_PATH
3 export PATH=/home/usuario/libsmodels/include:$PATH
```

Salve o arquivo e no terminal, digite o comando a seguir para atualizar o arquivo e aplicar as mudanças.

```
1 source .bashrc
```

Para testar se a instalação do OpenMPI está correta, digite no terminal:

```
1 ompi_info -a grep 'Fort integer size'
```

O resultado deverá ser:

```
1 Fort integer size: 8
```

Caso não tenha instalado no seu computador, será necessário instalar o *m4*, *make* e *perl*. Para checar se o seu computador já possui estes programas instalados:

```
1 which m4
2 which make
3 which perl
```

Caso a resposta seja negativa, instale a partir dos comandos a seguir:

```
1 sudo apt-get install m4
2 sudo apt-get install make
3 sudo apt-get install perl
```

Instale o *szip* 2.1 (<https://www.hdfgroup.org/HDF5/release/obtain5.html>). Baixe-o, descompacte na pasta */home/usuario/libsmodels* e entre, via terminal, na pasta do *szip* digite:

```
1 export F77=gfortran
2 export FC=gfortran
3 export CC=gcc
4 export CXX=g++
5 export FFLAGS="-m64 -fdefault-integer-8"
6 export FCFLAGS="-m64 -fdefault-integer-8"
7 export CFLAGS=-m64
8 export CXXFLAGS=-m64
9 ./configure --prefix=/home/usuario/libsmodels
10 sudo make
11 sudo make check
12 sudo make install
```

Instale o *zlib* 1.2.8 (<http://zlib.net/>). Baixe-o, descompacte na pasta */home/usuario/libsmodels* e entre, via terminal, na pasta do *zlib* digite:

```

1 export F77=gfortran
2 export FC=gfortran
3 export CC=gcc
4 export CXX=g++
5 export FFLAGS="-m64 -fdefault-integer-8"
6 export FCFLAGS="-m64 -fdefault-integer-8"
7 export CFLAGS=-m64
8 export CXXFLAGS=-m64
9 ./configure --prefix=/home/usuario/libmodels
10 sudo make
11 sudo make check
12 sudo make install

```

Instale agora o *Curl* 7.50.3 (<http://curl.haxx.se/download.html>). Baixe-o, descompacte na pasta */home/usuario/libmodels* e entre, via terminal, na pasta do *Curl* digite:

```

1 export F77=gfortran
2 export FC=gfortran
3 export CC=gcc
4 export CXX=g++
5 export FFLAGS="-m64 -fdefault-integer-8"
6 export FCFLAGS="-m64 -fdefault-integer-8"
7 export CFLAGS=-m64
8 export CXXFLAGS=-m64
9 ./configure --prefix=/home/usuario/libmodels/
10 sudo make
11 sudo make check
12 sudo make install

```

Instale as bibliotecas necessárias para usar o HDF5. Digite:

```

1 sudo apt-get install libgtk2.0-dev

```

Instale o *HDF5* 1.8.9 (<https://www.hdfgroup.org/HDF5/release/obtain5.html>). Baixe-o, descompacte na pasta */home/usuario/libmodels* e entre, via terminal, na pasta do *HDF5* digite:

```

1 export FC=gfortran
2 export CC=gcc
3 export CXX=g++
4 export LDFLAGS=-L/home/usuario/libmodels/lib
5 export CPPFLAGS=-I/home/usuario/libmodels/include

```

Na mesma linha de comando, digite:

```
1 ./configure --enable-fortran=yes --enable-fortran2003=yes --enable-cxx=yes  
2 --with-szlib=/home/usuario/libsmodeles --with-zlib=/home/usuario/libsmodeles --enable-hl  
3 --enable-shared --prefix=/home/usuario/libsmodeles
```

Complete a instalação do *HDF5* com os comandos abaixo:

```
1 sudo make  
2 sudo make check  
3 sudo make install
```

O próximo passo é instalar o *NetCDF-C 4.4.1* (<https://www.unidata.ucar.edu/downloads/netcdf/index.jsp>). Baixe-o, descompacte na pasta */home/usuario/libsmodeles* e na pasta do *NetCDF-C* digite:

```
1 export FC=gfortran  
2 export CC=gcc  
3 export CXX=g++  
4 export LDFLAGS=-L/home/usuario/libsmodeles/lib  
5 export CPPFLAGS=-I/home/usuario/libsmodeles/include
```

Na mesma linha de comando, digite:

```
1 ./configure --prefix=/home/usuario/libsmodeles  
2 --enable-netcdf4 --with-hdf5=/home/usuario/libsmodeles --enable-shared --enable-dap
```

Complete a instalação do *NetCDF-C* com os comandos abaixo:

```
1 sudo make  
2 sudo make check  
3 sudo make install
```

Instale agora o *NetCDF-Fortran 4.2* (<https://www.unidata.ucar.edu/downloads/netcdf/index.jsp>). Baixe-o, descompacte na pasta */home/usuario/libsmodeles* e na pasta do *NetCDF-Fortran* digite:

```
1 export FC=gfortran  
2 export F77=gfortran  
3 export CC=gcc  
4 export CXX=g++  
5 export LDFLAGS=-L/home/usuario/libsmodeles/lib  
6 export CPPFLAGS=-I/home/usuario/libsmodeles/include  
7 export LIBS='-lnetcdf -lhdf5 -lhdf5_hl -lz'
```

Na mesma linha de comando, digite:

```

1 ./configure --prefix=/home/usuario/libsmmodels --enable-netcdf4
2 --with-hdf5=/home/usuario/libsmmodels --enable-shared --enable-dap

```

Termine a instalação do *NetCDF-Fortran*:

```

1 sudo make
2 sudo make check
3 sudo make install

```

Instale o *JasPer 1.900.1* (<http://www.linuxfromscratch.org/blfs/view/svn/general/jasper.html>). Baixe-o, descompacte na pasta */home/usuario/libsmmodels* e na pasta do JasPer digite:

```

1 export FC=gfortran
2 export F77=gfortran
3 export CC=gcc
4 export CXX=g++
5 ./configure --prefix=/home/usuario/libsmmodels
6 sudo make
7 sudo make check
8 sudo make install

```

Instale o *libpng 1.6.24* (http://sourceforge.net/projects/libpng/?source=typ_redirect). Baixe-o, descompacte na pasta */home/usuario/libsmmodels* e na pasta do *libpng* digite:

```

1 export FC=gfortran
2 export F77=gfortran
3 export CC=gcc
4 export CXX=g++
5 export LDFLAGS=-L/home/usuario/libsmmodels/lib
6 export CPPFLAGS=-I/home/usuario/libsmmodels/include
7 export LIBS=' -lz'
8 ./configure --with-zlib=/home/usuario/libsmmodels --prefix=/home/usuario/libsmmodels
9 sudo make
10 sudo make check
11 sudo make install

```

Instale o *Udunits 2.1.24* (<ftp://ftp.unidata.ucar.edu/pub/udunits/>). Baixe-o, descompacte na pasta */home/usuario/libsmmodels* e entre na pasta do *Udunits* digite:

```

1 export FC=gfortran
2 export F77=gfortran
3 export CC=gcc
4 export CXX=g++
5 ./configure --prefix=/home/usuario/libsmmodels
6 sudo make
7 sudo make check
8 sudo make install

```

Instale o NetCDF4 para Python. Neste caso, baixe primeiramente o Git e use-o para baixar.

```
1 sudo apt-get install git
```

Clone o repositório do *NetCDF4-Python* com o comando:

```
1 git clone https://github.com/Unidata/netcdf4-python
```

Entre na pasta do NetCDF4-Python e modifique o arquivo *setup.cfg.template* e salve:

```
1 ncconfig      = /home/usuario/libmodels/bin/nc-config
2 netCDF4_dir   = /home/usuario/libmodels
3 HDF5_dir     = /home/usuario/libmodels
4 szip_dir     = /home/usuario/libmodels
5 jpeg_dir     = /home/usuario/libmodels
6 curl_dir     = /home/usuario/libmodels
```

Modifique o nome do arquivo *setup.cfg.template* para *setup.cfg*:

```
1 mv setup.cfg.template setup.cfg
```

Crie na pasta *libmodels* as subpastas *lib/python2.7/site-packages*:

```
1 mkdir lib
2 cd lib
3 mkdir python2.7
4 cd python2.7
5 mkdir site-packages
```

Volte para a pasta principal do *NetCDF4-Python* e compile:

```
1 python setup.py build
2 python setup.py install --prefix=/home/usuario/libmodels
```

Caso queira conferir se a instalação ocorreu com sucesso, tente importar a biblioteca no Python. Digite no terminal:

```

1 python
2 import netCDF4

```

Instale agora o *mpi4py* (<https://pypi.python.org/pypi/mpi4py>), extraia e digite:

```

1 python setup.py build
2 python setup.py install --prefix=/home/usuario/libsmodels

```

Para testar a instalação, digite:

```
1 mpiexec -n 5 python demo/helloworld.py
```

Instale agora o *ESMF* e *ESMPy* (<https://www.earthsystemcog.org/projects/esmpy/releases>). Primeiro digite os comandos a seguir para instalar os arquivos de bibliotecas necessárias:

```

1 sudo apt-get install python-setuptools
2 sudo easy_install pip
3 sudo pip install distribute
4 sudo pip install nose

```

Dentro da pasta do *ESMF*, digite no terminal:

```

1 export ESMF_F90COMPILER=gfortran
2 export ESMF_NETCDF_LIBS="-lncdf -lnetcdff"
3 export ESMF_DIR=/home/usuario/libsmodels/packages/esmp.ESMF_6_3_0rp1_ESMP_01/esmf
4 export ESMF_NETCDF="local"
5 export ESMF_COMPILER=gfortran
6 export ESMF_COMM=openmpi
7 export ESMF_TESTEXHAUSTIVE=on
8 export ESMF_TESTSHAREDOBJ=on
9 export ESMF_NETCDF_INCLUDE=/home/usuario/libsmodels/include
10 export ESMF_NETCDF_LIBPATH=/home/usuario/libsmodels/lib
11 export ESMF_INSTALL_PREFIX=/home/usuario/libsmodels/ESMF
12 make
13 make check
14 make all_tests
15 make install

```

Entre na pasta *scr/addon/ESMPy*:

```
1 cd src/addon/ESMPy/
```

Na mesma linha de comando, digite:

```
1 python setup.py build  
2 --ESMFMKFILE=/home/usuario/libmodels/ESMF/lib/lib0/Linux.gfortran.64.openmpi.default/esmf.mk
```

Complete a instalação:

```
1 python setup.py install --prefix=/home/usuario/libmodels  
2 python setup.py test
```

Instale o Octant, utilizando o Git:

```
1 git clone https://github.com/hetland/octant
```

Na pasta *octant/external*, exclua tudo e baixe os seguintes arquivos:

```
1 rm -rf *  
2 git clone https://github.com/sakov/gridgen-c.git  
3 git clone https://github.com/sakov/gridutils-c  
4 git clone https://github.com/sakov/csa-c  
5 git clone https://github.com/sakov/nm-c
```

E renomeie as pastas baixadas para, respectivamente: *gridgen*, *gridutils*, *csa* e *nm*. Entre na pasta *nn/nn*, exporte as bibliotecas e instale o *nn*:

```
1 cd nn/nn/  
2 export FC=gfortran  
3 export F77=gfortran  
4 export CC=gcc  
5 export CXX=g++  
6 ./configure --prefix=/home/usuario/libmodels  
7 make  
8 make install
```

Instale o *csa* da mesma maneira que foi instalado o *nn*:

```

1 cd ../../csa/csa/
2 export FC=gfortran
3 export F77=gfortran
4 export CC=gcc
5 export CXX=g++
6 ./configure --prefix=/home/usuario/libmodels
7 make
8 make install

```

Ainda na pasta do *csa*, compie o arquivo *csa.o* para a pasta */home/usuario/libmodels/lib*:

```

1 cp csa.o /home/usuario/libmodels/lib/l

```

Instale o *gridutils*:

```

1 cd ../../gridutils/gridutils/
2 export FC=gfortran
3 export F77=gfortran
4 export CC=gcc
5 export CXX=g++
6 LDFLAGS=-L/home/usuario/libmodels/lib
7 ./configure --prefix=/home/usuario/libmodels CPPFLAGS=-I/home/usuario/libmodels/include
8 make
9 make install

```

Instale o *gridgen*:

```

1 cd ../../gridgen/
2 export FC=gfortran
3 export F77=gfortran
4 export CC=gcc
5 export CXX=g++
6 LDFLAGS=-L/home/usuario/libmodels/lib
7 ./configure -prefix=/home/usuario/libmodels CPPFLAGS=-I/home/usuario/libmodels/include

```

Abra o arquivo *makefile* e altere no arquivo de acordo com o que está abaixo:

```

1 CFLAGS = -g -O2 -Wall -I/home/usuario/libmodels/include
2 LDFLAGS = -L/home/usuario/libmodels/lib
3 LIBS = -lgu -lm -L/home/usuario/libmodels/lib

```

Salve o arquivo, feche e digite no terminal:

```
1 make  
2 make lib  
3 make shlib  
4 make install
```

Dentro da pasta `/home/usuario/libmodels/octant/octant`, modifique o arquivo `grid.py` como o exemplo abaixo (aproximadamente na linha 898).

ATENÇÃO

Observe atentamente para o espaço em branco antes do início do texto, pois poderá dar erro na hora de instalar se a contagem de linha estiver errada.

```
1 self._libgridgen = np.ctypeslib.load_library('libgridgen.so', '/home/usuario/libmodels/lib')  
2 print octant.__path__[0]  
3 self._libgridgen = np.ctypeslib.load_library('_gridgen', octant.__path__[0])
```

Salve e feche o arquivo. Digite agora:

```
1 python setup.py build --fcompiler=gfortran
```

Em `/home/usuario/libmodels/octant/octant`, abra o arquivo `csa.py` e modifique como abaixo, observando corretamente o espaçamento:

```
1 _csa = np.ctypeslib.load_library('csa.o', '/home/usuario/libmodels/libs')
```

Por fim, em `/home/usuario/libmodels/octant`, digite:

```
1 python setup.py install --prefix=/home/usuario/pythonmodels/octant
```

Adicione os diretórios no seu `.bashrc`. Lembrando novamente que o `.bashrc` encontra-se na sua `/home/usuario`.

```
1 export PYTHONPATH=$PYTHONPATH:/home/usuario/libmodels/octant  
2 export PYTHONPATH=$PYTHONPATH:/home/usuario/libmodels/octant/include  
3 export PYTHONPATH=$PYTHONPATH:/home/usuario/libmodels/octant/lib/python2.7/site-packages  
4 export PYTHONPATH=$PYTHONPATH:/home/usuario/libmodels/octant/lib
```

Atualize o seu `.bashrc`. Digite no terminal:

```
1 source .bashrc
```

5.1.2 Utilizando o Conda

O Anaconda, é uma distribuição para Python e R que fornece suporte a várias bibliotecas e pacotes utilizados neste guia, facilitando a instalação. Para isso, baixe o Anaconda (<https://www.continuum.io/downloads>). Para instalar basta entrar no diretório onde está o arquivo de instalação, mudar as permissões de instalação do arquivo e seguir as instruções do instalador:

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

O Anaconda possui uma interface gráfica (GUI) para o Python, chamado Spyder. Para usar basta digitar no terminal:

```
1 spyder &
```

A seguir, digite os seguintes comandos no terminal para instalar as bibliotecas necessários para construir um projeto no ROMS usando o Python:

```
1 conda install -c conda-forge openmpi  
2 conda install -c anaconda gcc  
3 conda install -c mutirri szip  
4 conda install -c anaconda zlib  
5 conda install -c anaconda curl  
6 conda install -c anaconda hdf5  
7 conda install -c conda-forge netcdf-fortran  
8 conda install -c scitools jasper  
9 conda install -c anaconda libpng  
10 conda install -c conda-forge udunits  
11 conda install -c anaconda netcdf4  
12 conda install -c anaconda cython  
13 conda install -c anaconda numpy  
14 conda install -c anaconda scipy  
15 conda install -c anaconda mpi4py  
16 conda install -c conda-forge esmf
```

5.2 Instalando o Pyroms

Com as bibliotecas básicas já instaladas, nesta seção será mostrado como instalar o PYROMS. Ele é uma coleção de ferramentas para ajudar com arquivos de entrada e saída do ROMS. Foi originalmente iniciado por Rob Hetland como um projeto no GoogleCode e reescrito por Frederic Castruccio. Será seguido o mesmo esquema da seção anterior, com o endereço para baixar a biblioteca e os comandos que deverão ser digitados no terminal.

Baixe o PyROMS dentro do seu *libsmodels*:

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

Por fim, entre no diretório */home/usuario/libsmodels/pyroms/pyroms_toolbox*, digite.

```
1 export FC=gfortran
2 export F77=gfortran
3 export CC=gcc
4 export CXX=g++
5 python setup.py build --fcompiler=gfortran
6 python setup.py install --prefix=/home/usuario/libsmodels/pyroms
```

Instale o bathy_smoothen:

```
1 cd ../../bathy_smoothen/external/lp_solve_5.5/lp_solve/
2 sh ccc
3 cp bin/ux64/lp_solve /home/usuario/libsmodels/bin/
4 cd ../../lpsolve55
5 sh ccc
6 cp bin/ux64/* /home/usuario/libsmodels/lib/
```

Vá para o seguinte diretório e abra o *setup.py*:

```
1 cd ../../lp_solve_5.5/extra/Python/
2 gedit setup.py
```

Mude os caminhos descritos abaixo, no *setup.py*. Não se esqueça de observar as linhas em branco antes das linhas de comando.

```
1 include_dirs=['..','/usr/include','/home/usuario/libsmodels/include'],
2 library_dirs=['/home/usuario/libsmodels/pyroms/lib','/home/usuario/libsmodels/lib'],
```

Instale:

```
1 python setup.py build
2 python setup.py install --prefix=/home/usuario/libsmodels/pyroms
```

Vá para o diretório do bathy_smoothen, como exemplificado abaixo e digite no terminal:

```

1 cd ../../../../../../bathy_smoothener/
2 export FC=gfortran
3 export F77=gfortran
4 export CC=gcc
5 export CXX=g++
6 python setup.py build --fcompiler=gfortran
7 python setup.py install --prefix=/home/usuario/libsmodels/pyroms

```

Instale o Pyroms. Para isso use o comando para alterar o diretório:

```
1 cd ../pyroms/pyroms/
```

Abra o arquivo *hgrid.py* e mude na linha 1028 como abaixo:

```
1 self._libgridgen = np.ctypeslib.load_library('libgridgen.so', '/home/usuario/libsmodels/lib')
```

Além disso, adicione a seguinte biblioteca na linha 21:

```
1 from numpy import amin
```

Salve e feche o *hgrid.py* e digite:

```

1 cd ../
2 export FC=gfortran
3 export F77=gfortran
4 export CC=gcc
5 export CXX=g++
6 python setup.py build --fcompiler=gfortran
7 python setup.py install --prefix=/home/usuario/libsmodels/pyroms

```

Coloque os caminhos no seu *.bashrc*:

```

1 export PYTHONPATH=$PYTHONPATH:/home/usuario/libsmodels/pyroms/lib/python2.7/site-packages
2 export PYTHONPATH=$PYTHONPATH:/home/usuario/libsmodels/pyroms/lib/python2.7/site-packages
3 export PYTHONPATH=$PYTHONPATH:/home/usuario/libsmodels/pyroms/lib/python2.7/site-packages
4 export PYTHONPATH=$PYTHONPATH:/home/usuario/libsmodels/pyroms/lib
5 export PYTHONPATH=$PYTHONPATH:/home/usuario/libsmodels/pyroms/lib/python2.7/site-packages/pyroms_toolbox/Grid_HYCOM
6 export PYTHONPATH=$PYTHONPATH:/home/usuario/libsmodels/pyroms/examples

```

Atualize o *.bashrc*:

```
1 source .bashrc
```

Vá até `/home/usuario/libsmodels/pyroms/lib/python2.7/site-packages/bathy_smoothen/` e apague o arquivo `lpsolve55.so`.

Por fim, vá na pasta agora do `pyroms_toolbox/` e abra o `__init__.py` e comente a linha que importa `from move_river_t import move_river_t` (linha 51), depois na pasta `Grid_HYCOM` no arquivo `flood_fast.py` e comente na linha 9 o `import creep`.

Salve e tente importar o `pyroms_toolbox`:

```
1 python
2 import pyroms_toolbox
```

5.3 O model2roms

Nesta seção será mostrado como instalar o `model2roms`, programa utilizado para gerar as condições do ROMS. O `model2roms` é composto por vários scripts para criação dos arquivos de entrada e saída do ROMS. Foi originalmente iniciado por Trond Kristiansen (<https://github.com/trondkr/model2roms>) e foi adaptado para funcionar no cluster Kerana do INPE. Será seguido o mesmo esquema da seção anterior, com o endereço para baixar a biblioteca e os comandos que deverão ser digitados no terminal.

Para baixar o `model2roms`, digite no terminal:

```
1 git clone https://github.com/usutil/model2roms
```

5.3.1 O ETOPO1 1 Arc-Minute Global Relief Model

Para a grade do ROMS utilizamos os dados do ETOPO1 (<https://www.ngdc.noaa.gov/mgg/global/global.html>). O ETOPO1 (Figura 5.1) é produzido pelo National Geophysical Data Center (Colorado) e fornece duas camadas de informações de relevo. As camadas incluem batimetria sobre os oceanos e alguns dos principais lagos da Terra. A topografia terrestre do ETOPO1 e a batimetria oceânica baseiam-se na topografia SRTM30 e através de vários cruzeiros batimétricos.

5.3.2 Gerar a grade do ROMS

Neste caso, utilizamos o Matplotlib Basemap Toolkit para gerar a grade. Esta biblioteca utiliza os dados em ASCII do ETOPO1, que já está dentro da pasta Grade do `model2roms`. Também é necessário baixar os dados do ETOPO1 para a sua área de interesse, portanto, entre no site listado acima, clique em *Extract Custom Grid*, recorte a área desejada e baixe os dados com formato NetCDF.

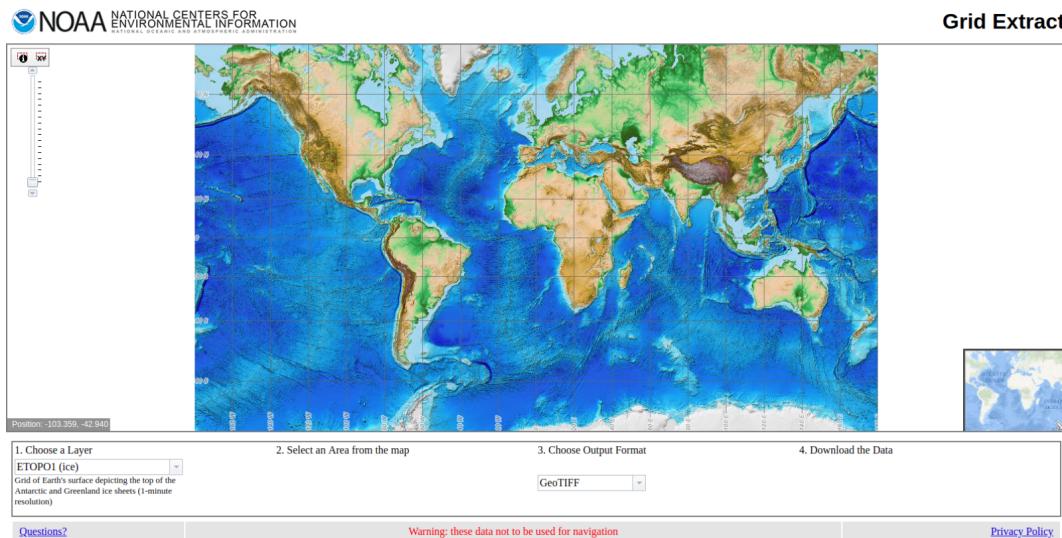


Figura 5.1. Apresentação do site do ETOPO1.

Com os dados do ETOPO1 baixados, abra o script *make_grid.py* para construir a grade do ROMS. E altere as variáveis de acordo com a Figura 5.2.

```

27 # Longitudes
28 lon0          = -70
29 lon1          = -20
30 lon2          = -20
31 lon3          = -70
32 # Latitudes
33 lat0          = -10
34 lat1          = -10
35 lat2          = -60
36 lat3          = -60
37 # Sets up a basemap with specified projections (try to make it bigger than the grid)
38 lat_0          = -80
39 lat_1          = -10
40 lat_2          = -10
41 lon_0          = -80
42 # Directories|
43 grd_name       = 'SAO'
44 main_dir       = '/home/uesleisutil/Documentos/PCI/model2roms/'
45 datadir        = main_dir+'Grade/'
46 bathy_data     = main_dir+'Grade/etopo1.nc'
47 etopo1_data    = main_dir+'Grade/etopo20data.gz'
48 etopo1_lat     = main_dir+'Grade/etopo20lats.gz'
49 etopo1_lon     = main_dir+'Grade/etopo20lons.gz'
50 grd_final      = main_dir+'Grade/'+grd_name+'_final.nc'
51 write_ROMS_grid_dir = main_dir+'Grade/'+grd_name+'_.grd.nc'
52 # Grid settings
53 Lm             = 600
54 Mm             = 600
55 hmin           = 10
56 theta_b        = 0.6
57 theta_s        = 5.0
58 Tcline          = 50
59 N               = 30
60 interp_method   = 'linear' # Options: 'linear' or 'nn' (Nearest Neighborhood)

```

Figura 5.2. Captura de tela de parte do script *make_grid.py*, que se encontra dentro da pasta *model2roms* no diretório de repositórios (*/home/adriano.sutil/repositorio/model2roms*).

É necessário delimitar os quatro pontos da grade com suas respectivas latitudes e longitudes, além de especificar as latitudes para o Basemap. É necessário colocar os diretórios de outputs e inputs. As configurações da grade do ROMS estão ordenadas como:

- **Lm**: Pontos de grade.
- **Mm**: Pontos de grade.
- **hmin**: Valor mínimo de h.
- **theta_b**: S-coordinate bottom control parameter
- **theta_s**: S-coordinate surface control parameter
- **Tcline**: Critical depth (hc) in meters (positive) controlling the stretching. It can be interpreted as the width of surface or bottom boundary layer in which higher vertical resolution.
- **N**: Número de camadas Sigma;
- **interp_method**: Método de interpolação da grade: Interpolação linear (*linear*) ou por Vizinho Próximo (*nn*).

Abra o arquivo *gridid.txt* e modifique os parâmetros no arquivo de acordo com o que está no arquivo *make_grid.py*. A mudança é necessária porque o PyROMS utiliza este arquivo para buscar as configurações de grade.

Após realizar as modificações, basta apenas rodar o script principal, digite no terminal:

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

5.3.3 O Simple Ocean Data Assimilation (SODA)

O *model2roms* foi desenvolvido para gerar as condições do ROMS utilizando o *Simple Ocean Data Assimilation* versão 3.7.1. O SODA é um conjunto de dados de reanálises desenvolvido por Carton e Giese (2008) e produzidos por um modelo de circulação global dos oceanos e a partir de assimilação de dados observacionais oriundas de diversas fontes. Os dados do SODA possuem um campo global de grade de 0,5°x 0,5°, 40 níveis verticais e resolução temporal de 5 dias ou mensal.

Os dados a cada 5 dias do SODA se encontram no cluster Kerana no diretório:

```
1 /home/luciano.pezzi/SODA3.3.1
```

5.3.4 Gerar as condições do ROMS

A *toolbox model2roms* completa se encontra dentro do repositório na Kerana. Para baixar, entre no diretório:

```
1 /home/adriano.sutil/repositorio/ROMS_scripts/model2roms
```

Ao abrir a pasta do *model2roms* é possível observar diversos scripts. Usaremos apenas o script *config.py* e o *compile.py*.

Compile a *toolbox* com o comando:

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

Abra o arquivo *config.py* e, conforme a Figura 5.3, pela definição *defineromsgridpath*. *ProjetoROMS* é o seu nome de projeto, portanto altere se achar conveniente. Em seguida altere o caminho onde está localizada a grade do ROMS. Em *defineforcingdatapath*, altere o diretório onde se encontram os dados do SODA.

```
def defineromsgridpath(self):
    return {'ProjetoROMS': '/home/usuario/Projeto/projeto_roms_grid.nc'}[self.outgrid]

def defineforcingdatapath(self):
    return {'SODA3': "/Users/usuario/Projeto/SODA/"}[self.indatatype]
```

Figura 5.3. Caminhos para identificar o nome do projeto, a grade do ROMS e o diretório do SODA.

Em seguida, faça alterações na guia *EDIT* conforme desejado (Figura 5.4), alterando entre *True* ou *False*.

- *self.showprogress*: Mostra uma barra de progresso. Caso deseje utilizar, é necessário instalar o pacote *progressbar2*. Instale através do Conda com o comando:

```
1 conda install -c anaconda progressbar2
```

```

# EDIT =====
# Set showprogress to "False" if you do not want to see the progress
# indicator for horizontal interpolation.
self.showprogress = False
# Set compileAll to True if you want automatic re-compilation of all the
# fortran files necessary to run model2roms. Options are "gfortran" or "ifort". Edit
# compile.py to add other Fortran compilers.
self.compileall = False
# Extract time-series of data for given longitude/latitude
self.extractstations = False
# Define a set of longitude/latitude positions with names to extract into
# station files (using extractStations)
if self.extractstations:
    # stationNames = ['NorthSea', 'Iceland', 'EastandWestGreenland', 'Lofoten', 'Georges Bank']
    # lonlist = [2.4301, -22.6001, -47.0801, 13.3801, -67.2001]
    # latlist = [54.5601, 63.7010, 60.4201, 67.5001, 41.6423]

    self.stationnames = ["Ytre Utsira", "Indre Utsira", "Lista"]
    self.latlist = [59.316667, 59.316667, 58.016667]
    self.lonlist = [4.800000, 4.983333, 6.533333]

# Create the bry, init, and clim files for a given grid and input data
self.createoceanforcing = True
# Create atmospheric forcing for the given grid
self.createatmosforcing = False # currently in beta stages and unavailable
# Create a smaller resolution grid based on your original. Decimates every second for
# each time run
self.decimategridfile = False
# Write ice values to file (for Arctic regions)
self.writeice = True
# Use ESMF for the interpolation. This requires that you have ESMF and ESMPy installed (import ESMF)
self.useesmf = True
# Apply filter to smooth the 2D fields after interpolation (time consuming but enhances results)
self.usefilter = True
# Format to write the output to: 'NETCDF4', 'NETCDF4_CLASSIC', 'NETCDF3_64BIT', or 'NETCDF3_CLASSIC'
# Using NETCDF4 automatically turns on compression of files (ZLIB)
self.myformat = 'NETCDF4'
self.myzlib = True
# Frequency of the input data: usually monthly
self.timefrequencyofinputdata = "month" # , "month", "hour"

```

Figura 5.4. Definições para utilizar o *model2roms* no script *config.py*.

- ***self.compileall***: Marque *True* caso queira compilar automaticamente toda vez que utilizar a *toolbox*.
- ***self.extractstations***: Caso queira extrair os dados de estações, fornecendo a latitude e longitude dos pontos.
- ***self.createoceanforcing***: Para criar as forçantes oceânicas.
- ***self.createatmosforcing***: Para criar as forçantes atmosféricas. Atualmente está em fase beta.
- ***self.decimategridfile***: Cria uma nova grade com resolução menor que a atual.
- ***self.writeice***: Caso a grade esteja em domínio com gelo, cria as variáveis para este fim.
- ***self.useesmf***: Usa o *ESMF* e o *ESMPy* para interpolação.
- ***self.usefilter***: Suaviza os campos 2D, porém consome mais tempo de processamento.
- ***self.myformat***: Define qual o formato do arquivo *netcdf*. Utilizar o formato *NETCDF4* automaticamente ativa a biblioteca *zlib*.
- ***self.timefrequencyofinputdata***: Define a frequência dos dados do SODA.

Na guia *IN GRIDTYPES* estão descritas as opções de entrada da grade e do SODA, conforme a Figura 5.5.

```

# IN GRIDTYPES -----
# Define what grid type you want to interpolate from (input MODEL data)
# Options:
# 1. SODA, 2. SODAMONTHLY, 3.WOAMONTHLY, 4. NORESM, 4. GLORYS, 5. SODA3
self.indatatype = 'SODA3'

# Define contact info for final NetCDF files
self.authorname = "Trond Kristiansen"
self.authoremail = "me (at) trondkristiansen.com"

# Define what grid type you want to interpolate from: Can be Z for SIGMA for ROMS
# vertical coordinate system or ZLEVEL. also define the name of the dimensions in the input files.
# Options:
# 1. SIGMA (not properly implemented yet), 2. ZLEVEL
self.ingridtype = "SIGMA"

# Define the names of the geographical variables in the input files
self.grdtype = 'regular'
self.lonname = "longitude"
self.latname = "latitude"
self.depthname = "depth"
self.timename = "time"
self.realm = "ocean"
self.fillvaluein = -1.e20

# OUT GRIDTYPES -----
# Define what grid type you want to interpolate to
# Options: This is just the name of your grid used to identify your selection later
self.outgrid = "ROMS"
self.outgridtype = "ROMS"

```

Figura 5.5. Definições de entrada para utilizar o *model2roms* no script *config.py*.

- *self.indatatype*: Seleciona qual a fonte de entrada dos dados.
- *self.authorname*: Escreve o nome do autor.
- *self.authoremail*: Escreve o email do autor.
- *self.ingridtype*: Define qual o tipo de grade do ROMS que será utilizada para gerar as condições.
- *self.grdtype*: Define qual o tipo de grade que será criada.
- *self.lonname*: Define o nome da variável de longitude.
- *self.depthname*: Define o nome da variável de profundidade.
- *self.timename*: Define o nome da variável de tempo.
- *self.realm*: Informa que está criando os dados oceânicos ou atmosféricos.
- *self.fillvaluein*: Valor de *fillvalue* utilizado nos arquivos *netcdf*.

Na aba *OUT GRIDTYPES* estão detalhados os dados de grade, conforme a figura 5.6.

```

# OUT GRIDTYPES -----
# Define what grid type you want to interpolate to
# Options: This is just the name of your grid used to identify your selection later
self.outgrid = "ProjetoROMS"
self.outgridtype = "ROMS"

# Subset input data. If you have global data you may want to subset these to speed up reading. Make
# sure that your input data are cartesian (0-360 or -180:180, -90:90)
self.subsetindata = False
if self.subsetindata:
    self.subset = self.definesubsetforindata()

# Define number of output depth levels
self.nlevels = 40
# Define the grid stretching properties (leave default if uncertain what to pick)
self.vstretching = 4
self.vtransform = 2
self.theta_s = 7.0
self.theta_b = 0.1
self.tcline = 250.0
self.hc = 250

```

Figura 5.6. Definições de saída para utilizar o *model2roms* no script *config.py*.

- *self.outgrid*: O nome do seu projeto do ROMS.
- *self.outgridtype*: O tipo de grade que será criado.
- *self.nlevels*: A quantidade de níveis verticais ds condições.
- *self.vstretching*: Número da opção da função de alongamento das coordenadas verticais.
- *self.vtransform*: Número da opção da função de transformações de coordenadas verticais.
- *self.theta_s*: Parâmetro usado para controlar as coordenadas na superfície.
- *self.theta_b*: Parâmetro usado para controlar as coordenadas na camada inferior.
- *self.tcline*: Parâmetro usado para controlar a largura entre a camada superficial e interior.
- *self.hc*: Parâmetro utilizado para a camada crítica.

Na aba *DATE AND TIME DETAILS* estão detalhados as datas de início e fim do experimento, conforme a Figura 5.7.

```

# DATE AND TIME DETAILS -----
# Define the period to create forcing for
self.start_year = 2013
self.end_year = 2013
self.start_month = 1
self.end_month = 12
self.start_day = 15
self.end_day = 15

```

Figura 5.7. Definições de datas para utilizar o *model2roms* no script *config.py*.

- *self.start_year*: Ano inicial.
- *self.end_year*: Ano final.
- *self.start_month*: Mês inicial.
- *self.end_month*: Mês final.
- *self.start_day*: Dia inicial.

- *self.end_day*: Dia final.

Agora, com os dados todos completados, basta executar a *toolbox* com o comando:

```
1 ipython config.py --pylab
```

5.4 Compilando o ROMS com o modelo de gelo marinho



6. Construindo o SWAN

O SWAN utiliza os arquivos com extensão *.bot* e *.grd* para suas simulações. Neste caso, criaremos eles a partir da grade do ROMS, porém sem introduzir dados iniciais, pois como ele rodará acoplado, apenas alguns dias de simulação são o suficiente para que o modelo estabilize e envie as troca de informação entre os modelos ocorra normalmente.

Neste caso, utilize o script do MATLAB *make_swan.m* que está dentro da pasta de repositório:

```
1 /home/nome.sobrenome/repositorio/SWAN_scripts
```

Conforme a Figura 6.1, o script possui a seguinte construção:

```
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);
```

Figura 6.1. O script *make_swan.m*.

Para gerar os dois arquivos do SWAN, procure no script a variável *ncfile* e altere o diretório para o caminho onde está sua grade do ROMS.

Execute o script e serão criados os dois arquivos: *swan_coord.grd* e *swan_bathy.bot*. Os dois arquivos deverão ser colocados dentro da sua pasta de projetos.



7. Construindo os pesos entre grades com o SCRIP

7.1 Construindo os pesos

Como visto na seção 1.5, o SCRIP é empregado para interpolar os pesos entre duas ou mais grades de modelos diferentes. No COAWST, o pacote foi modificado para gerar somente um arquivo NetCDF que sera utilizado durante as integrações.

O diretório do SCRIP está localizado em:

```
1 /home/nome.sobrenome/COAWST/Lib/SCRIP
```

Dentro da pasta, procure pelo arquivo com extensão *.in*. Como no exemplo da Figura 7.1:

```

$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_NCPFILE='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)=1

| 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
PARENT_ID(2)=1

| The $END statement below is required

```

Figura 7.1. Arquivo .in do SCRIP para o projeto Sandy.

Em *OUTPUT_NCFILE*, altere se julgar necessário, o nome do arquivo NetCDF que será gerado.

Na seção 2 do arquivo, altere as variáveis *NGRIDS_ROMS*, *NGRIDS_SWAN* e *NGRIDS_WRF* de acordo com o número de grades, existentes no seu projeto, no ROMS, SWAN e WRF, respectivamente.

Na terceira seção do arquivo, renove os diretórios das grades do ROMS de acordo com os nomes no seu projeto.

Para o SWAN, na quarta seção, além de mudar os diretórios das grades do SWAN (*SWAN_COORD* e *SWAN_BATH*), altere o número de pontos de grade existentes, de acordo com o seu projeto, nas variáveis *SWAN_NUMX* e *SWAN_NUMY*.

Por fim, na quinta seção, altere os diretórios das grades do WRF (*WRF_GRIDS*). Em *PARENT_GRID_RATIO*, caso seu projeto conte com o aninhamento entre as grades do WRF, altere para a relação usada entre as grades usadas no seu projeto. Em *PARENT_ID*, adicione a identificação das grades.

Salve as modificações no arquivo *.in*.

Para executar o SCRIP, procure no repositório pelo arquivo *qsub_scrip.sh*:

```
1 /home/nome.sobrenome/repositorio/qsub_scrip.sh
```

Mova o arquivo para o diretório do SCRIP:

```
1 mv /home/nome.sobrenome/repositorio/qsub_scrip.sh /home/nome.sobrenome/COAWST/Lib/SCRIP
```

Abra o arquivo *qsub_scrip.sh*:

```
1 nedit qsub_scrip.sh
```

Altere e salve o arquivo *.sh*, como no exemplo da Figura 7.2:

```
#!/bin/sh
#PBS -l mppwidth=1
#PBS -N SCRIP
#PBS -j oe
#PBS -o Joe_test.out
#PBS -W walltime=33600:00:00
#PBS -q workq
#echo "Running GEOPGRID 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 OMP_NUM_THREADS=1
# export OMP_NUM_THREADS=1
#
# EXECDIR=/scratch/nome.sobrenome/COAWST/Lib/SCRIP
chmrt
# export ATP_ENABLED=1
# ulimit -a
# ulimit -c unlimited
# ulimit -s unlimited
# ulimit -m unlimited
# ulimit -s

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

Figura 7.2. Arquivo *.sh* usado para executar o SCRIP.

Para iniciar o SCRIP, digite:

```
1 qsub qsub_scrip.sh
```

Ao final será criado o arquivo *scrip_static.nc*. Agora mová-os para a pasta do seu projeto e pronto! O COAWST está pronto para ser executado.

7.2 Executanto seu projeto no COAWST

Agora, com suas condições e grades prontas, seu projeto está pronto para ser executado. Visite a Seção 2.10 para relembrar como executar o projeto.



8. Trabalhos do LOA

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 e 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.

J. Pullen et al. (2017). *The Science of Ocean Prediction, The Sea*. P. Lermusiaux and K. Brink. Chap. Coupled ocean-atmosphere modeling and predictions

Disponível em: 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 e 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.

L. F. Mendonça et al. (2017). “Regional modeling of the water masses and circulation annual variability at the Southern Brazilian Continental Shelf”. In: *Journal of Geophysical Research: Oceans* 122, pp. 1232–1253.
DOI: [10.1002/2016JC011780](https://doi.org/10.1002/2016JC011780)

Disponível em: <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, 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.

C. K. Parise et al. (2014). “The Influence of Sea Ice Dynamics on the Climate Sensitivity and Memory to Increased Antarctic Sea Ice”. In: *Journal of Climate* 28, pp. 9642–9668. DOI: 10.1175/JCLI-D-14-00748.1

Disponível em: <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.

D. F. Dias et al. (2014). “Modeling the spawning strategies and larval survival of the Brazilian sardine (*Sardinella brasiliensis*)”. In: *Progress in Oceanography* 123, pp. 38–53. DOI: 10.1016/j.pocean.2014.03.009

Disponível em: <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.

I. P. Silveira & L. P. Pezzi (2014). “Sea surface temperature anomalies driven by oceanic local forcing in the Brazil-Malvinas Confluence”. In: *Ocean Dynamics* 347.64, pp. 347–360. DOI: 10.1007/s10236-014-0699-4

Disponível em: <https://link.springer.com/article/10.1007%2Fs10236-014-0699-4>



Agradecimentos

Ao CNPq pela Bolsa de Capacitação Interna do Instituto Nacional de Pesquisas Espaciais, processo 301110/2017-4 fornecida a U. A. Sutil e também pela bolsa do programa de Produtividade em Pesquisa concedida a L. P. Pezzi (CNPq 304009/2016-4).

A CAPES pelo fomento no projeto 'Advanced Studies in Medium and High Latitudes Oceanography' (23038.004304/2014-28).

Ao Trond Kristiansen (<https://github.com/trondkr/model2roms>) por disponibilizar o pacote *model2roms*, que auxiliou no desenvolvimento para gerar as condições do ROMS.

Ao Matheus Fagundes pela ajuda no Python.

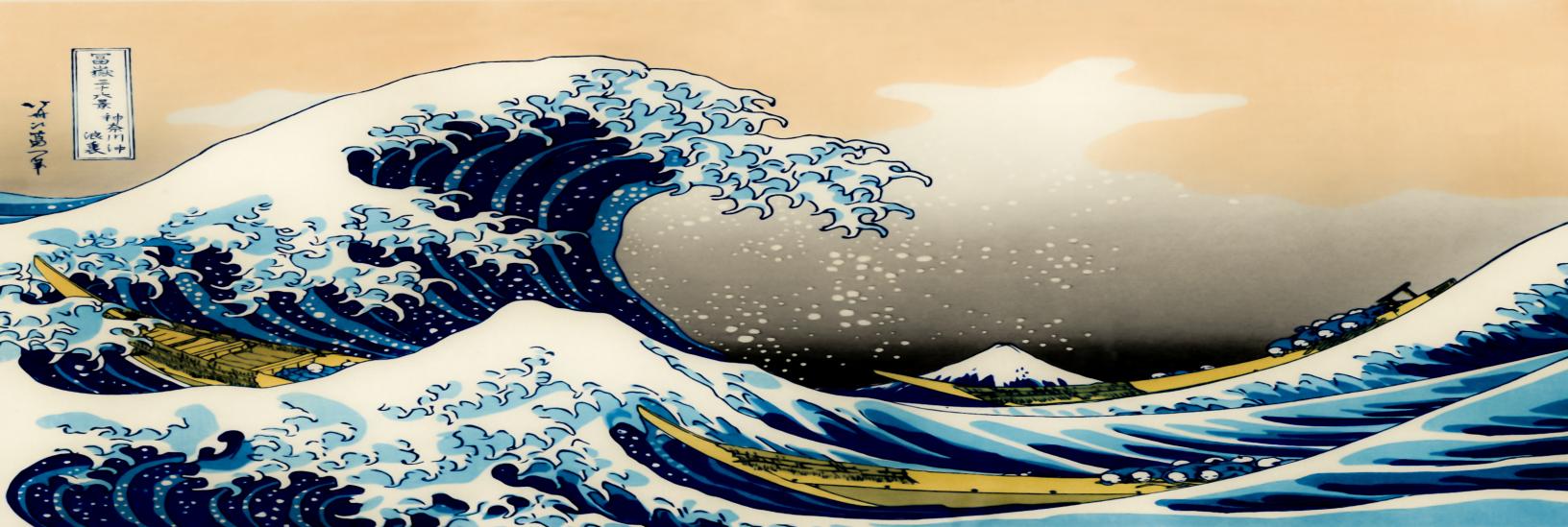
Ao João Hackerott pela ajuda em compilar o WPS no cluster Kerana e nas inúmeras dúvidas com o WRF.

À Eliana Bertol Rosa e Mylene Cabrera pela revisão do guia.

À toda equipe de modelagem que desenvolveu o ROMS e distribuiu de forma aberta os códigos fontes, especialmente ao Hernan G. Arango e Kate Hedstrom.

Ao John Warner, idealizador e desenvolvedor do COAWST.

Ao Mathias Legrand, Vel e Andrea Hidalgo pelo template no L^AT_EX.



Referências bibliográficas

- Arakawa, A. & V. R. Lamb (1977). "Computational design of the basic dynamical processes of the UCLA general circulation model". In: *Methods in Computational Physics* 17, pp. 173–265. DOI: 10.1016/B978-0-12-460817-7.50009-4.
- Booij, N., L. H. Holthuijsen & R. C. Ris (1996). "The SWAN Wave Model for Shallow Water". In: *Coastal Engineering Proceedings* 25, pp. 668–675. DOI: 10.9753/icce.v25.%25p.
- Booij, N., R. C. Ris & L. H. Holthuijsen (1999). "A third-generation wave model for coastal regions: 1. Model description and validation". In: *Journal of Geophysical Research: Oceans* 104.C4, pp. 7649–7666. DOI: 10.1029/98JC02622. URL: <http://dx.doi.org/10.1029/98JC02622>.
- Charnock, H. (1955). "Wind stress on a water surface". In: *Quarterly Journal of the Royal Meteorology Society* 81.350, pp. 639–640. DOI: 10.1002/qj.49708135027.
- Dias, D. F., L. P. Pezzi, D. F. M. Gherardi & R. Camargo (2014). "Modeling the spawning strategies and larval survival of the Brazilian sardine (*Sardinella brasiliensis*)". In: *Progress in Oceanography* 123, pp. 38–53. DOI: 10.1016/j.pocean.2014.03.009.
- Duda, Michel (2008). *The WRF Preprocessing System*. 2006 WRF-ARW Summer tutorial.
- Gouveia, M. B. (2015). "A influência dos processos de circulação oceânica nos extremos de produção comercial da Sardinha Verdadeira". Dissertação de Mestrado. São José dos Campos, SP: Sensoriamento Remoto, Instituto Nacional de Pesquisas Espaciais.
- Haidvogel, D. B., H. G. Arango, W. P. Budgell, B. D. Cornuelle, E. Curchitser, E. Lorenzo, K. Fennel, W. R. Geyer, A. J. Hermann, L. Lanerolle, J. Levin, J. C. McWilliams, A. J. Miller, A. M. Moore, T. M. Powell, A. F. Shchepetkin, C. R. Sherwood, R. P. Signell, J. C. Warner & J. Wilkin (2008). "Ocean forecasting in terrain-following coordinates: Formulation and skill assessment of the Regional Ocean Modeling System." In: *J. Comput. Physics* 227.7, pp. 3595–3624. DOI: 10.1016/j.jcp.2007.06.016.
- Jacob, R. L., J. W. Larson & E. T. Ong (2005). "M x N Communication and Parallel Interpolation in Community Climate System Model Version 3 Using the Model Coupling Toolkit." In: *IJHPCA* 19.3, pp. 293–307. DOI: 10.1177/1094342005056116.

- Jones, P. W. (1998). *A User's Guide for SCRIP: A Spherical Coordinate Remapping and Interpolation Package version 1.5*. Los Alamos, NM. URL: <http://oceans11.lanl.gov svn/SCRIP/trunk/SCRIP/doc/SCRIPusers.pdf>.
- (1999). “First and Second Order Conservative Remapping Schemes for Grids in Spherical Coordinates”. In: *Monthly Weather Review* 127, pp. 2204–2210. DOI: 10.1175/1520-0493(1999)127<2204:FASOCR>2.0.CO;2.
- Larson, J. W., R. L. Jacob & E. T. Ong (2005). “The Model Coupling Toolkit: A New Fortran90 Toolkit for Building Multiphysics Parallel Coupled Models.” In: *IJHPCA* 19.3, pp. 277–292. DOI: 10.1177/1094342005056115. URL: <http://dblp.uni-trier.de/db/journals/ijhpc/ijhpc19.html#LarsonJ005>.
- Mendonça, L. F., R. B. Souza, L. P. Pezzi, O. O. Möller & R. C. M. Alves (2017). “Regional modeling of the water masses and circulation annual variability at the Southern Brazilian Continental Shelf”. In: *Journal of Geophysical Research: Oceans* 122, pp. 1232–1253. DOI: 10.1002/2016JC011780.
- Miller, A. J., M. Collins, G. Silvio, T. G. Jensen, V. Misra, L. P. Pezzi, D. W. Pierce, D. Putrasahan, H. Seo & Y.-H. Tseng (2018). “Coupled ocean-atmosphere modeling and predictions”. In: *Journal of Marine Research* 42.3, pp. 361–402. DOI: 10.1357/002224017821836770.
- Parise, C. K., L. P. Pezzi, K. I. Hodges & F. Justino (2014). “The Influence of Sea Ice Dynamics on the Climate Sensitivity and Memory to Increased Antarctic Sea Ice”. In: *Journal of Climate* 28, pp. 9642–9668. DOI: 10.1175/JCLI-D-14-00748.1.
- Pullen, J., R. Allard, H. Seo, A. J. Miller, S. Chen, L. P. Pezzi, T. Smith, P. Chu, J. Alves & R. Caldeira (2017). *The Science of Ocean Prediction, The Sea*. P. Lermusiaux and K. Brink. Chap. Coupled ocean-atmosphere modeling and predictions.
- Pullen, J., R. Allard, H. Seo, A. J. Miller, S. Chen, L. P. Pezzi, T. Smith, P. Chu, J. Alves & R. Caldera (2018). “Coupled ocean-atmosphere forecasting at short and medium time scales”. In: *Journal of Marine Science* 17, p. 1.
- Saha, Suranjana, Shrinivas Moorthi, Hua-Lu Pan, Xingren Wu, Jie Wang, Sudhir Nadiga, Patrick Tripp, Robert Kistler, John Woollen, David Behringer, Haixia Liu, Diane Stokes, Robert Grumbine, George Gayno, Jun Wang, Yu-Tai Hou, Hui-Ya Chuang, Hann-Ming H. Juang, Joe Sela, Mark Iredell, Russ Treadon, Daryl Kleist, Paul Van Delst, Dennis Keyser, John Derber, Michael Ek, Jesse Meng, Helin Wei, Rongqian Yang, Stephen Lord, Huug van den Dool, Arun Kumar, Wanqiu Wang, Craig Long, Muthuvvel Chelliah, Yan Xue, Boyin Huang, Jae-Kyung Schemm, Wesley Ebisuzaki, Roger Lin, Pingping Xie, Mingyue Chen, Shuntai Zhou, Wayne Higgins, Cheng-Zhi Zou, Quanhua Liu, Yong Chen, Yong Han, Lidia Cucurull, Richard W. Reynolds, Glenn Rutledge & Mitch Goldberg (2010). *NCEP Climate Forecast System Reanalysis (CFSR) 6-hourly Products, January 1979 to December 2010*. Boulder, CO. URL: <https://doi.org/10.5065/D69K487J>.
- Shchepetkin, A. F. & J. C. McWilliams (2005). “The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model”. In: *Ocean Modelling* 9, pp. 347–404. DOI: 10.1016/j.ocemod.2004.08.002.
- Silva, P. E. D. (2013). “Caracterização do padrão de ondas na costa do Brasil por meio de modelagem numérica”. Dissertação de Mestrado. São José dos Campos, SP: Meteorologia, Instituto Nacional de Pesquisas Espaciais.

- Silveira, I. P. & L. P. Pezzi (2014). “Sea surface temperature anomalies driven by oceanic local forcing in the Brazil-Malvinas Confluence”. In: *Ocean Dynamics* 347.64, pp. 347–360. DOI: 10.1007/s10236-014-0699-4.
- Skamarock, W. C., J. B. Klemp, J. Dudhia, D. O. Gill, D. M. Barker, M. G. Duda, X.-Y. Huang, W. Wang & J. G. Powers (2008). “A Description of the Advanced Research WRF Version 3.” In: NCAR TN-475+STR.
- UCAR (2017). *The NCAR Command Language*. Version 6.4.0. URL: <http://dx.doi.org/10.5065/D6WD3XH5>.
- Warner, J. C. (Apr. 2018). *A Coupled-Ocean-Atmosphere-Wave- Sediment Transport Modeling System*. URL: <https://woodshole.er.usgs.gov/operations/modeling/COAWST/>.
- Warner, J. C., B. Armstrong, R. He & J. B. Zambon (2010). “Development of a Coupled Ocean–Atmosphere–Wave–Sediment Transport (COAWST) Modeling System”. In: *Ocean Modelling* 35.3, pp. 230–244. DOI: 10.1016/j.ocemod.2010.07.010.
- Warner, J. C., C. R. Sherwood, R. P. Signell, C. K. Harris & H. G. Arango (2008). “Development of a three-dimensional, regional, coupled wave, current, and sediment-transport model.” In: *Computers & Geosciences* 34.10, pp. 1284–1306. DOI: 10.1016/j.cageo.2008.02.012.