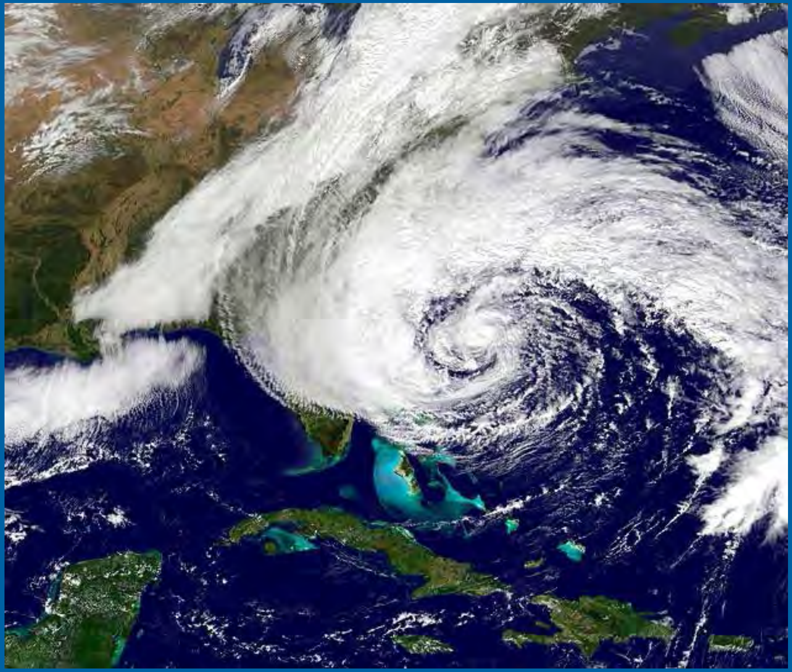
火曜日午後 1分～

ROMS Application Tutorial



Projects/Sandy- Hurricane Sandy example

U.S. Department of the Interior
U.S. Geological Survey

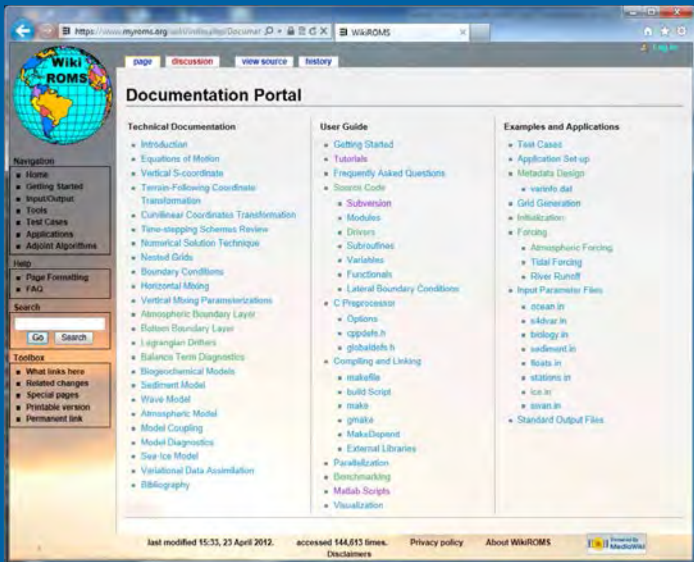
Regional Ocean Modeling System

とばす

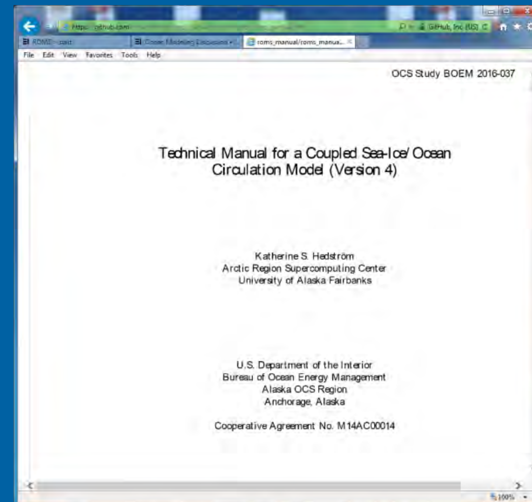
- Free surface, hydrostatic ocean model
- Finite-difference 3D Reynolds-averaged Navier-Stokes equations
- Horizontal orthogonal curvilinear Arakawa C grid
- Vertical stretched terrain-following Sigma coordinates
- Wide range of advection schemes: (e.g. 3rd-order upstream-biased, 4^{th}-order)
- Wide range of open boundary conditions: (e.g. Radiation, clamped, nudged)
- CF-compliant NetCDF I/O
- Wide range of vertical mixing schemes (k-epsilon, k-omega, MY2.5, KPP, GLS)
- Ice model
- Biological modules
- Model adjoint for data assimilation
- Fortran 90; Runs on Unix, Mac, and Windows
- Parallel code in MPI and OpenMP

ROMS information

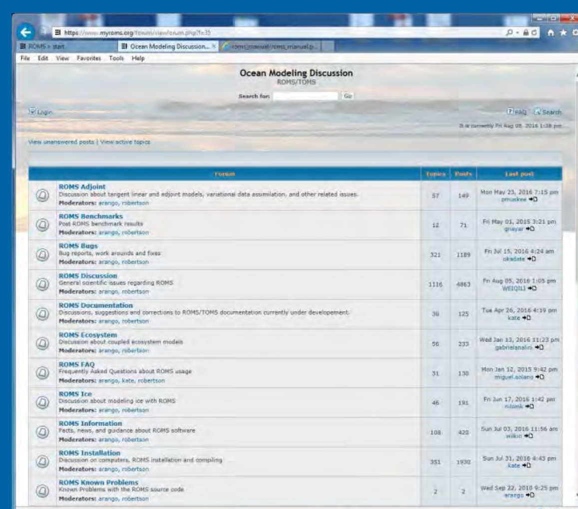
5:50~



https://www.myroms.org/wiki

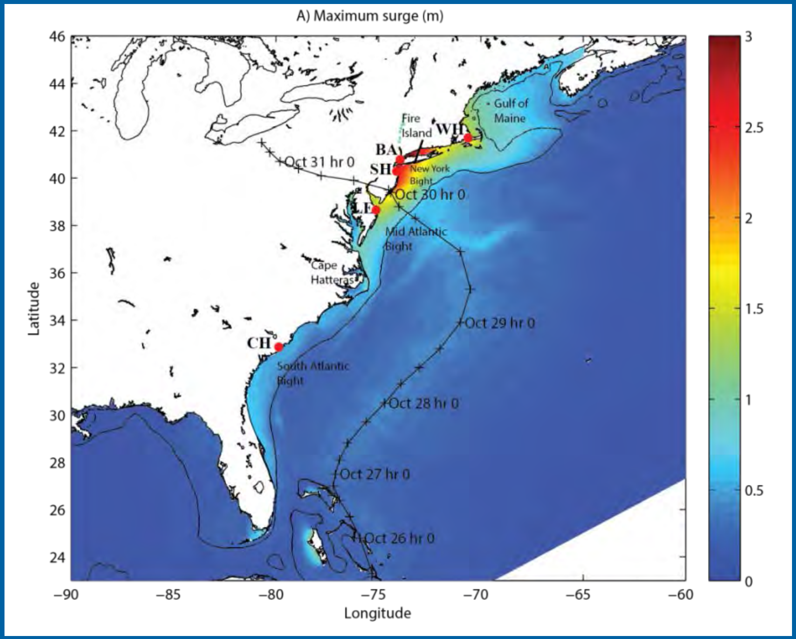


https://github.com/kshedstrom/roms_manual/blob/master/roms_manual.pdf

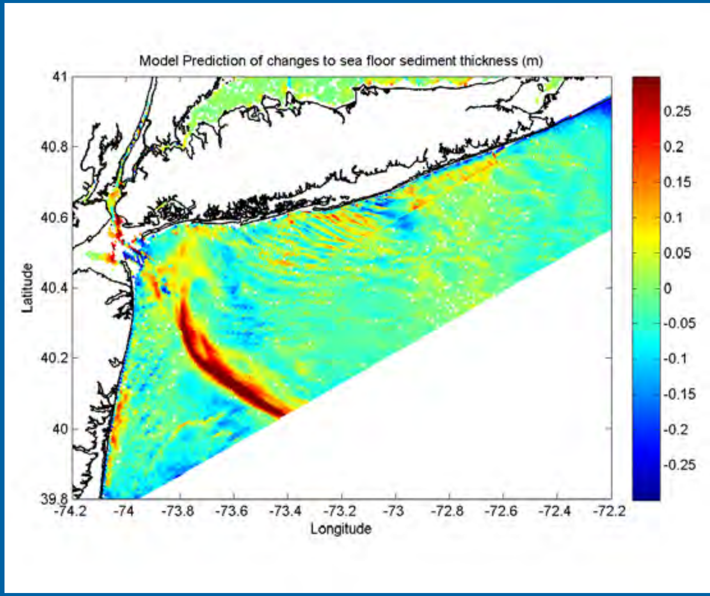


https://www.myroms.org/forum/index.php

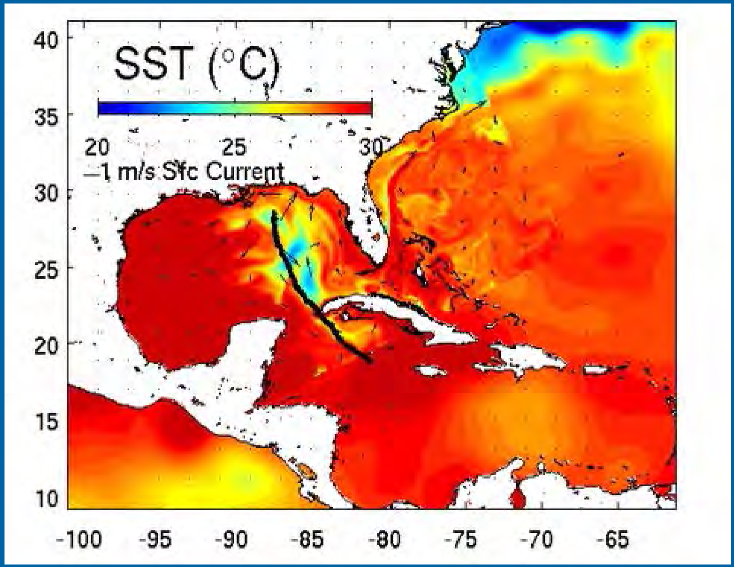
Wide Range of realistic Applications



Storm surge (H. Sandy)



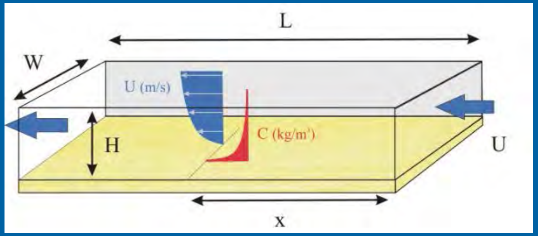
seafloor bed thickness change (H. Sandy)

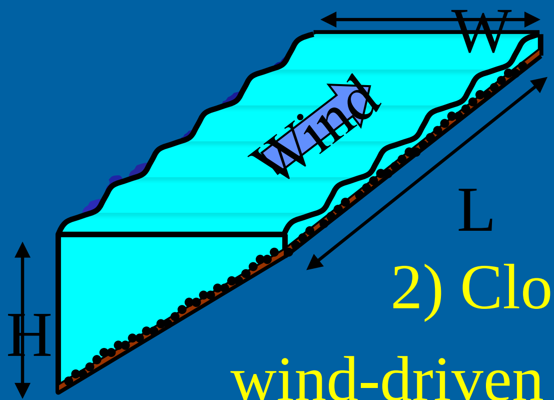


SST (H. Ivan)

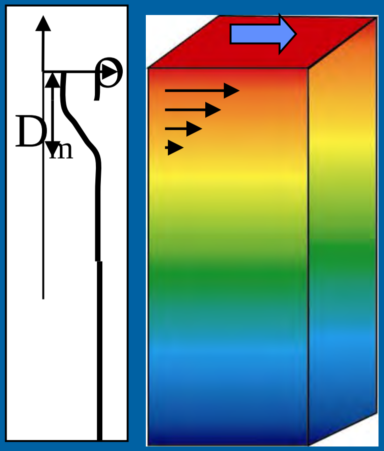
Test Cases

1) Open channel flow

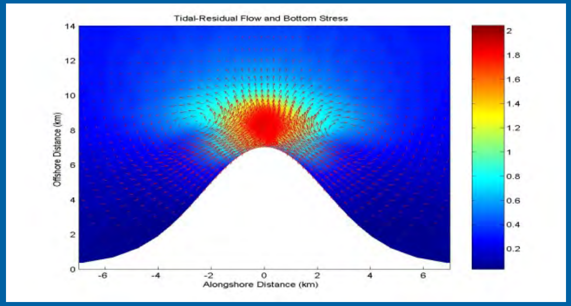




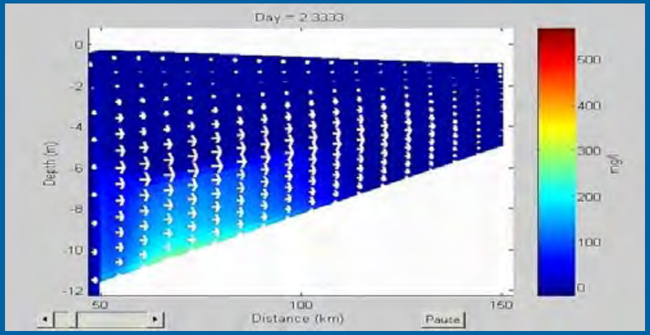
2) Closed basin,
wind-driven circulation



3) Mixed layer deepening



4) Tidal flow around a headland



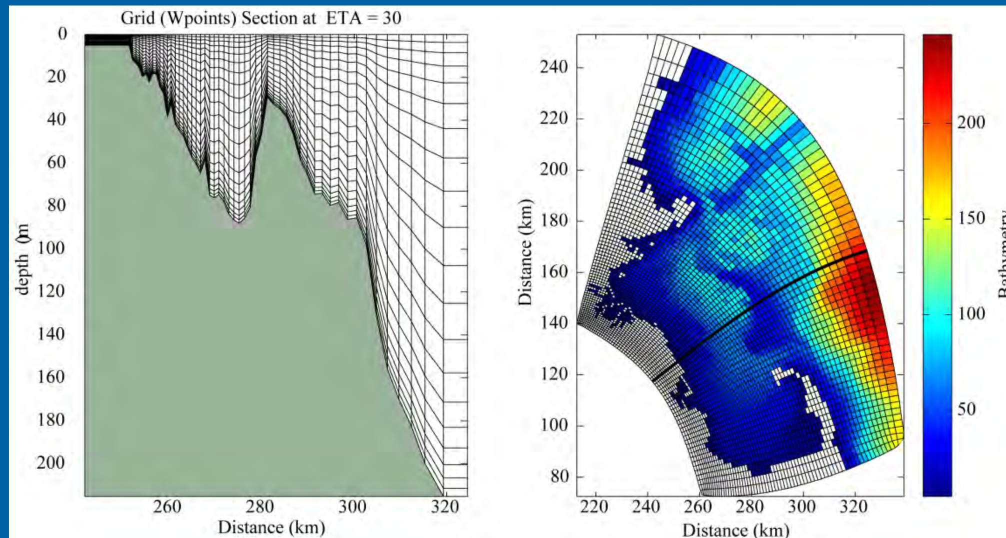
5) Estuarine circulation

http://woodshole.er.usgs.gov/project-pages/sediment-transport/

ROMS Grid

9分45秒~

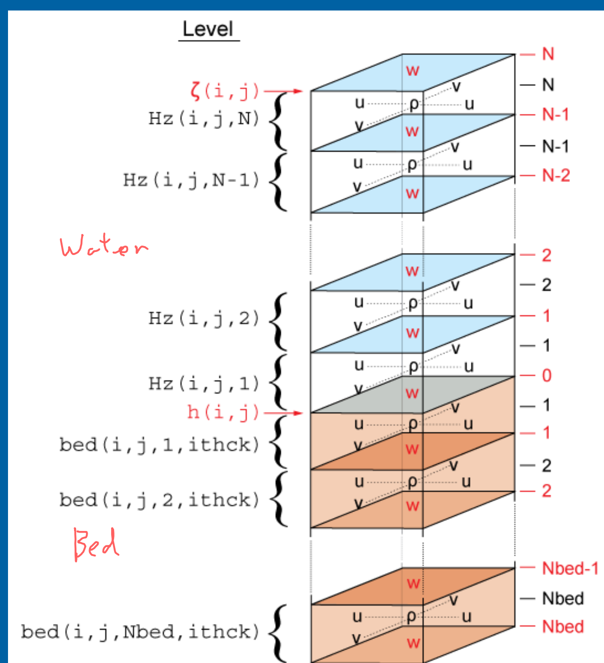
terrain following
coordinates

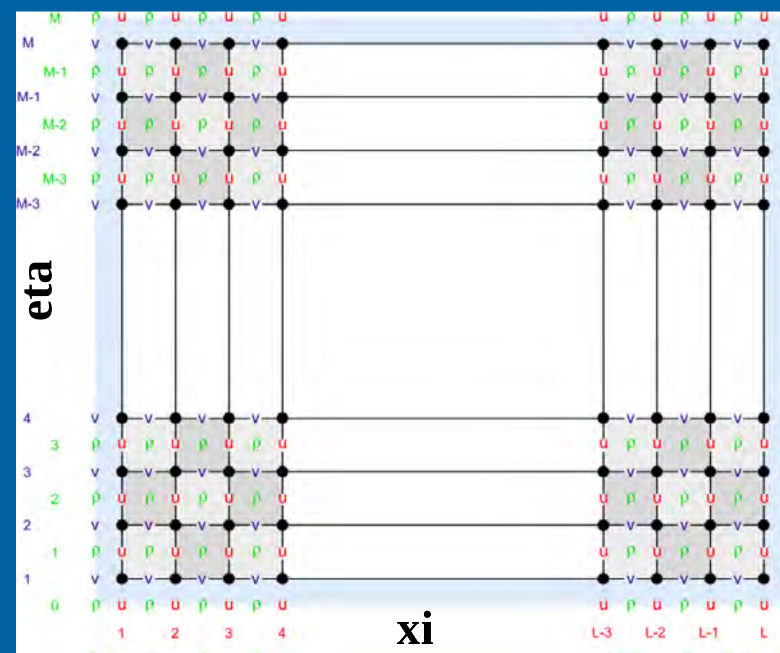


- masking
- curvature
- stretching

Horizontal
Arakawa "C" grid

Vertical

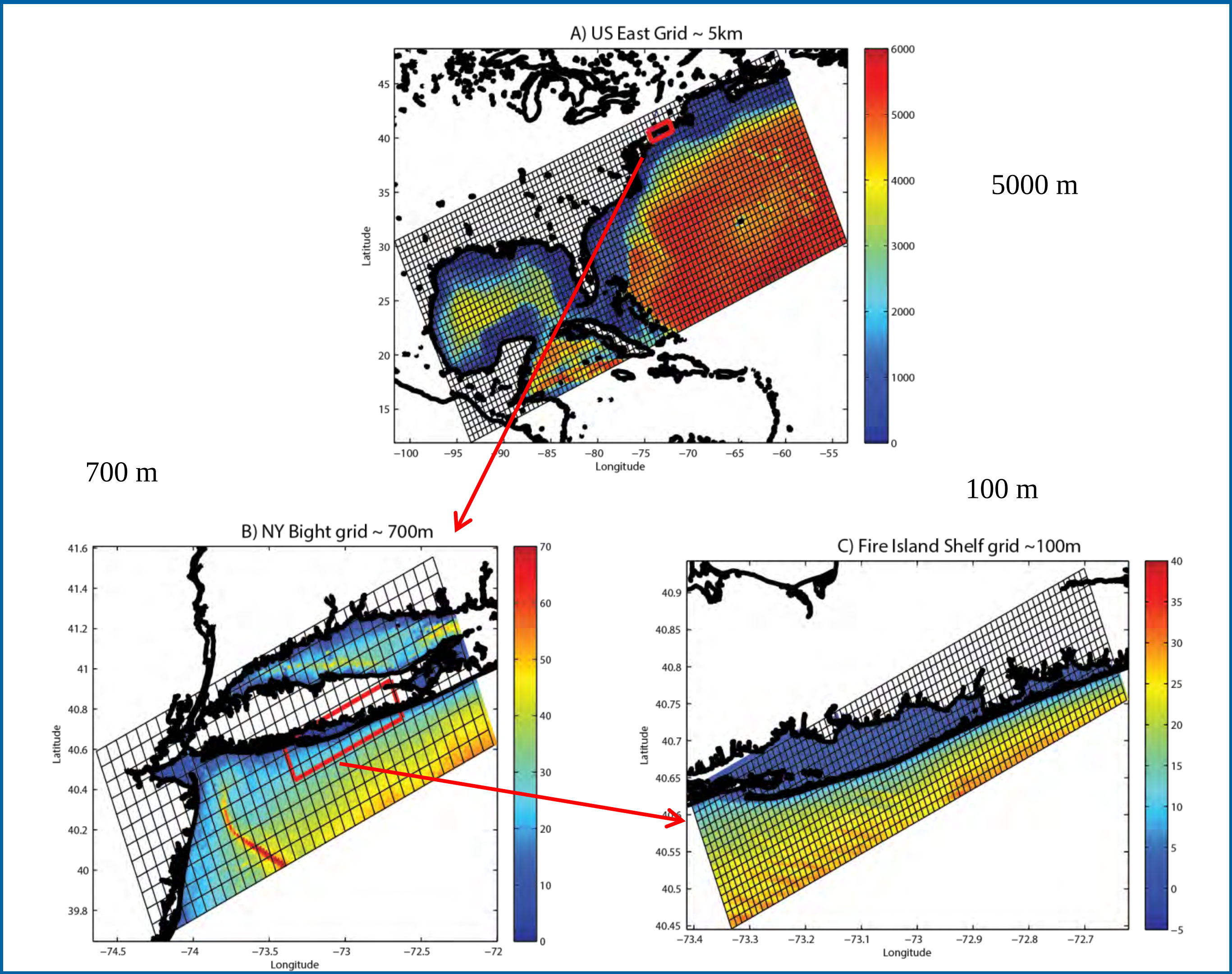




USGS

Grid resolution – horizontal static nests

For structured grids, use grid refinement to increase resolution.



Want to increase resolution in areas of strong gradients
and in areas of changing processes.

14分50秒～

Terrain following transformations

(Vtransform + Vstrectching)

Vtransform

Transformation Equations

The following vertical coordinate transformations are available:

$$z(x,y,\sigma,t) = S(x,y,\sigma) + \zeta(x,y,t)\left[1 + \frac{S(x,y,\sigma)}{h(x,y)}\right], \qquad (1)$$
$$S(x,y,\sigma) = h_c\,\sigma + [h(x,y) - h_c]\;C(\sigma)$$

1

or

$$z(x,y,\sigma,t) = \zeta(x,y,t) + [\zeta(x,y,t) + h(x,y)]\;S(x,y,\sigma), \qquad (2)$$
$$S(x,y,\sigma) = \frac{h_c\,\sigma + h(x,y)\,C(\sigma)}{h_c + h(x,y)}$$

★ **2**

where $S(x,y,\sigma)$ is a nonlinear vertical transformation functional, $\zeta(x,y,t)$ is the time-varying free-surface, $h(x,y)$ is the unperturbed water column thickness and $z = -h(x,y)$ corresponds to the ocean bottom, σ is a fractional vertical stretching coordinate ranging from $-1 \le \sigma \le 0$, $C(\sigma)$ is a nondimensional, monotonic, vertical stretching function ranging from $-1 \le C(\sigma) \le 0$, and h_c is a positive thickness controlling the stretching. In sediment applications, $h = h(x,y,t)$ is changed at every time-step since it is affected by erosion and deposition processes.

USGS

https://www.myroms.org/wiki/index.php/Vertical_S-coordinate

Vstretch (1 of 4)

Available Stretching Functions:

1. Song and Haidvogel (1994) function available in ROMS since its beginning, Vstretching = 1. $C(\sigma)$ is defined as:

$$C(\sigma) = (1 - \theta_B)\frac{\sinh(\theta_S\,\sigma)}{\sinh\theta_S} + \theta_B\left[\frac{\tanh[\theta_S(\sigma + \frac{1}{2})]}{2\tanh(\frac{1}{2}\theta_S)} - \frac{1}{2}\right] \tag{5}$$

where θ_S and θ_B are the surface and bottom control parameters. Their ranges are $0 < \theta_S \leq 20$ and $0 \leq \theta_B \leq 1$, respectively. This function has the following features:

- It is infinitely differentiable in σ.
- The larger the value of θ_S, the more resolution is kept above h_c.
- For $\theta_B = 0$, the resolution all goes to the surface as θ_S is increased.
- For $\theta_B = 1$, the resolution goes to both the surface and the bottom equally as θ_S is increased.
- For $\theta_S \neq 0$, there is a subtle mismatch in the discretization of the model equations, for instance in the horizontal viscosity term. We recommend that you stick with *reasonable* values of θ_S, say $\theta_S \leq 8$.
- Some applications turn out to be sensitive to the value of θ_S used.

USGS

Vstretch (2 of 4)

2. A. Shchepetkin (2005) UCLA-ROMS deprecated function, Vstretching = 2. $C(\sigma)$ is defined as a piecewise function:

$$C(\sigma) = \mu\, C_{sur}(\sigma) + (1-\mu)\, C_{bot}(\sigma),$$

$$C_{sur}(\sigma) = \frac{1-\cosh(\theta_S\,\sigma)}{\cosh(\theta_S)-1}, \qquad \text{for } \theta_S > 0, \qquad C_{bot}(\sigma) = \frac{\sinh[\theta_B\,(\sigma+1)]}{\sinh(\theta_B)} - 1,$$

$$\mu = (\sigma+1)^{\alpha} \left[1 + \frac{\alpha}{\beta}\left(1-(\sigma+1)^{\beta}\right)\right]$$

This function is similar in meaning to the Song and Haidvogel (1994), but note that hyperbolic function in C_{sur} is $\cosh$ instead of $\sinh$ and

$$\frac{\partial C}{\partial \sigma} \to 0 \qquad \text{as } \sigma \to 0.$$

Vstretch (3 of 4)

3. R. Geyer function for high bottom boundary layer resolution in relatively shallow applications, Vstretching = 3. $C(\sigma)$ is defined as a piecewise function:

$$C(\sigma) = \mu\, C_{bot}(\sigma) + (1-\mu)\, C_{sur}(\sigma),$$

$$C_{sur}(\sigma) = -\frac{\log[\cosh(\gamma\, \mathrm{abs}(\sigma)^{\theta_S}]}{\log[\cosh(\gamma)]}, \qquad C_{bot}(\sigma) = \frac{\log[\cosh(\gamma\,(\sigma+1)^{\theta_B}]}{\log[\cosh(\gamma)]} - 1,$$

$$\mu = \frac{1}{2}\left[1 - \tanh\left(\gamma\left(\sigma + \frac{1}{2}\right)\right)\right]$$

where the power exponents θ_S and θ_B are the surface and bottom control parameters specified in standard input file ocean.in, as before. Here, γ is a scale factor for all hyperbolic functions. Currently, $\gamma = 3$. Typical values for the control parameters are:

$\theta_S = 0.65$	minimal increase of surface resolution
$\theta_S = 1.0$	significant surface amplification
$\theta_B = 0.58$	no bottom amplification
$\theta_B = 1.0$	significant bottom amplification

USGS

Vstretch (4 of 4)

4. A. Shchepetkin (2010) UCLA-ROMS current function, Vstretching = 4.$C(\sigma)$ is defined as a continuous, double stretching function:

   Surface refinement function:

$$C(\sigma) = \frac{1 - \cosh(\theta_S\,\sigma)}{\cosh(\theta_S) - 1}, \qquad \text{for } \theta_S > 0, \qquad C(\sigma) = -\sigma^2, \qquad \text{for } \theta_S \leq 0$$

   Bottom refinement function:

$$C(\sigma) = \frac{\exp(\theta_B\,C(\sigma)) - 1}{1 - \exp(-\theta_B)}, \qquad \text{for } \theta_B > 0 \tag{9}$$

   Notice that the bottom function (9) is the second stretching of an already stretched transform (8). The resulting stretching function is continuous with respect to θ_S and θ_B as their values approach zero. The range of meaningful values for θ_S and θ_B are:

$$0 \leq \theta_S \leq 10 \qquad \text{and} \qquad 0 \leq \theta_B \leq 4$$

   However, users need to pay attention to extreme r-factor (rx1) values near the bottom.

   Due to its functionality and properties Vtransform = 2 and Vstretching = 4 are now the default values for ROMS.

USGS

Equations in Mass Flux form

$$\left(\mathbf{u}^l,\omega^l\right)=\left(\mathbf{u},\omega\right)+\left(\mathbf{u}^{St},\omega^{St}\right)$$

$u,\ v,\ \omega_s$	= Eulerian Velocity
$u^l,\ v^l,\ \omega_s{}^l$	= Lagrangian Velocity
$u^{st},\ v^{st},\ \omega_s{}^{St}$	= Stokes Velocity
f	= Coriolis parameter
φ	= Dynamic pressure
H_z	= Grid cell thickness
$\mathcal{F}^{\eta};\mathcal{F}^{\xi}$	= Non wave body force
$D^{\eta};D^{\xi}$	= Momentum mixing terms
$\mathcal{F}^{w\eta};\mathcal{F}^{w\xi}$	= Non-conservative wave force

Continuity

$$\frac{\partial}{\partial t}\left(\frac{H_z}{mn}\right)+\frac{\partial}{\partial\xi}\left(\frac{H_z u^l}{n}\right)+\frac{\partial}{\partial\eta}\left(\frac{H_z v^l}{m}\right)+\frac{\partial}{\partial s}\left(\frac{\omega_s^l}{mn}\right)=0$$

xi-direction Momentum Balance

$$\underbrace{\frac{\partial}{\partial t}\left(\frac{H_z}{mn}u\right)}_{\text{ACC}}+\underbrace{\frac{\partial}{\partial\xi}\left(\frac{H_z u}{n}u\right)+\frac{\partial}{\partial\eta}\left(\frac{H_z v}{m}u\right)+u\frac{\partial}{\partial\xi}\left(\frac{H_z u^{St}}{n}\right)+u\frac{\partial}{\partial\eta}\left(\frac{H_z v^{St}}{m}\right)}_{\text{HA}}$$

$$+\underbrace{\frac{\partial}{\partial s}\left(\frac{\omega_s}{mn}u\right)+u\frac{\partial}{\partial s}\left(\frac{\omega_s^{St}}{mn}\right)}_{\text{VA}}\underbrace{-H_z\left(\frac{fv}{mn}\right)}_{\text{COR}}\underbrace{-H_z\left(\frac{fv^{st}}{mn}\right)}_{\text{StCOR}}=\underbrace{-\frac{H_z}{n}\frac{\partial\varphi^c}{\partial\xi}\bigg|_z}_{\text{PG}}+\underbrace{H_z v^{st}\left(\frac{1}{n}\frac{\partial v}{\partial\xi}-\frac{1}{m}\frac{\partial u}{\partial\eta}\right)-\omega_s^{St}\frac{\partial}{\partial s}\left(\frac{u}{mn}\right)}_{\text{HVF}}$$

wave breaking

$$+\underbrace{\frac{H_z\mathcal{F}^{\xi}}{mn}}_{\text{BF}}+\underbrace{\frac{H_z\mathcal{F}^{w\xi}}{mn}}_{\text{BA+RA+BtSt+SuSt}}+\underbrace{\frac{H_z D^{\xi}}{mn}}_{\text{HM}}\underbrace{-\frac{\partial}{\partial s}\left(\overline{u'w'}-\frac{\nu}{H_z}\frac{\partial u}{\partial s}\right)}_{\text{VM}}+\underbrace{\hat{\mathcal{F}}^u}_{\text{FCurv}}$$

eta-Direction Momentum Balance

$$\underbrace{\frac{\partial}{\partial t}\left(\frac{H_z}{mn}v\right)}_{\text{ACC}}+\underbrace{\frac{\partial}{\partial \xi}\left(\frac{H_z u}{n}v\right)+\frac{\partial}{\partial \eta}\left(\frac{H_z v}{m}v\right)+v\frac{\partial}{\partial \xi}\left(\frac{H_z u^{St}}{n}\right)+v\frac{\partial}{\partial \eta}\left(\frac{H_z v^{St}}{m}\right)}_{\text{HA}}$$

$$+\underbrace{\frac{\partial}{\partial s}\left(\frac{\omega_s}{mn}v\right)+v\frac{\partial}{\partial s}\left(\frac{\omega_s^{St}}{mn}\right)}_{\text{VA}}+\underbrace{H_z\left(\frac{fu}{mn}\right)}_{\text{COR}}+\underbrace{H_z\left(\frac{fu^{st}}{mn}\right)}_{\text{StCOR}} = \underbrace{-\frac{H_z}{m}\frac{\partial \varphi^c}{\partial \eta}\bigg|_z}_{\text{PG}}\underbrace{-H_z u^{st}\left(\frac{1}{n}\frac{\partial v}{\partial \xi}-\frac{1}{m}\frac{\partial u}{\partial \eta}\right)-\omega_s^{St}\frac{\partial}{\partial s}\left(\frac{v}{mn}\right)}_{\text{HVF}}$$

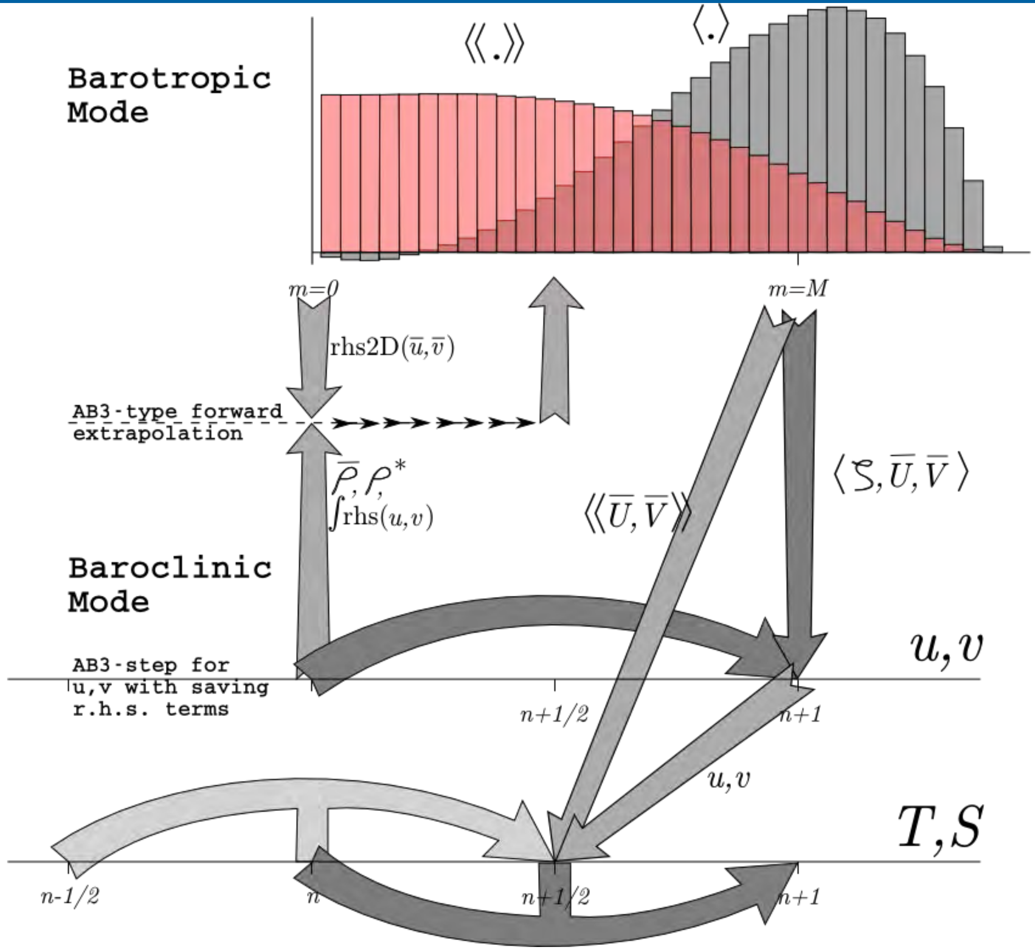
$$\underbrace{+\frac{H_z\mathcal{F}^{\eta}}{mn}}_{\text{BF}}+\underbrace{\frac{H_z\mathcal{F}^{w\eta}}{mn}}_{\text{BA+RA+BtSt+SuSt}}\underbrace{+\frac{H_z D^{\eta}}{mn}}_{\text{HM}}\underbrace{-\frac{\partial}{\partial s}\left(\overline{v'w'}-\frac{v}{H_z}\frac{\partial v}{\partial s}\right)}_{\text{VM}}\underbrace{+\hat{\mathcal{F}}^{v}}_{\text{FCurv}}$$

Description of Terms

ACC	= Local Acceleration
HA	= Horizontal Advection
VA	= Vertical Advection
COR	= Coriolis Force
StCOR	= Stokes-Coriolis Force
PG	= Pressure Gradient
HVF	= Horizontal Vortex Force
BF	= Body Force
BA+RA+BuSt+SuSt	= Breaking Acceleration+ Roller Acceleration+ Bottom Streaming+ Surface Streaming
HM	= Horizontal Mixing
VM	= Vertical Mixing
Fcurv	= Curvilinear terms

Kumar, N., Voulgaris, G., Warner, J.C., and M., Olabarrieta (2012). Implementation of a vortex force formalism in a coupled modeling system for inner-shelf and surf-zone applications. Ocean Modelling, 47, 65-95.

Solution techniques- mode splitting



Shchepetkin, A. F., and J. C. McWilliams, 2008: Computational kernel algorithms for fine-scale, multi-process, long-time oceanic simulations. In: *Handbook of Numerical Analysis: Computational Methods for the Ocean and the Atmosphere*, eds. R. Temam & J. Tribbia, Elsevier Science, ISBN-10: 0444518932, ISBN-13: 978-0444518934.

USGS

Numerical algorithms

$$\frac{\partial}{\partial t}\frac{H_z C}{mn} = -\frac{\partial}{\partial \xi}F^\xi - \frac{\partial}{\partial \eta}F^\eta - \frac{\partial}{\partial \sigma}F^\sigma$$

Advection schemes

- 2nd order centered

$$F^\xi = \frac{\overline{H_z}^\xi u \overline{C}^\xi}{\overline{n}^\xi}$$
$$F^\eta = \frac{\overline{H_z}^\eta v \overline{C}^\eta}{\overline{m}^\eta}$$
$$F^\sigma = \frac{\overline{H_z}^\sigma \Omega \overline{C}^\sigma}{mn}.$$

- 4th order centered

- 4th order Akima

- TSU3 3rd order

- MPDATA

- HSIMT

many choices, see manual for details. ……

USGS

24分35秒〜

How do we select different schemes

- c pre-processor definitions (list them in your project *.h “header” file)
- during compilation, .F → .f90s
- compiles f90's into objects
- compiles objects to libs
- ar the libs to make 1 exe   ライブラリから1つの実行形式を作る
- for coupling, it makes wrf, roms, and swan as libs, then pull them together for coupling and only produces one exe → coawstM [.exe]

Build errors

無視するエラー

- **During the build, the error "can not make wrf.exe" or 'undefined reference to MAIN" is ok.**

libdev/frtl/src/libfor/for_main.c:(.text+0x2a): undefined reference to `MAIN__'

make[2]: [em_wrf] Error 1 (ignored)

This "Error" is ok.

If you get other errors, do this:

scrip build.txt script コマンドで作業を記録する　ファイル名を指定しない場合は typescript が作成

./coawst.bash exitで記録終了

exit

and that will create a file build.txt

Edit that file and search for the word "error"

USGS

COAWST specific cpp options

29分～

You need to use at least 1 model of the 3:

- **#define ROMS_MODEL** — **if you want to use the ROMS model**
- **#define SWAN_MODEL** — **if you want to use the SWAN model**
- **#define WRF_MODEL** — **if you want to use the WRF model**

- **#define MCT_LIB** — **if you have more than one model selected**

Interpolation

Use these 3 INTERP calls if the models are on different grids.

- **#define MCT_INTERP_WV2AT** — **allows grid interpolation between the wave and atmosphere models**
- **#define MCT_INTERP_OC2AT** — **allows grid interpolation between the ocean and atmosphere models**
- **#define MCT_INTERP_OC2WV** — **allows grid interpolation between the ocean and wave models**

- **#define NESTING** — **allows grid refinement in roms or in swan**
  For now, I suggest if you couple ROMS+SWAN and use NESTING, then use the same grids for R+S.

USGS

COAWST specific cpp options

atmosphere ocean

If you activate ROMS + WRF, then pick one of these:

- **#define ATM2OCN_FLUXES** — **provide consistent fluxes between atm and ocn. WRF send USTRESS,VSTRESS, LH, HFX, GSW, GLW**

おすすめ

--- OR --- **If you don’t use this, then**

- **#define BULK_FLUXES** — **this will send Uwind, Vwind, Patm, RH, Tair, cloud, GSW, GLW**

Related: E - P (Evaporation - Presipitation)

- **#define EMINUSP** — **send EVAP and RAIN from WRF to ROMS**
- **#define CLOUDS** — **send cloud fraction from WRF to ROMS**
- **#define ATM_PRESS** — **send MSLP (Patm) from WRF to ROMS**

These 3 options use SWAN wave data for computation of ocean surface stress in ROMS bulk_fluxes and in WRF myjsfc and mynn surface layer schemes :

- **#define COARE_TAYLOR_YELLAND** — **wave enhanced roughness (swan to roms or wrf)**
- **#define COARE_OOST** — **wave enhanced roughness (swan to roms or wrf)**
- **#define DRENNAN** — **wave enhanced roughness (swan to roms or wrf)**

- **#define DRAGLIM_DAVIS** — **feature added to WRF and SWAN to limit the ocean roughness drag to be a maximum of 2.85E-3**

USGS

COAWST specific cpp options

If you couple ROMS + WRF then you can select this for testing:

- **#define SST_CONST** — **do not allow SST from ROMS to affect WRF**

Wave Options

- **#define UV_CONST** — **send vel = 0 from the ocn to wave model**
- **#define ZETA_CONST** — **send zeta = 0 from the ocn to wave model**
- **#define UV_KIRBY** — **compute "depth-avg" current based on Hwave to be sent from the ocn to the wav model for coupling**
- **#define WEC_MELLOR** — **radiation stress terms from Mellor 08**
- **#define WEC_VF** — **wave-current stresses from Uchiyama et al.**
- **#define WDISS_THORGUZA** — **wave dissipation from Thorton/Guza**
- **#define WDISS_CHURTHOR** — **wave dissipation from Church/Thorton**
- **#define WDISS_WAVEMOD** — **wave dissipation from a wave model**
- **#define WDISS_INWAVE** — **wave dissipation from a InWave model**
- **#define ROLLER_SVENDSEN** — **wave roller based on Svendsen**
- **#define ROLLER_MONO** — **wave roller for monochromatic waves**
- **#define ROLLER_RENIERS** — **wave roller based on Reniers**
- **#define BOTTOM_STREAMING** — **wave enhanced bottom streaming**
- **#define SURFACE_STREAMING** — **wave enhanced surface streaming**

USGS

COAWST specific cpp options

38分20秒~

Wave Options

- #define CHARNOK — Charnok surface roughness from wind stress
- #define CRAIG_BANNER — Craig and Banner wave breaking surface flux
- --- or ---
- #define ZOS_HSIG — surface roughness from wave amplitude
- #define TKE_WAVEDISS — wave breaking surface flux from wave amplitude

Vegetation Options

- # define VEGETATION — Activate vegetation module
- # define VEG_DRAG — Drag terms Luhar M. et.al (2011)
- # define VEG_FLEX — Flexible vegetation terms
- # define VEG_TURB — Turbulence terms, Uittenbogaard R. (2003)
- # define VEG_SWAN_COUPLING — Exchange of VEG data btwn. ROMS and SWAN
- # define VEG_STREAMING — Wave streaming effects
- # define MARSH_WAVE_THRUST — Wave thrust on marshes, Tonelli, M. et al. (2010)

Tracer Advection Option

- # define TS_HSIMT — Positive definite tracer advection (Wu and Zhu, OM 2010)

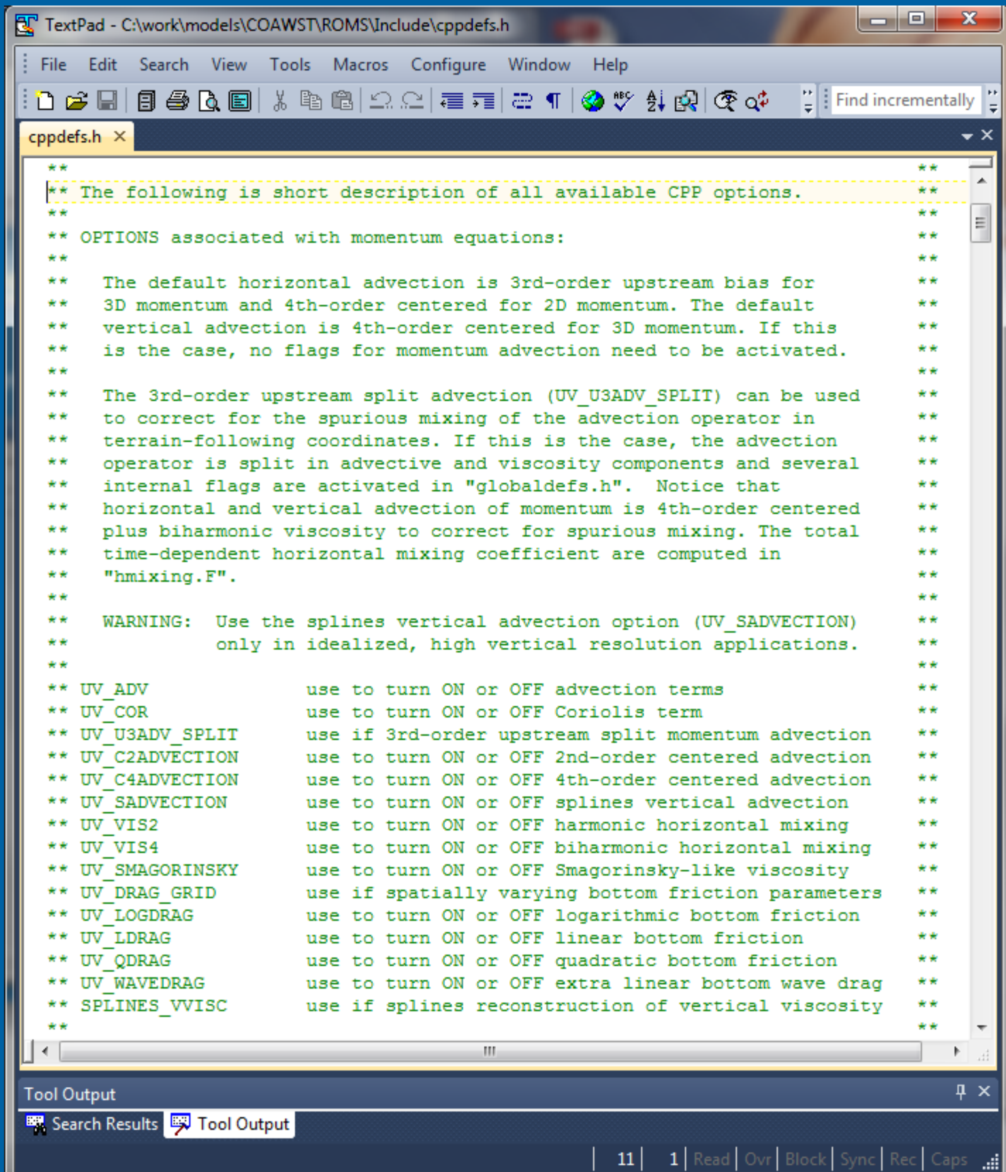
USGS

+ ROMS Specific CPP options …..

ROMS/Include/cppdefs.h

40分 ~

See cppdefs for a complete list of ROMS options

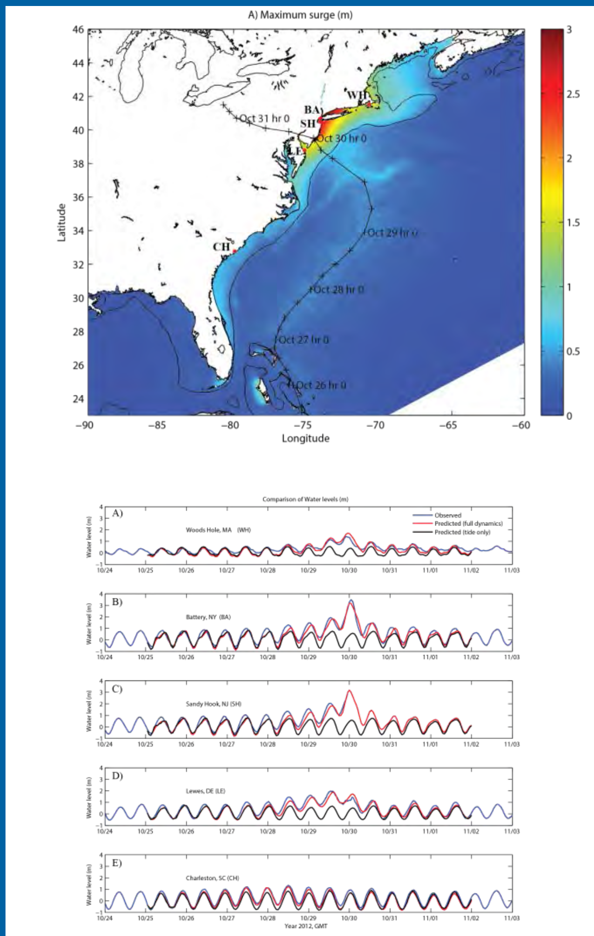


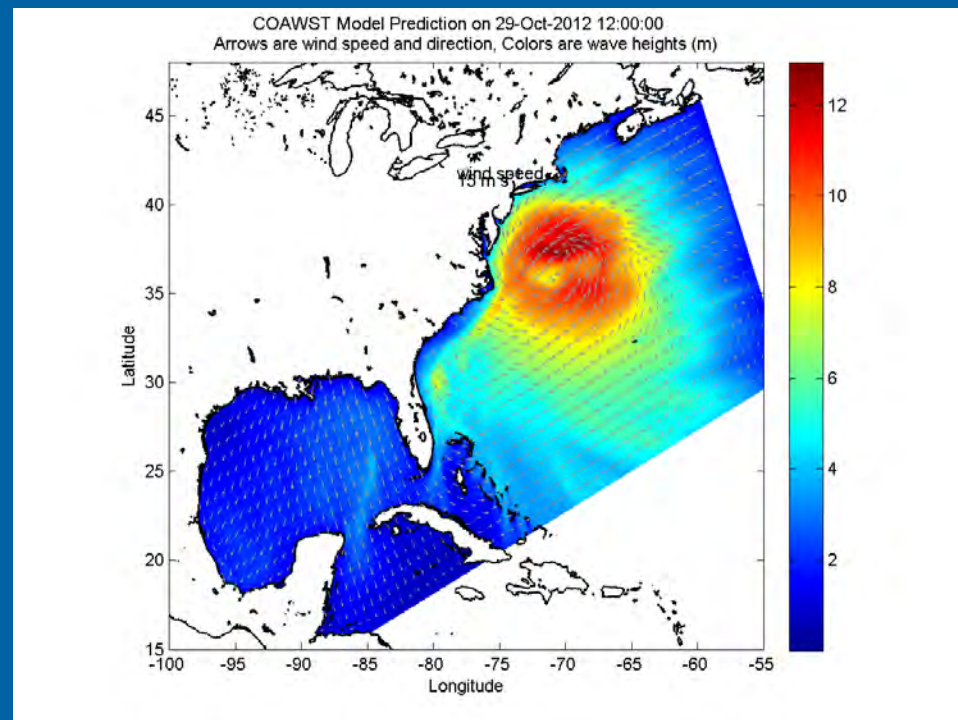

```
**                                                                        **
** The following is short description of all available CPP options.      **
**                                                                        **
** OPTIONS associated with momentum equations:                           **
**                                                                        **
**   The default horizontal advection is 3rd-order upstream bias for     **
**   3D momentum and 4th-order centered for 2D momentum. The default     **
**   vertical advection is 4th-order centered for 3D momentum. If this   **
**   is the case, no flags for momentum advection need to be activated.  **
**                                                                        **
**   The 3rd-order upstream split advection (UV_U3ADV_SPLIT) can be used **
**   to correct for the spurious mixing of the advection operator in     **
**   terrain-following coordinates. If this is the case, the advection   **
**   operator is split in advective and viscosity components and several **
**   internal flags are activated in "globaldefs.h".  Notice that        **
**   horizontal and vertical advection of momentum is 4th-order centered **
**   plus biharmonic viscosity to correct for spurious mixing. The total **
**   time-dependent horizontal mixing coefficient are computed in        **
**   "hmixing.F".                                                         **
**                                                                        **
**   WARNING:  Use the splines vertical advection option (UV_SADVECTION) **
**             only in idealized, high vertical resolution applications. **
**                                                                        **
** UV_ADV              use to turn ON or OFF advection terms             **
** UV_COR              use to turn ON or OFF Coriolis term               **
** UV_U3ADV_SPLIT      use if 3rd-order upstream split momentum advection **
** UV_C2ADVECTION      use to turn ON or OFF 2nd-order centered advection **
** UV_C4ADVECTION      use to turn ON or OFF 4th-order centered advection **
** UV_SADVECTION       use to turn ON or OFF splines vertical advection  **
** UV_VIS2             use to turn ON or OFF harmonic horizontal mixing  **
** UV_VIS4             use to turn ON or OFF biharmonic horizontal mixing **
** UV_SMAGORINSKY      use to turn ON or OFF Smagorinsky-like viscosity  **
** UV_DRAG_GRID        use if spatially varying bottom friction parameters **
** UV_LOGDRAG          use to turn ON or OFF logarithmic bottom friction **
** UV_LDRAG            use to turn ON or OFF linear bottom friction      **
** UV_QDRAG            use to turn ON or OFF quadratic bottom friction   **
** UV_WAVEDRAG         use to turn ON or OFF extra linear bottom wave drag **
** SPLINES_VVISC       use if splines reconstruction of vertical viscosity **
**                                                                        **
```

This list goes on for many many more pages …..

42分～

Projects/Sandy Application





USGS

Steps to create ROMS application

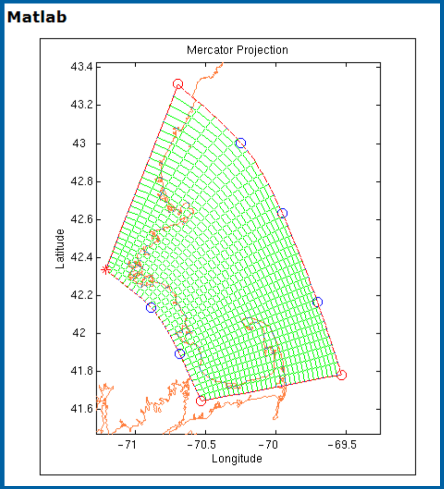
1) parent grid
2) masking
3) bathymetry
4) child grid
5) 3D: BC's (u,v,temp,salt), init, and climatology
6) 2D: BC's (ubar, vbar, zeta) = tides
7) Surface forcing (heat and momentum fluxes)
8) ROMS input files: sandy.h and ocean_sandy.in
9) Build and run: coawst.bash
10) Output

USGS

These steps will follow the User Manual Section 9

44分30秒~

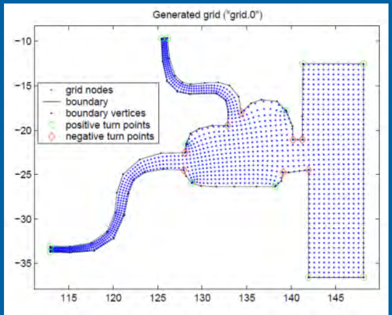
1) Grid generation tools



- **Seagrid - matlab**

http://woodshole.er.usgs.gov/operations/modeling/seagrid/

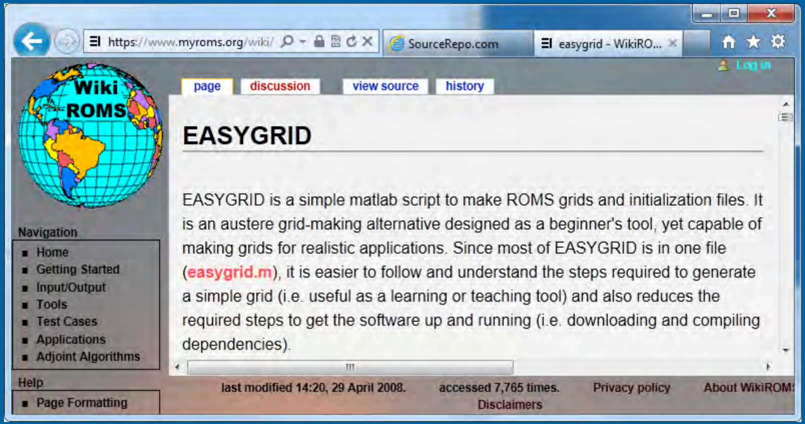
(needs unsupported netcdf interface)



- **gridgen - command line**

http://code.google.com/p/gridgen-c/

- **EASYGRID**

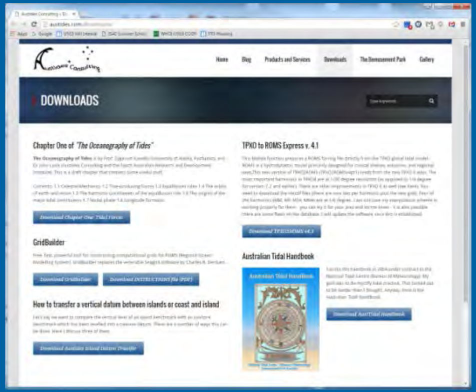


USGS

1) Grid generation tools

- **Austides - Gridbuilder**

http://austides.com/downloads/



- **COAWST/Tools/mfiles/mtools/wrf2roms _mw.m**

function wrf2roms_mw(theWRFFile, theROMSFile)

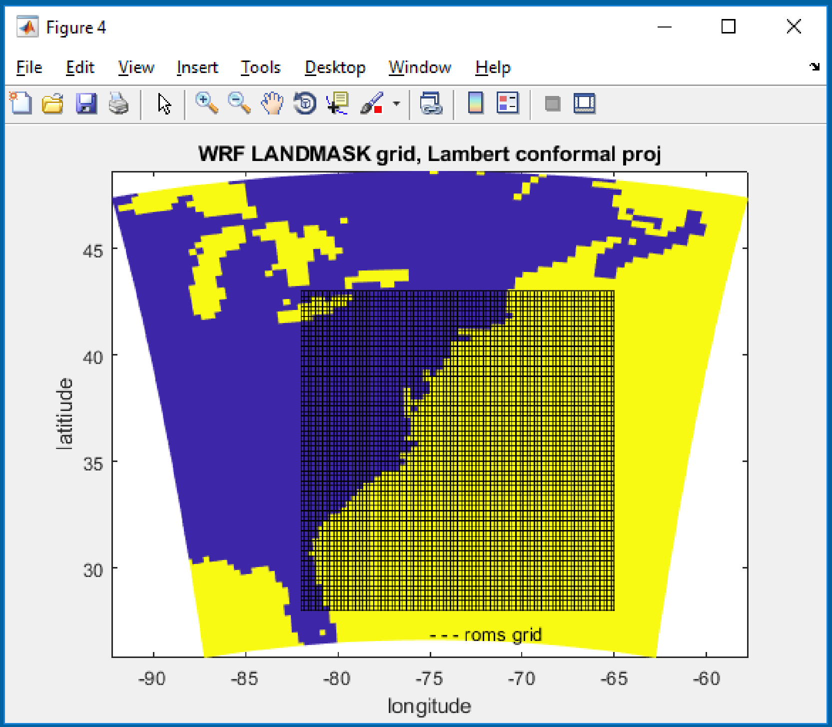
Generates a ROMS grid from a WRF grid.

- **COAWST/Tools/mfiles/mtools/create_roms_xygrid.m**

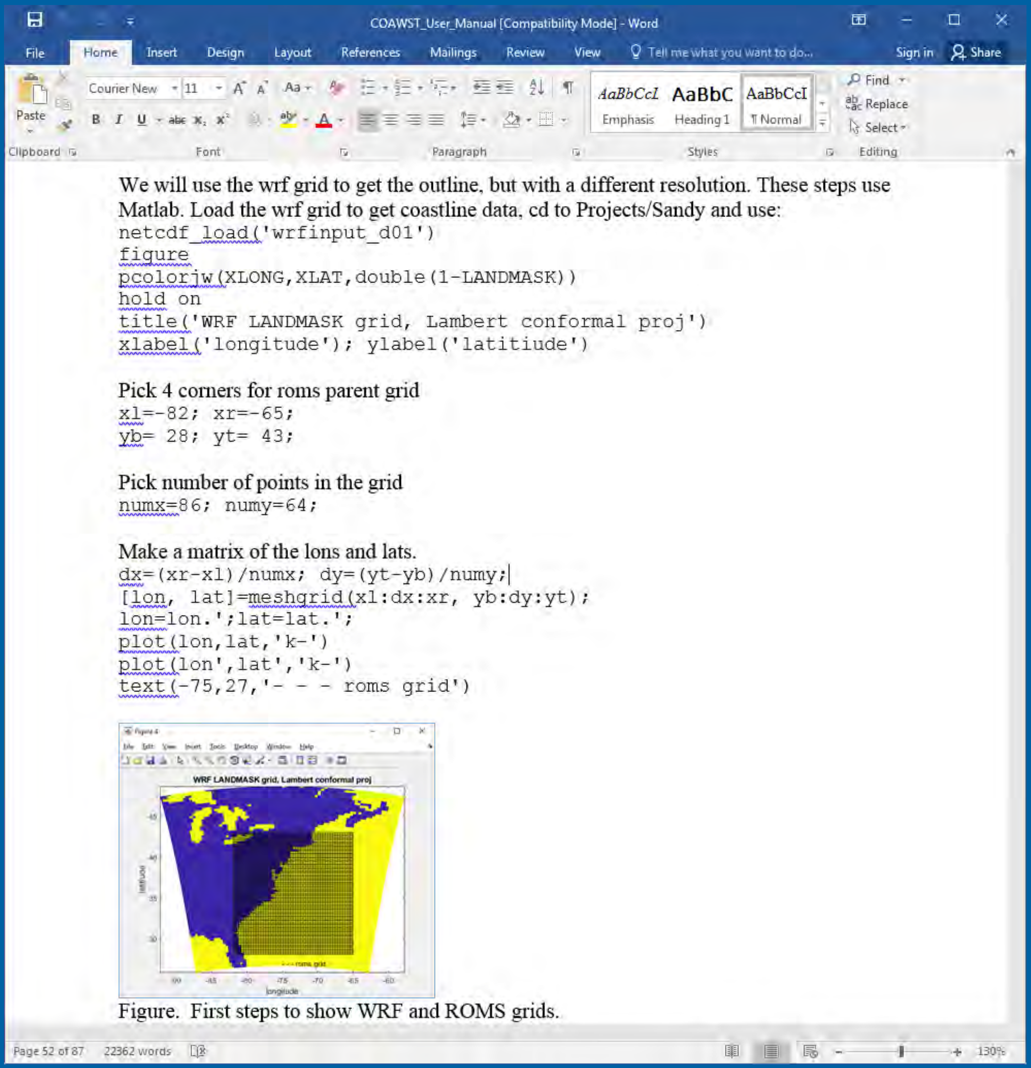
intended for simple rectilinear grids

- **or any other method that you know**

1) Grid generation tools – matlab



select 4 corners



COAWST_User_Manual [Compatibility Mode] - Word

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

```
netcdf_load('wrfinput_d01')
figure
pcolorjw(XLONG,XLAT,double(1-LANDMASK))
hold on
title('WRF LANDMASK grid, Lambert conformal proj')
xlabel('longitude'); ylabel('latitude')
```

Pick 4 corners for roms parent grid

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

Pick number of points in the grid

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

Make a matrix of the lons and lats.

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

Figure. First steps to show WRF and ROMS grids.

Page 52 of 87 22362 words

1) Grid generation tools

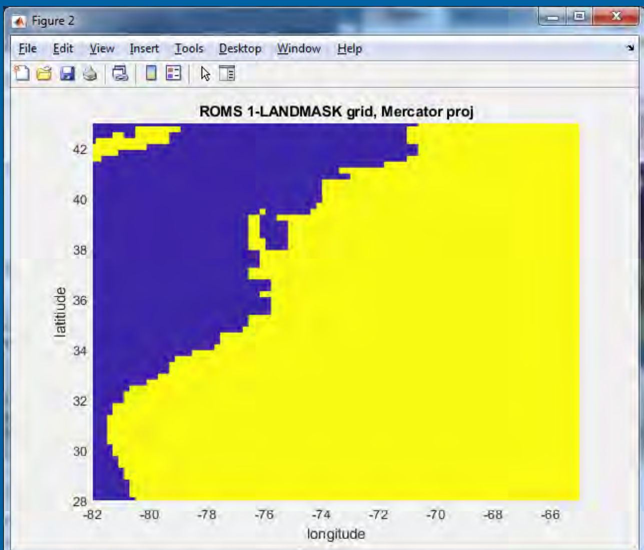
Call generic grid creation.

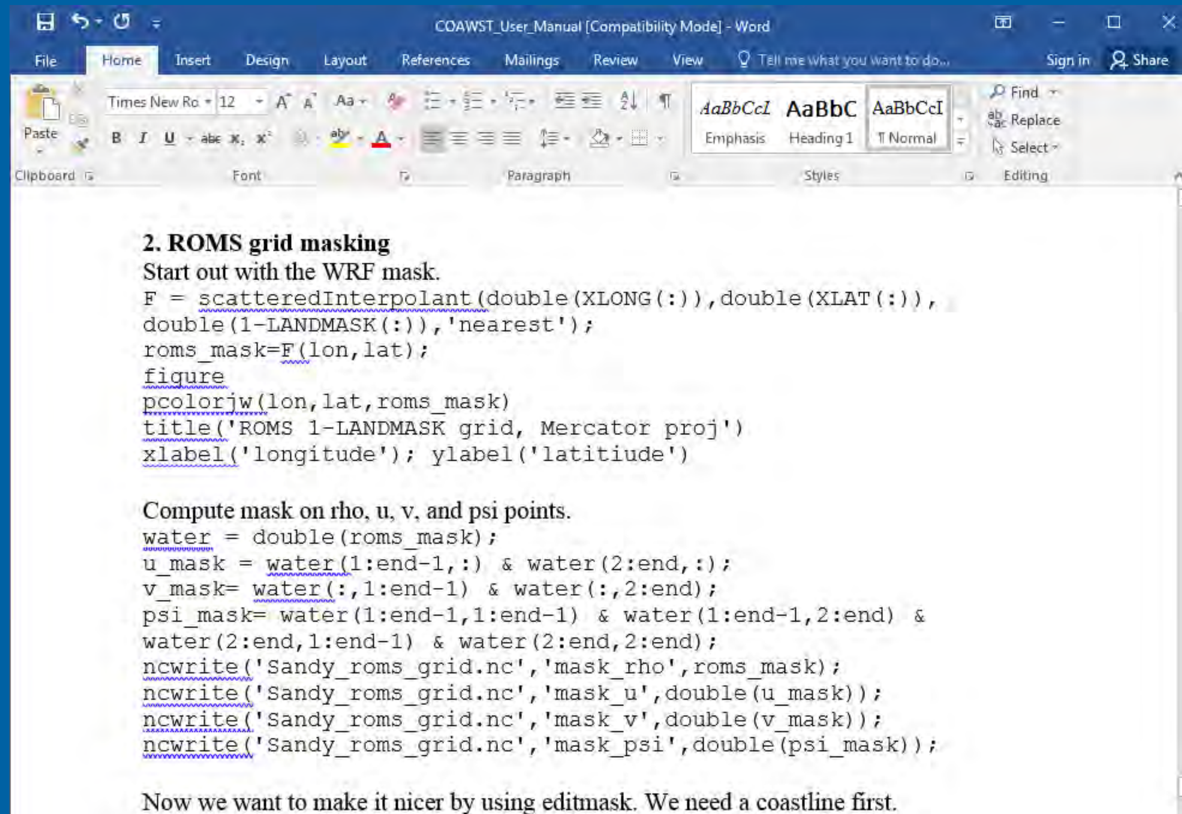
```
roms_grid='Sandy_roms_grid.nc';
rho.lat=lat; rho.lon=lon;
rho.depth=zeros(size(rho.lon))+100;  % for now just make zeros
rho.mask=zeros(size(rho.lon));       % for now just make zeros
spherical='T';
%projection='lambert conformal conic';
projection='mercator';
save temp_jcw33.mat rho spherical projection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

This step creates a roms grid file called ‘Sandy_roms_grid.nc’

USGS

2) masking – first base it on WRF





2. ROMS grid masking

Start out with the WRF mask.

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

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

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

Now we want to make it nicer by using editmask. We need a coastline first.

2) Masking – get coastline

May also need a coastline, can obtain this here:

http://www.ngdc.noaa.gov/mgg/coast/

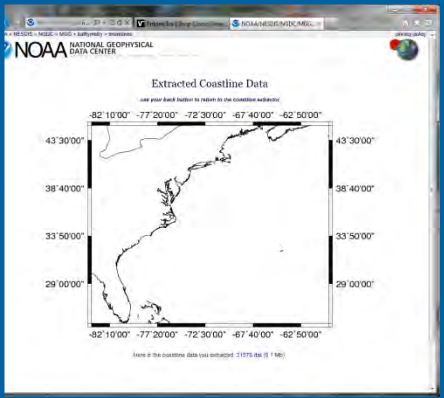
45

-84

-60

25

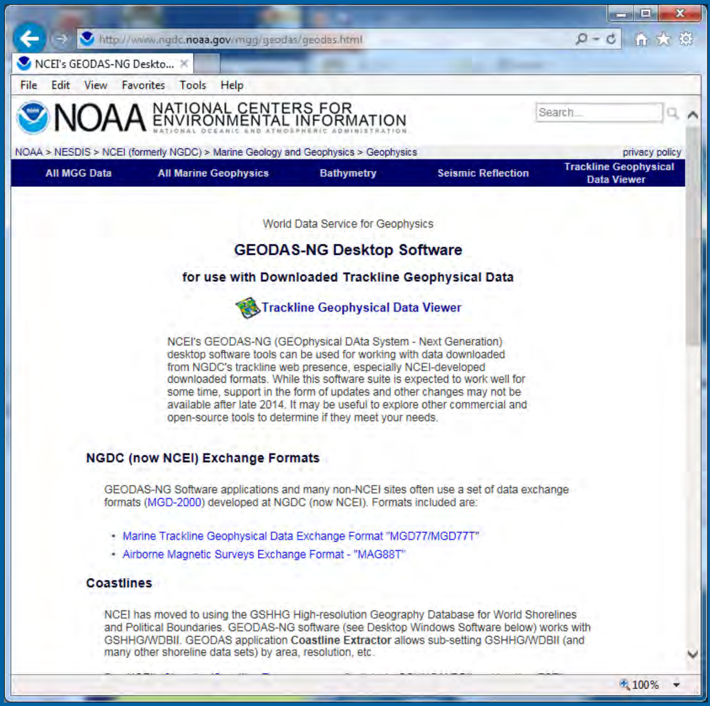
No longer active.



lon=coastline(:,1);

lat=coastline(:,2);

save coastline.mat lon lat



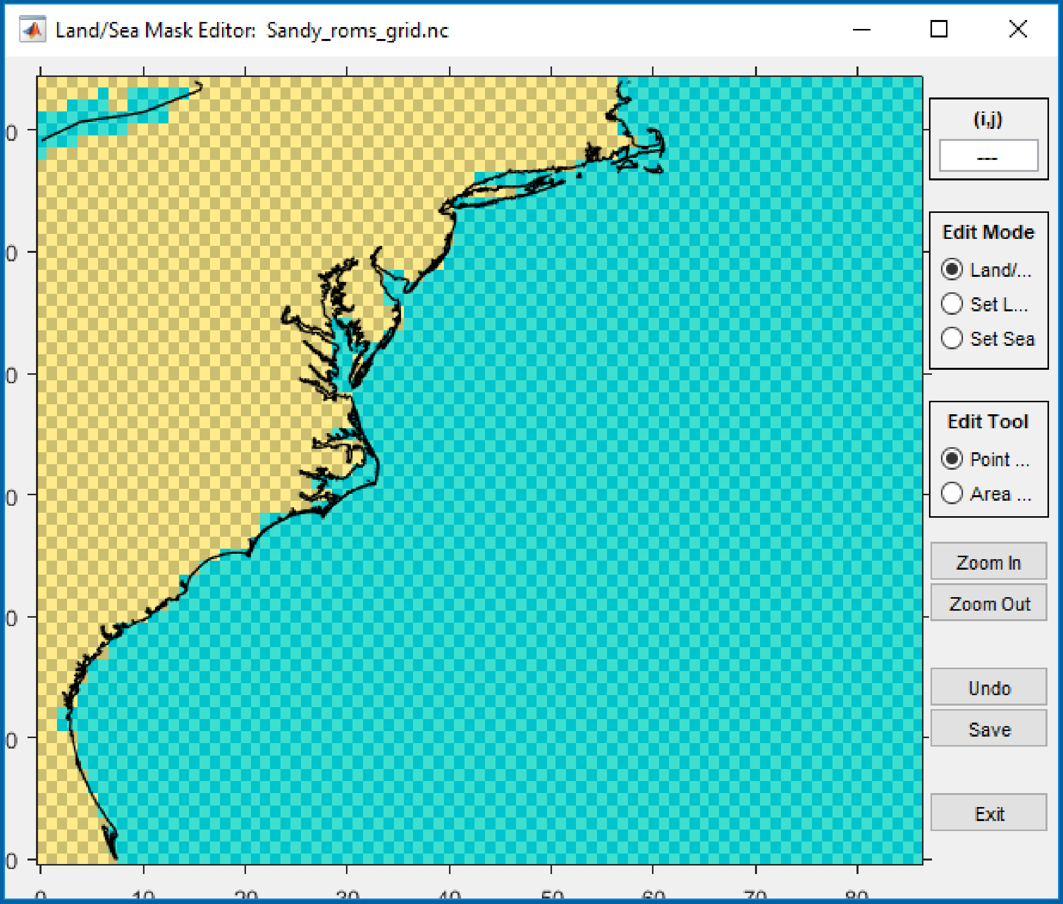
http://www.ngdc.noaa.gov/mgg/~~geodas/geodas.html~~

shorelines/

2) Masking

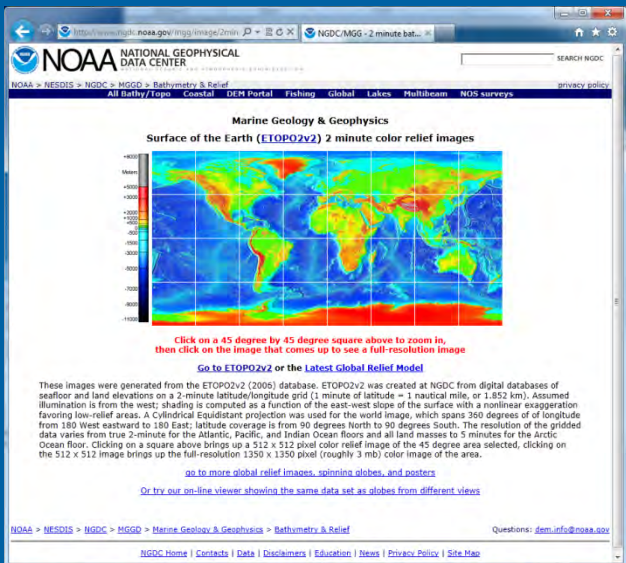
use COAWST/Tools/mfiles/rutgers/landmask/editmask m file

editmask(‘Sandy_roms_grid.nc','coastline.mat')

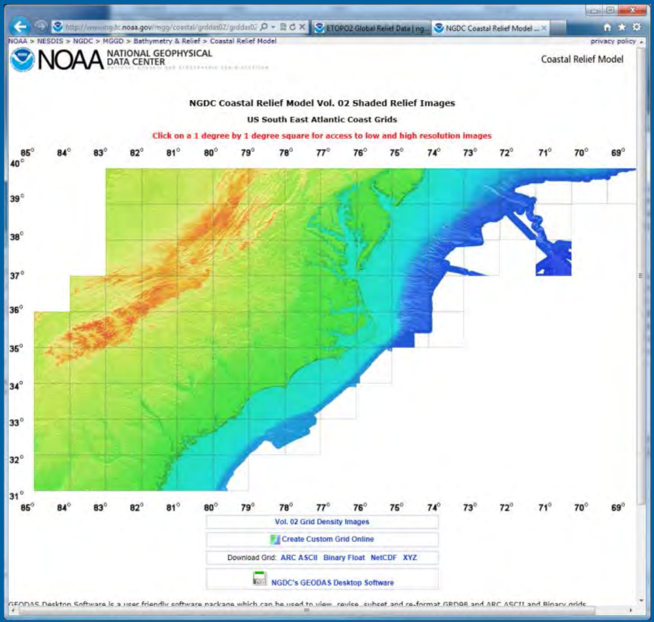


3) bathymetry

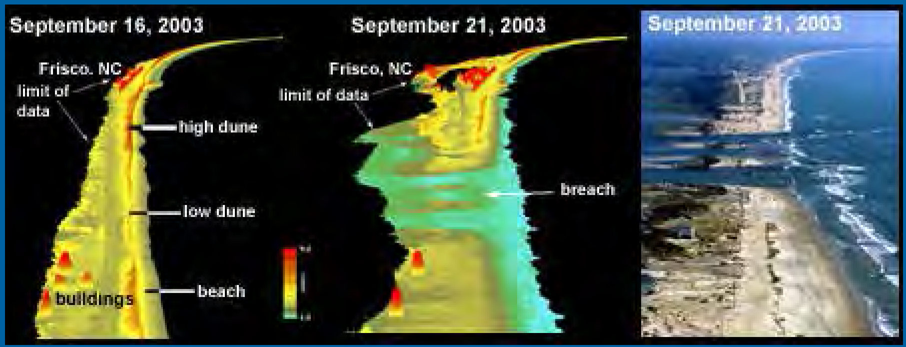
many sources



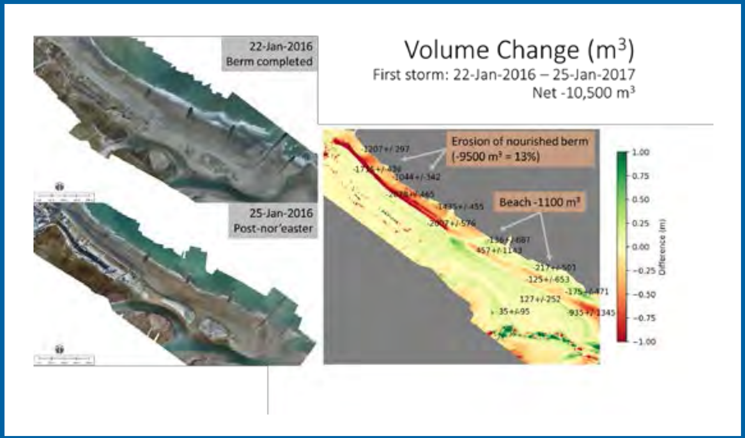
ETOPO2



Coastal Relief Model

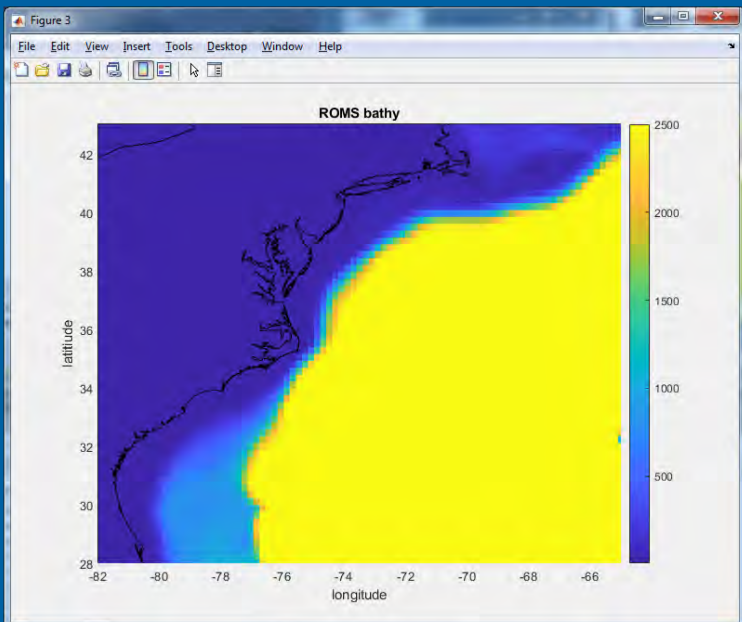


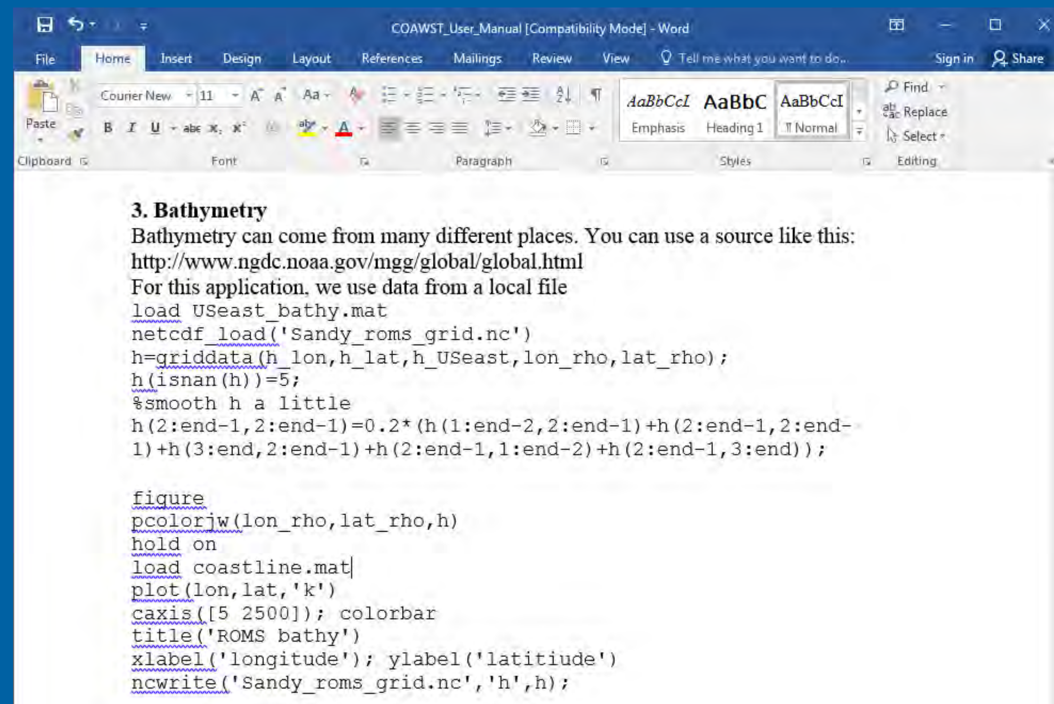
LIDAR



SFM structure from motion

3) bathymetry





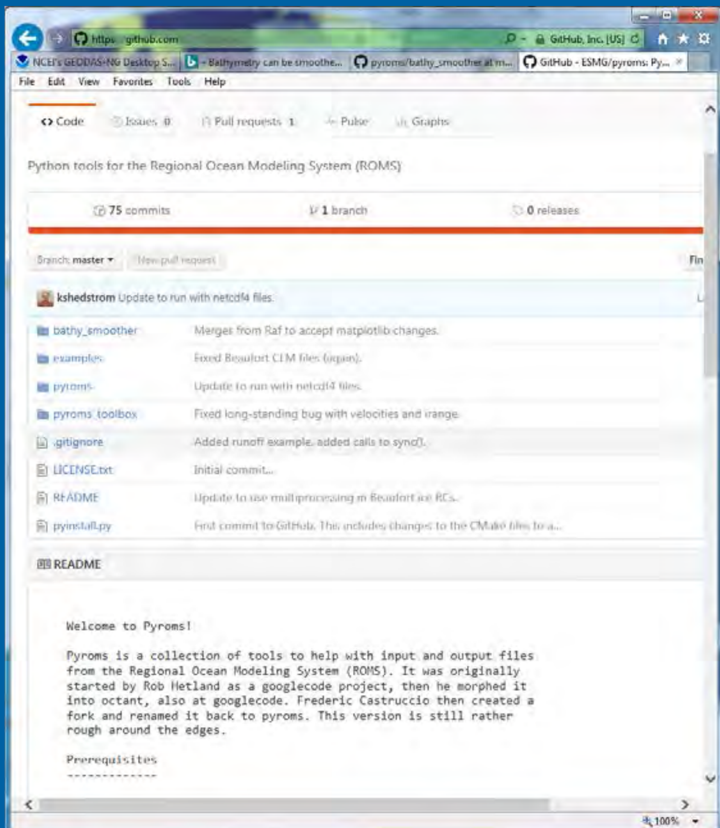
3. Bathymetry

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

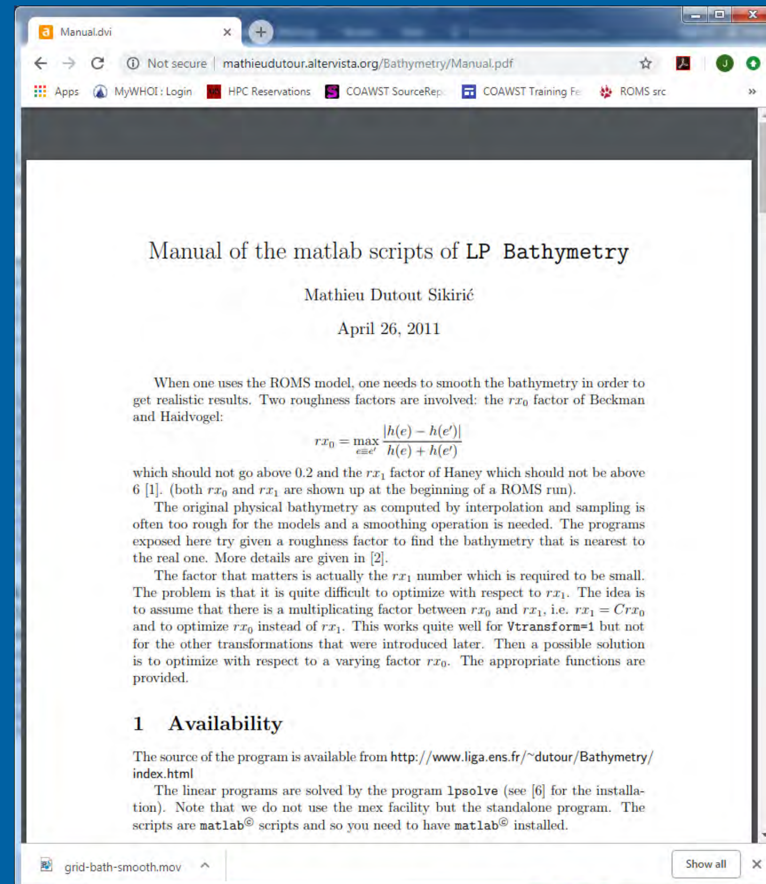
```
load USeast_bathy.mat
netcdf_load('Sandy_roms_grid.nc')
h=griddata(h_lon,h_lat,h_USeast,lon_rho,lat_rho);
h(isnan(h))=5;
%smooth h a little
h(2:end-1,2:end-1)=0.2*(h(1:end-2,2:end-1)+h(2:end-1,2:end-
1)+h(3:end,2:end-1)+h(2:end-1,1:end-2)+h(2:end-1,3:end));

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

USGS

3) bathymetry



Bathy smoothing:
https://github.com/ESMG/pyroms



LP Bathy

USGS

Grid Parameters

Beckman & Haidvogel number (1993)

$$r_{xo} = \max\left(\frac{\Delta h}{2\bar{h}}\right) = \max\left(\frac{|h_i - h_{i-1}|}{h_i + h_{i-1})}\right)$$

should be < 0.2 but can be fine up to ~ 0.4

determined only by smoothing

Haney number (1991)

$$r_{x1} = \max\left(\frac{z_{i,j,k} - z_{i-1,j,k} + z_{i,j,k-1} - z_{i-1,j,k-1}}{z_{i,j,k} + z_{i-1,j,k} - z_{i,j,k-1} - z_{i-1,j,k-1}}\right)$$

should be < 9 but can be fine up to ~ 16 in some cases

determined by smoothing AND vertical coordinate functions

If these numbers are too large, you will get large pressure gradient errors and Courant number violations and the model will typically blow up right away

4) Child grid

COAWST_User_Manual [Compatibility Mode] - Word

4. Roms child grid

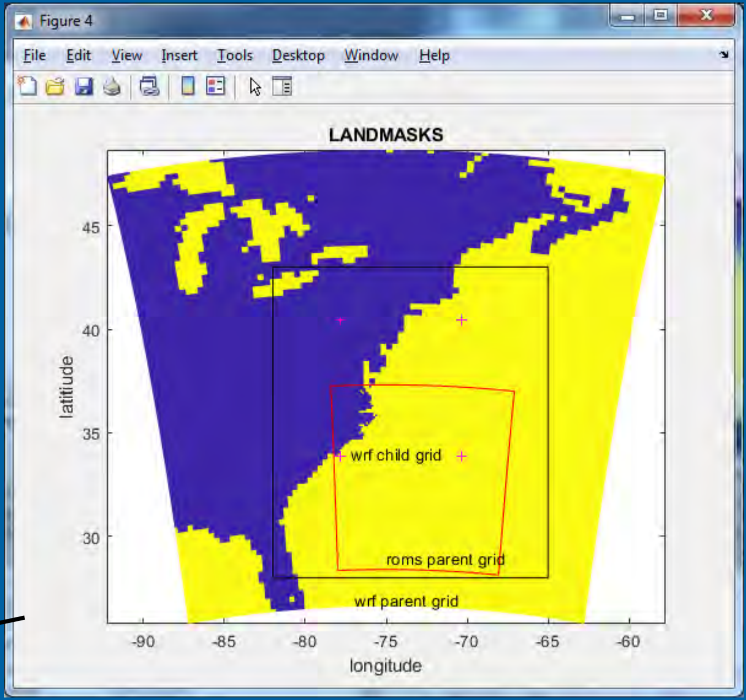
To create a nested child grid, we want to see how the 2WRF and 2 ROMS grids will overlay.

```
netcdf_load('wrfinput_d01')
figure
pcolorjw(XLONG,XLAT,double(1-LANDMASK))
hold on
netcdf_load('wrfinput_d02')
pcolorjw(XLONG,XLAT,double(1-LANDMASK))
plot(XLONG(1,:),XLAT(1,:),'r');
plot(XLONG(end,:),XLAT(end,:),'r')
plot(XLONG(:,1),XLAT(:,1),'r');
plot(XLONG(:,end),XLAT(:,end),'r')
% plot roms parent grid
netcdf_load('Sandy_roms_grid.nc');
plot(lon_rho(1,:),lat_rho(1,:),'k');
plot(lon_rho(end,:),lat_rho(end,:),'k')
plot(lon_rho(:,1),lat_rho(:,1),'k');
plot(lon_rho(:,end),lat_rho(:,end),'k')
text(-75,29,'roms parent grid')
text(-77,27,'wrf parent grid')
text(-77.2,34,'wrf child grid')
title('LANDMASKS')
xlabel('longitude'); ylabel('latitiude')
```

Select child indices and plot location of roms child grid.

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

Page 55 of 87   22362 words



4) Child grid

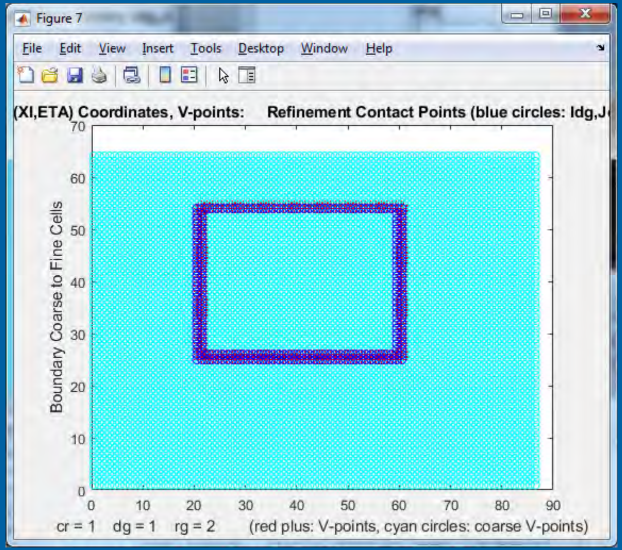
4. Roms child grid

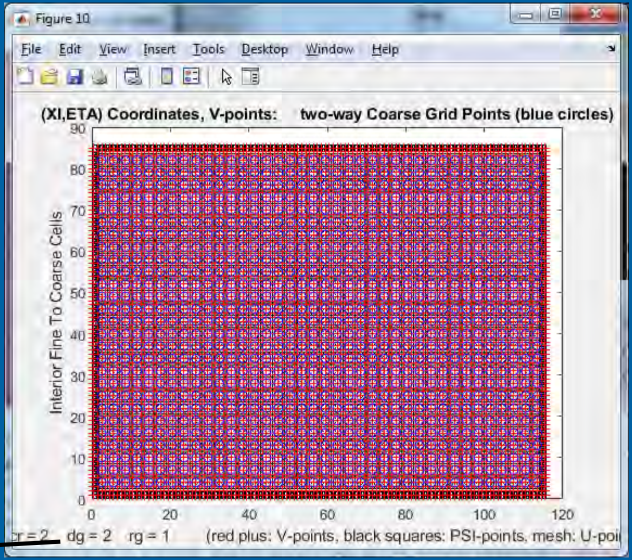
To create a nested child grid, we want to see how the 2WRF and 2 ROMS grids will overlay.

```
netcdf_load('wrfinput_d01')
figure
pcolorjw(XLONG,XLAT,double(1-LANDMASK))
hold on
netcdf_load('wrfinput_d02')
pcolorjw(XLONG,XLAT,double(1-LANDMASK))
plot(XLONG(1,:),XLAT(1,:),'r');
plot(XLONG(end,:),XLAT(end,:),'r')
plot(XLONG(:,1),XLAT(:,1),'r');
plot(XLONG(:,end),XLAT(:,end),'r')
% plot roms parent grid
netcdf_load('Sandy_roms_grid.nc');
plot(lon_rho(1,:),lat_rho(1,:),'k');
plot(lon_rho(end,:),lat_rho(end,:),'k')
plot(lon_rho(:,1),lat_rho(:,1),'k');
plot(lon_rho(:,end),lat_rho(:,end),'k')
text(-75,29,'roms parent grid')
text(-77,27,'wrf parent grid')
text(-77.2,34,'wrf child grid')
title('LANDMASKS')
xlabel('longitude'); ylabel('latitiude')
```

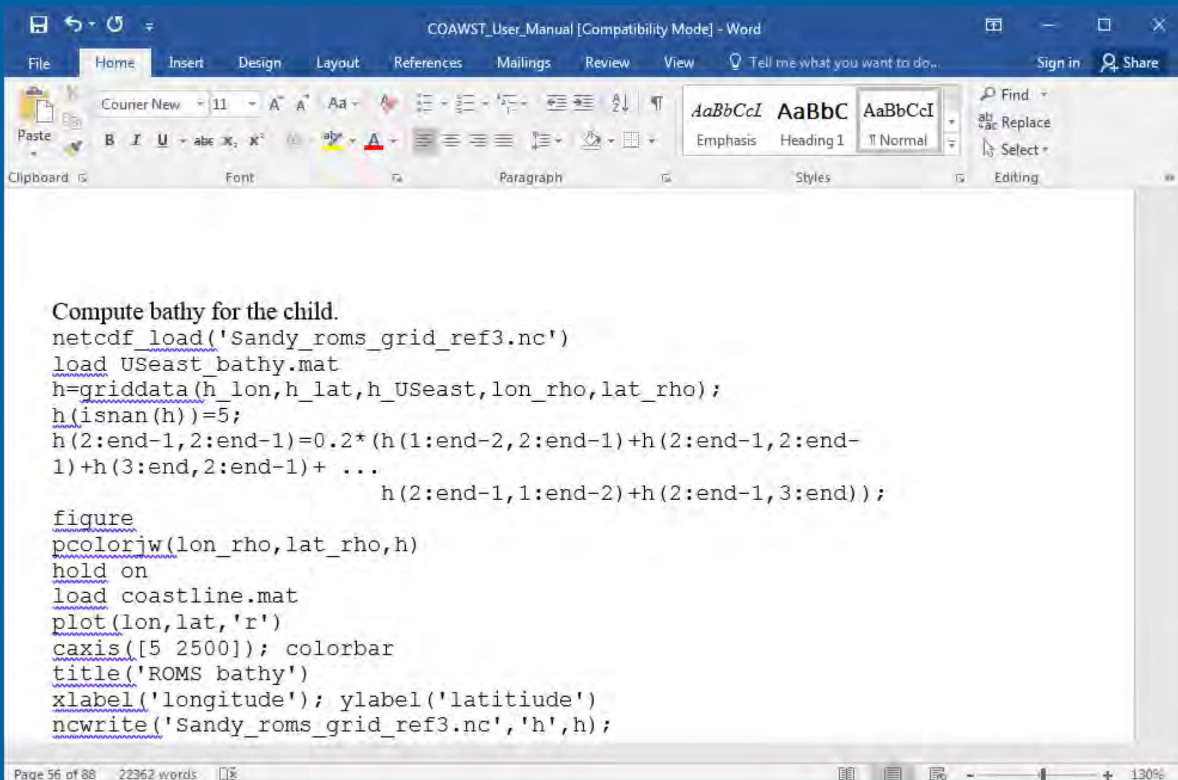
Select child indices and plot location of roms child grid.

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



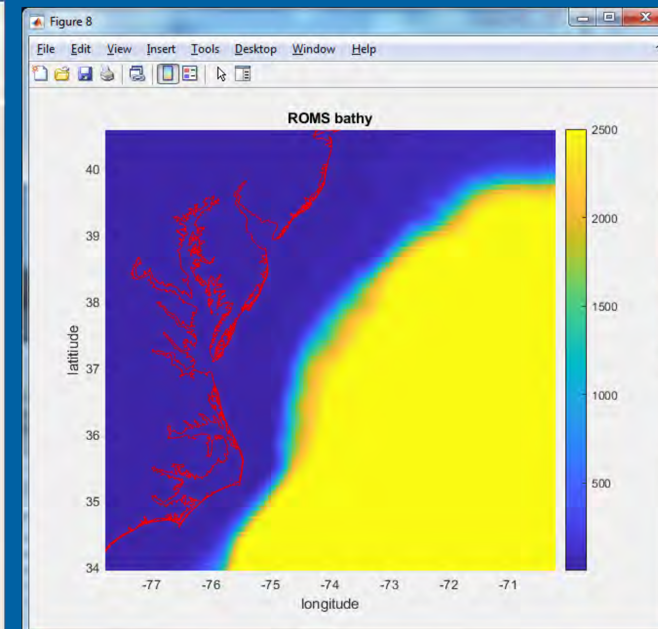


4) Child grid- bathy



Compute bathy for the child.

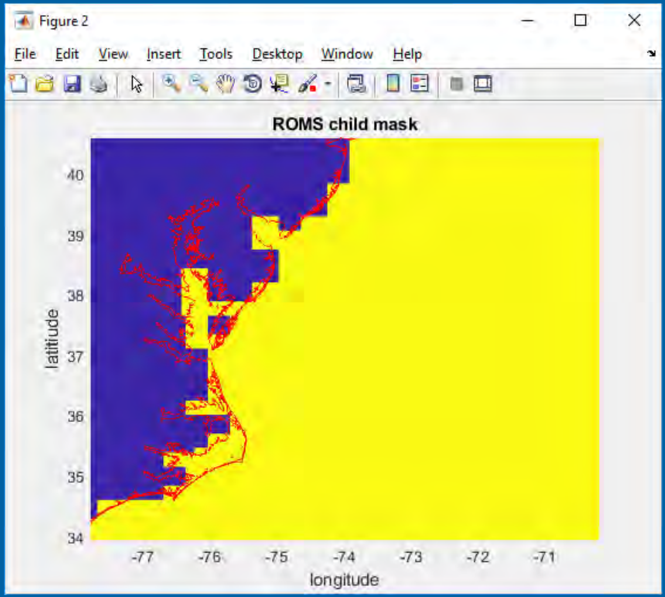
```
netcdf_load('Sandy_roms_grid_ref3.nc')
load USeast_bathy.mat
h=griddata(h_lon,h_lat,h_USeast,lon_rho,lat_rho);
h(isnan(h))=5;
h(2:end-1,2:end-1)=0.2*(h(1:end-2,2:end-1)+h(2:end-1,2:end-
1)+h(3:end,2:end-1)+ ...
                        h(2:end-1,1:end-2)+h(2:end-1,3:end));
figure
pcolorjw(lon_rho,lat_rho,h)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



4) Child grid- masking

Recompute child mask based on WRF mask

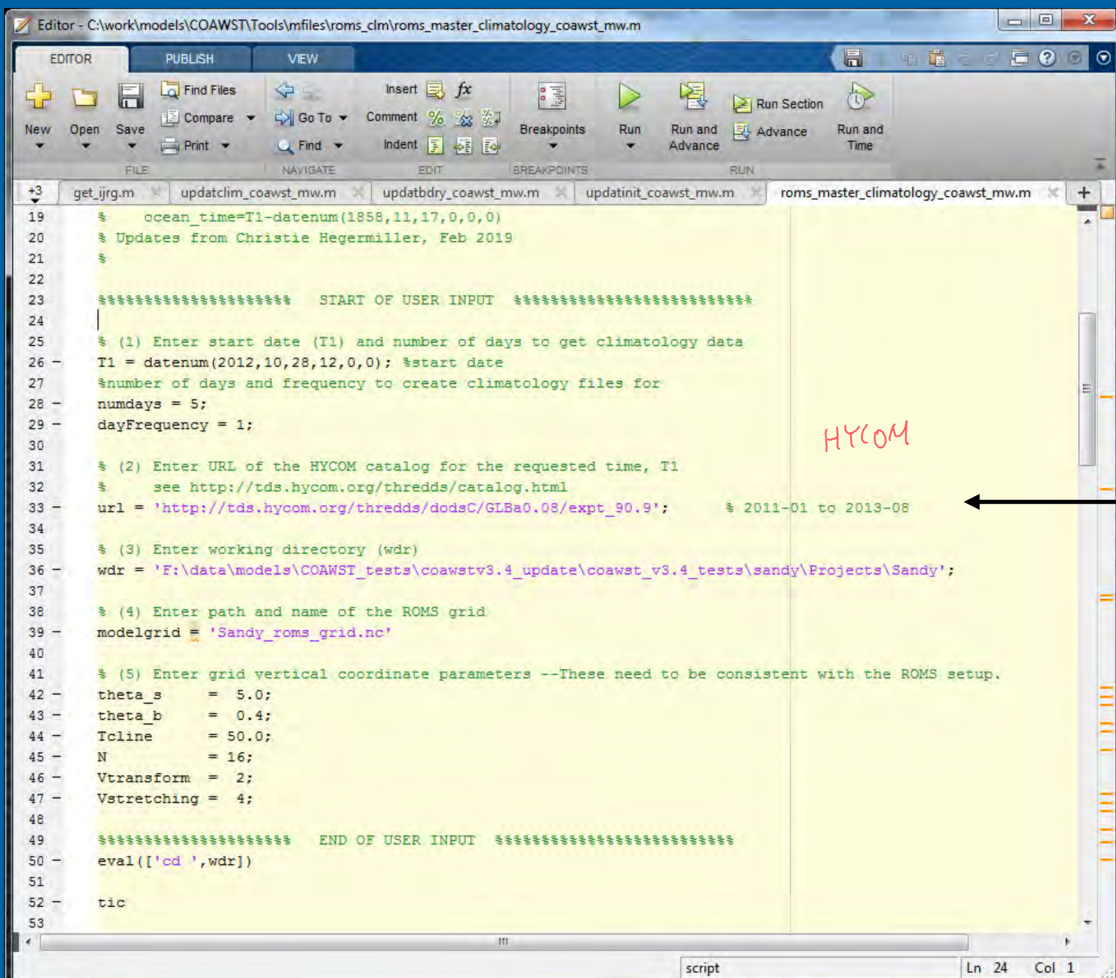
```
netcdf_load('wrfinput_d01');
F = TriScatteredInterp(double(XLONG(:)),double(XLAT(:)), ...
                       double(1-LANDMASK(:)),'nearest');
roms_mask=F(lon_rho,lat_rho);
figure
pcolorjw(lon_rho,lat_rho,roms_mask)
title('ROMS child mask')
xlabel('longitude'); ylabel('latitiude')
hold on
plot(lon,lat,'r')
water = double(roms_mask);
u_mask = water(1:end-1,:) & water(2:end,:);
v_mask= water(:,1:end-1) & water(:,2:end);
psi_mask= water(1:end-1,1:end-1) & water(1:end-1,2:end) &
water(2:end,1:end-1) & water(2:end,2:end);
ncwrite('Sandy_roms_grid_ref3.nc','mask_rho',roms_mask);
ncwrite('Sandy_roms_grid_ref3.nc','mask_u',double(u_mask));
ncwrite('Sandy_roms_grid_ref3.nc','mask_v',double(v_mask));
ncwrite('Sandy_roms_grid_ref3.nc','mask_psi',double(psi_mask));
```



You can use editmask again to make this nicer!

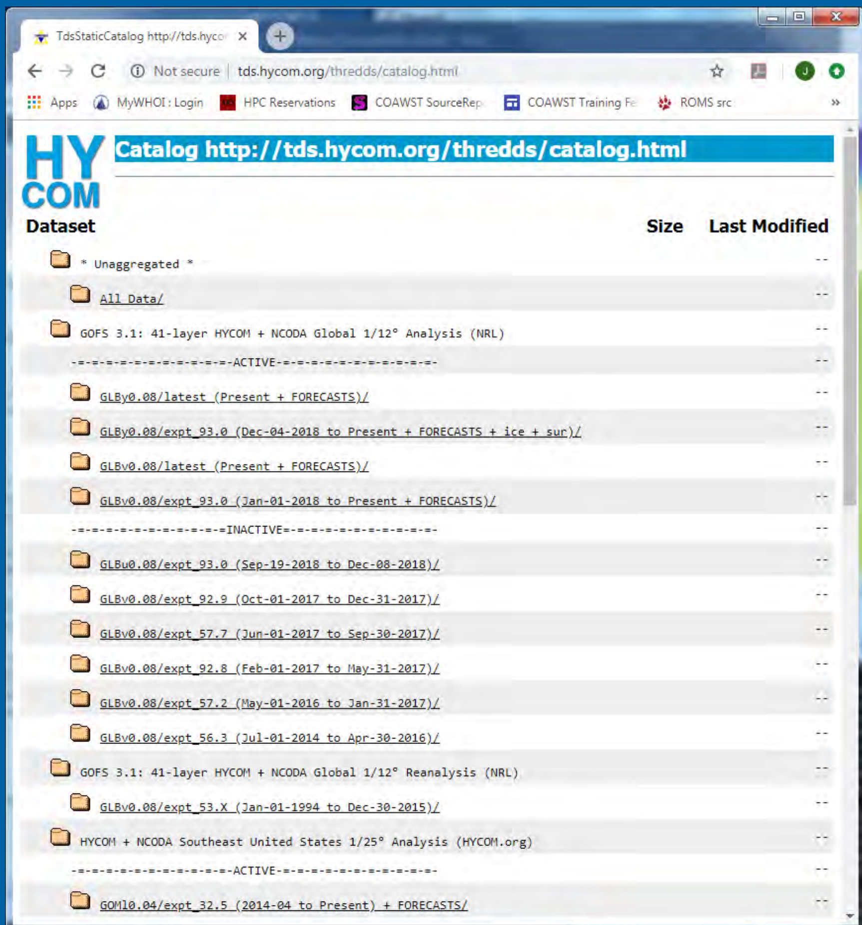
5) 3D BC's (u,v,temp,salt), init, clim

This step creates roms init conditions, bc's, and nudging files.
There are several tools out there. Here we will use
Tools/mfiles/mtools/roms_master_climatology_coawst_mw
This m file will create BC, IC, and Climatology for the parent grid.

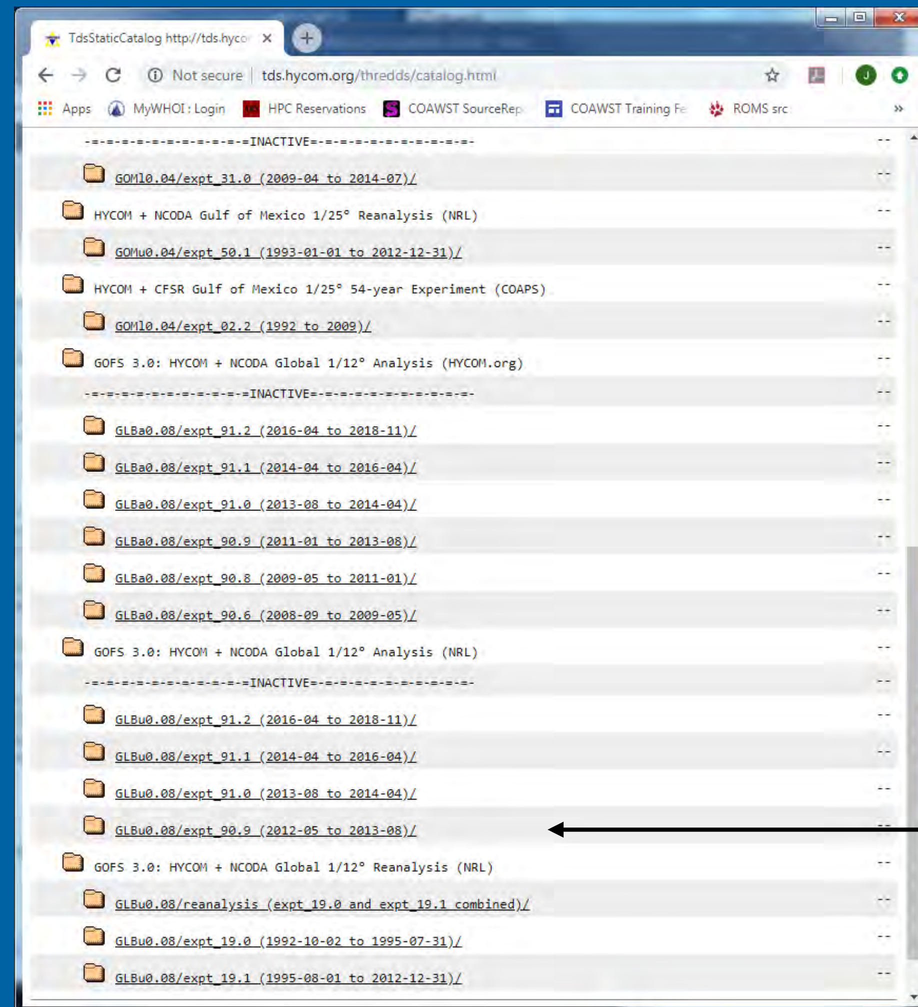


Inactive

5) 3D BC's (u,v,temp,salt), init, clim

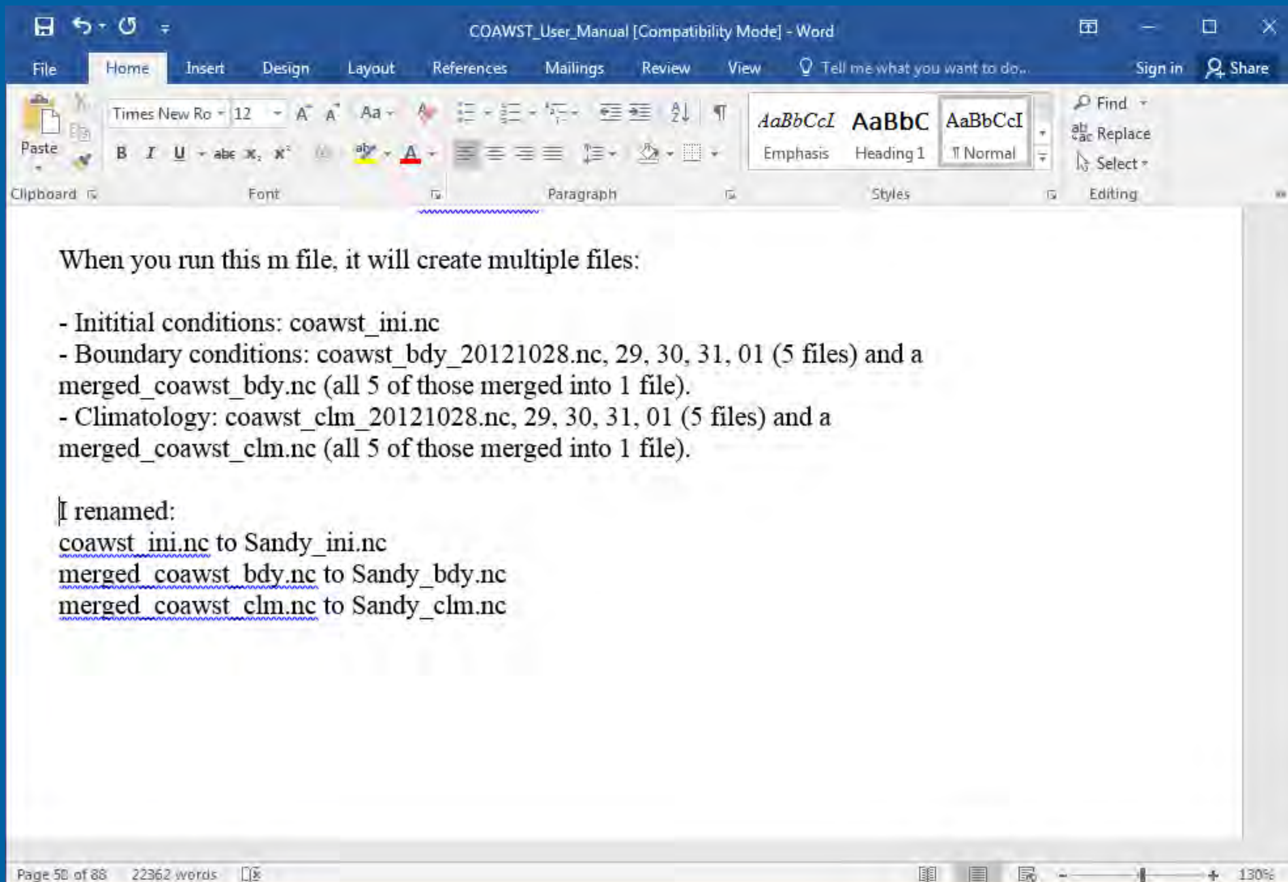


Active



Inactive

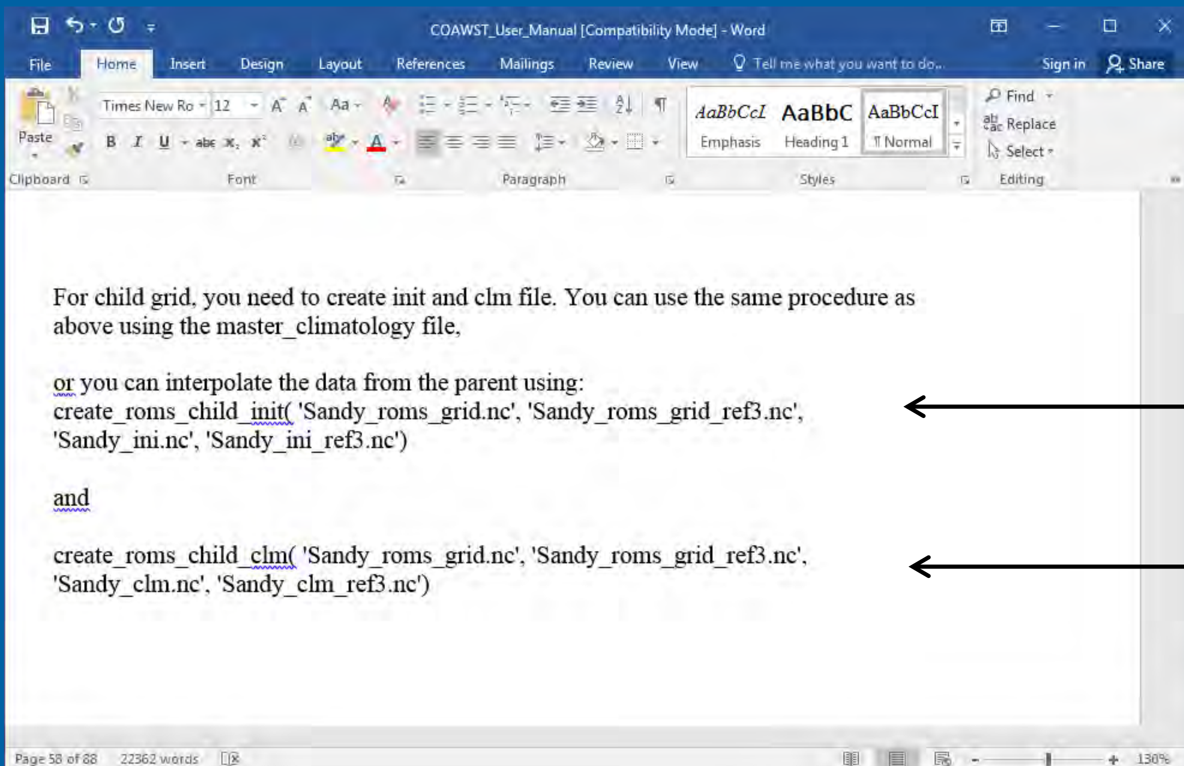
5) 3D BC's (u,v,temp,salt), init, clim


When you run this m file, it will create multiple files:

- Inititial conditions: coawst_ini.nc
- Boundary conditions: coawst_bdy_20121028.nc, 29, 30, 31, 01 (5 files) and a merged_coawst_bdy.nc (all 5 of those merged into 1 file).
- Climatology: coawst_clm_20121028.nc, 29, 30, 31, 01 (5 files) and a merged_coawst_clm.nc (all 5 of those merged into 1 file).

I renamed:
coawst_ini.nc to Sandy_ini.nc
merged_coawst_bdy.nc to Sandy_bdy.nc
merged_coawst_clm.nc to Sandy_clm.nc

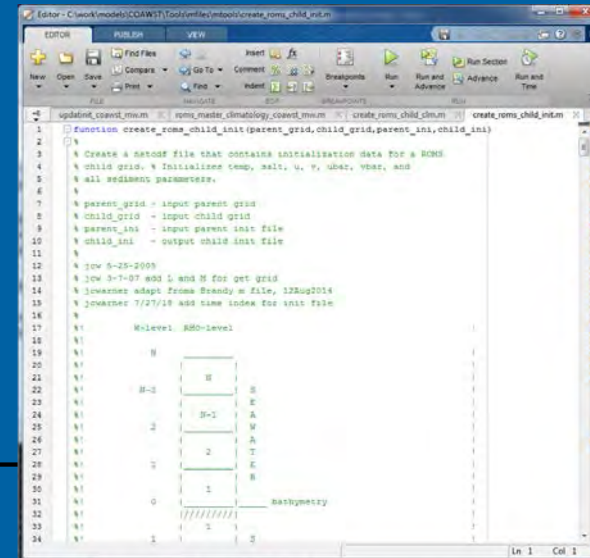
5) For child grid need init and clim.

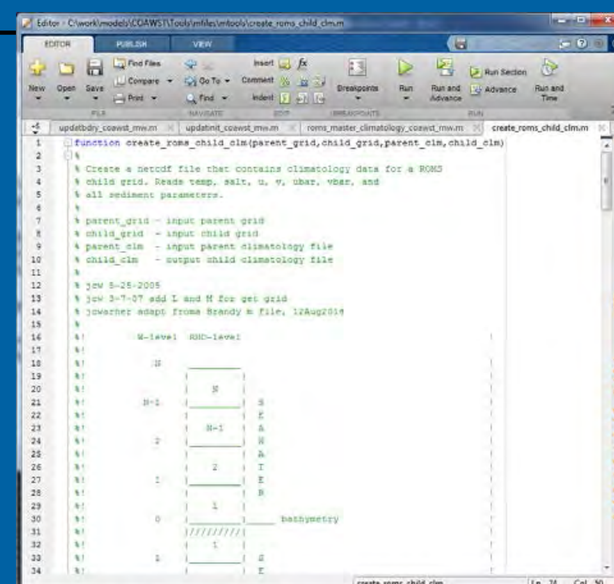

For child grid, you need to create init and clm file. You can use the same procedure as above using the master_climatology file,

or you can interpolate the data from the parent using:
create_roms_child_init('Sandy_roms_grid.nc', 'Sandy_roms_grid_ref3.nc', 'Sandy_ini.nc', 'Sandy_ini_ref3.nc')

and

create_roms_child_clm('Sandy_roms_grid.nc', 'Sandy_roms_grid_ref3.nc', 'Sandy_clm.nc', 'Sandy_clm_ref3.nc')





6) 2D: BC's (ubar, vbar, zeta) = tides

page | discussion | view source | history

Description of Tidal Forcing in ROMS

Contents [hide]

Tidal Forcing Files in ROMS

Once the appropriate CPP options have been set (e.g. SSH_TIDES, UV_TIDES, RAMP_TIDES), a netcdf file of tidal constituents must be generated.

1. **Download the desired tidal constituent database and associated software**. There are many tidal constituent databases available for download and the choice of databases depends on the desired constituents and location of the ROMS grid. Two possibilities which have broad geographic range and generally the dominant constituents are the OSU Tidal Data Inversion Software (OTIS) and the ADCIRC tidal database. Remember to download the associated software with each data base as there ire typically routines which facilitate the extraction and interpolation of the database to the ROMS grid.
2. **Interpolate the tidal constituent database to the desired ROMS grid**. See the above comment. While it is possible to write code (e.g. MATLAB, FORTRAN) to perform this task, it is typically easier to use the provided packages.
3. **Verify all open boundary grid cells contain valid data**. During the interpolation process, depending on the land mask for the ROMS grid and the tidal database grid, it is possible to have grid cells near land points which the ROMS grid may define as water, but the tidal grid defines as land. Should this occur a 180 degree phase shift along the open boundary near land points is possible. As ROMS is tidally forced on the open boundary, this could be problematic.
4. **The phase of the u/v/zeta components of the tidal constituents should be shifted to the appropriate reference time(t=to)**.It is typical for tidal constituent databases to be stored with the phase shifted by the equilibrium tidal argument. Consequently the reference time for the tide is not the desired time, as set in ocean.in using the variable TIDE_START. In addition, nodal corrections need to be made due to long period tides. The equilibrium argument and nodal corrections can be calculated using the tidal database software or Rich Pawlowicz's T_TIDE MATLAB package. Also, see below for more thoughts on the 18.6 year business.
5. **Convert the amplitude/phase information to tidal ellipse parameters, if necessary**. ROMS requires tidal information to be stored as ellipse parameters for use. One tidal ellipse package is ap2ep.m from Zhigang Xu.MATLAB tidal ellipse code is available.
6. **Export tidal ellipse parameters to a ROMS netcdf forcing file**.

Two examples of this process are described below.

⚠ Newer Matlab versions are incompatible with older versions of t_tide and will cause errors. In particular, the "finite" function has been replaced with "isfinite" Rich Pawlowicz has provided an updated version of t_tide on his website http://www.eos.ubc.ca/~rich/#T_Tide. I have also provided a link here [1]

OSU Tidal Prediction Software Example (Matlab)

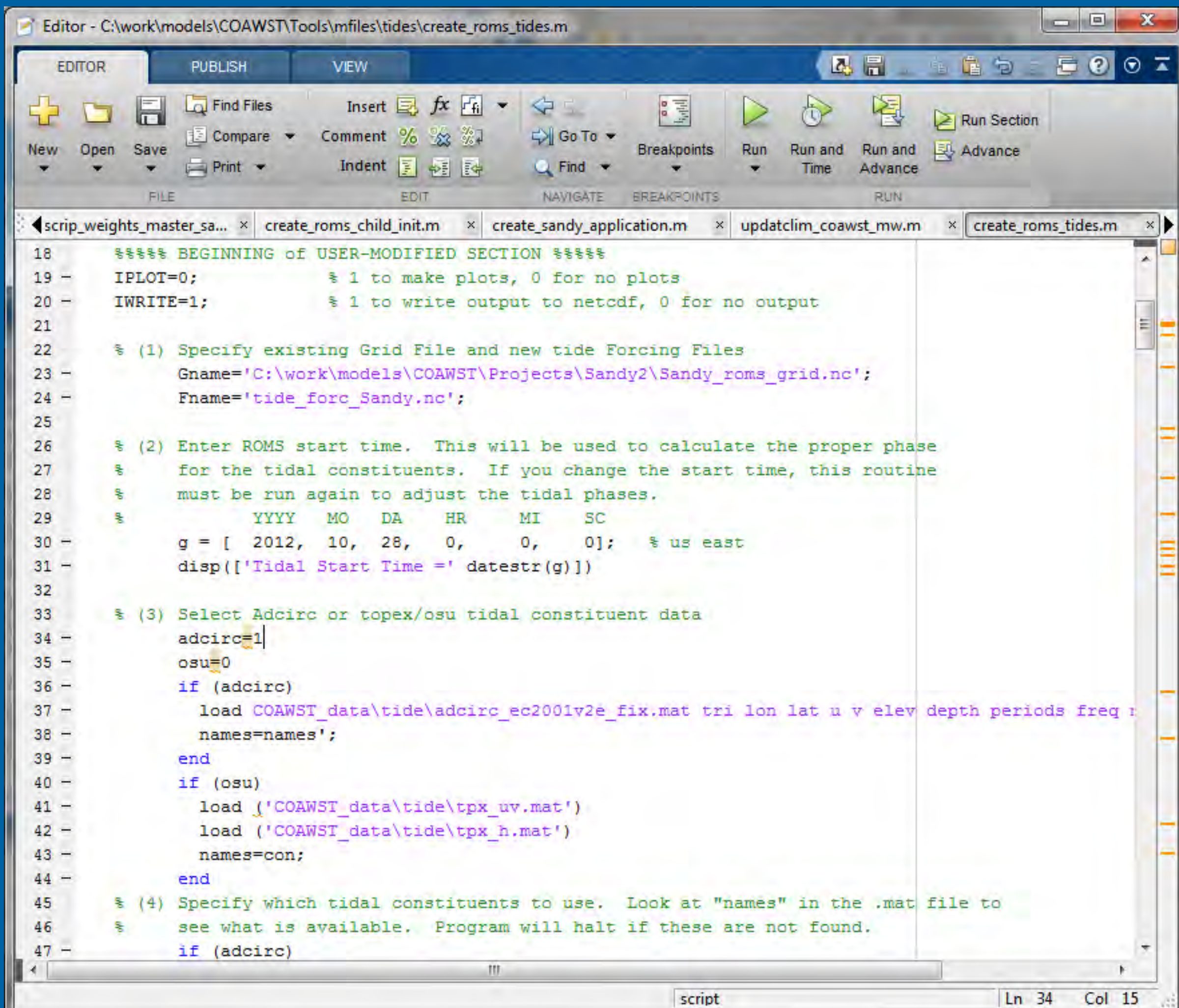
The processing in this example was carried out in MATLAB using routines found in http://marine.rutgers.edu/~hunter/roms/tides/otps/. It has been

6) 2D: BC's (ubar, vbar, zeta) = tides

1) Get the tidal data at

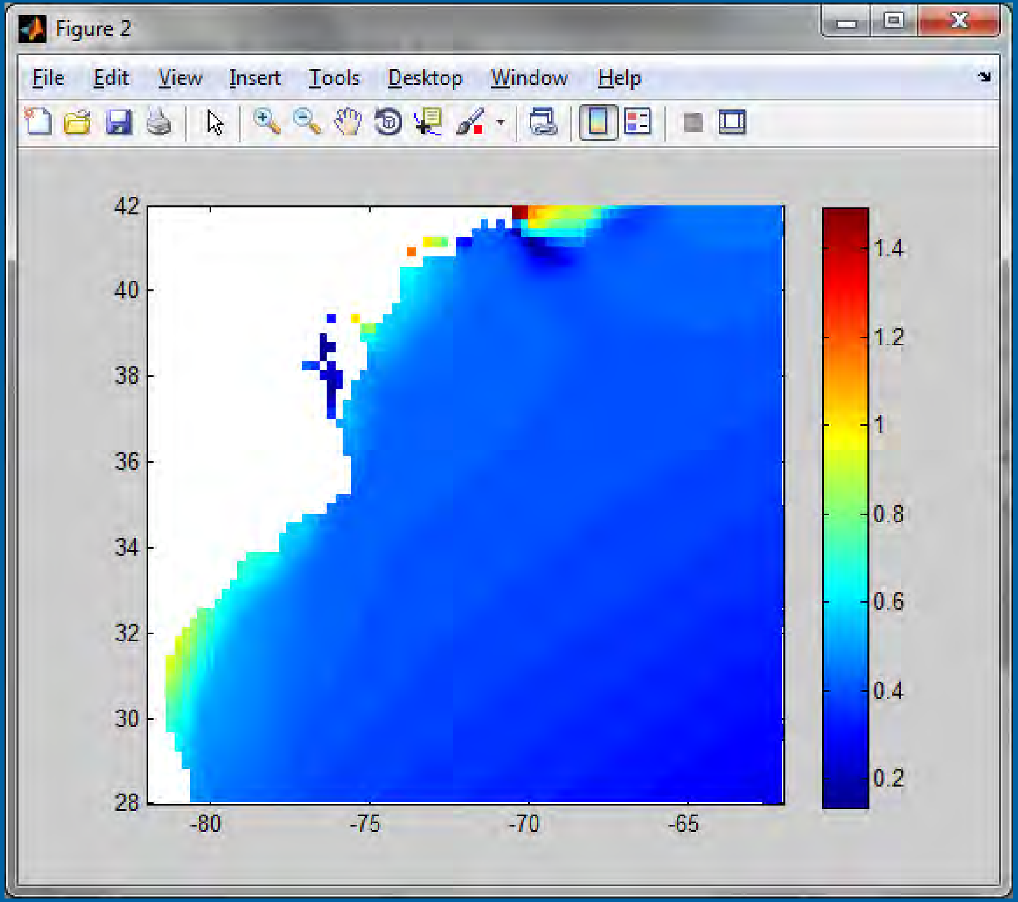
svn checkout https://coawstmodel.sourcerepo.com/coawstmodel/data .

2) edit Tools/mfiles/tides/create_roms_tides



6) 2D: BC's (ubar, vbar, zeta) = tides

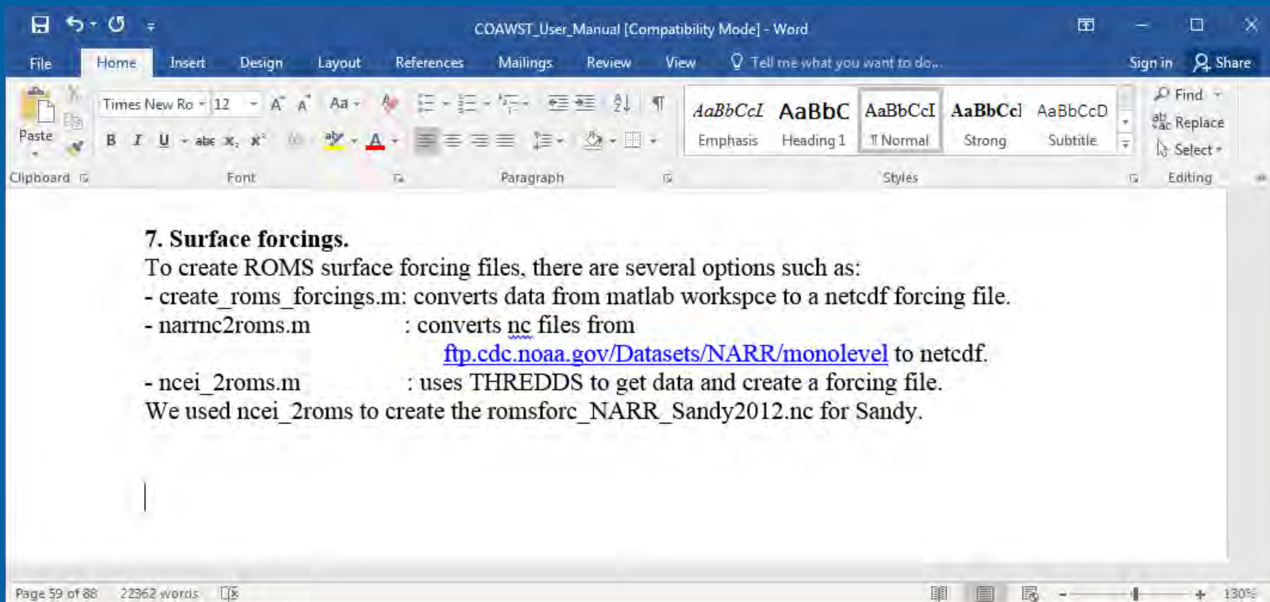
```
netcdf_load('Sandy_roms_grid.nc')
pcolorjw(lon_rho,lat_rho,squeeze(tide_Eamp(:,:,1)))
```



M2 tidal amplitude

7) Surface forcings

1時間9分30秒へ

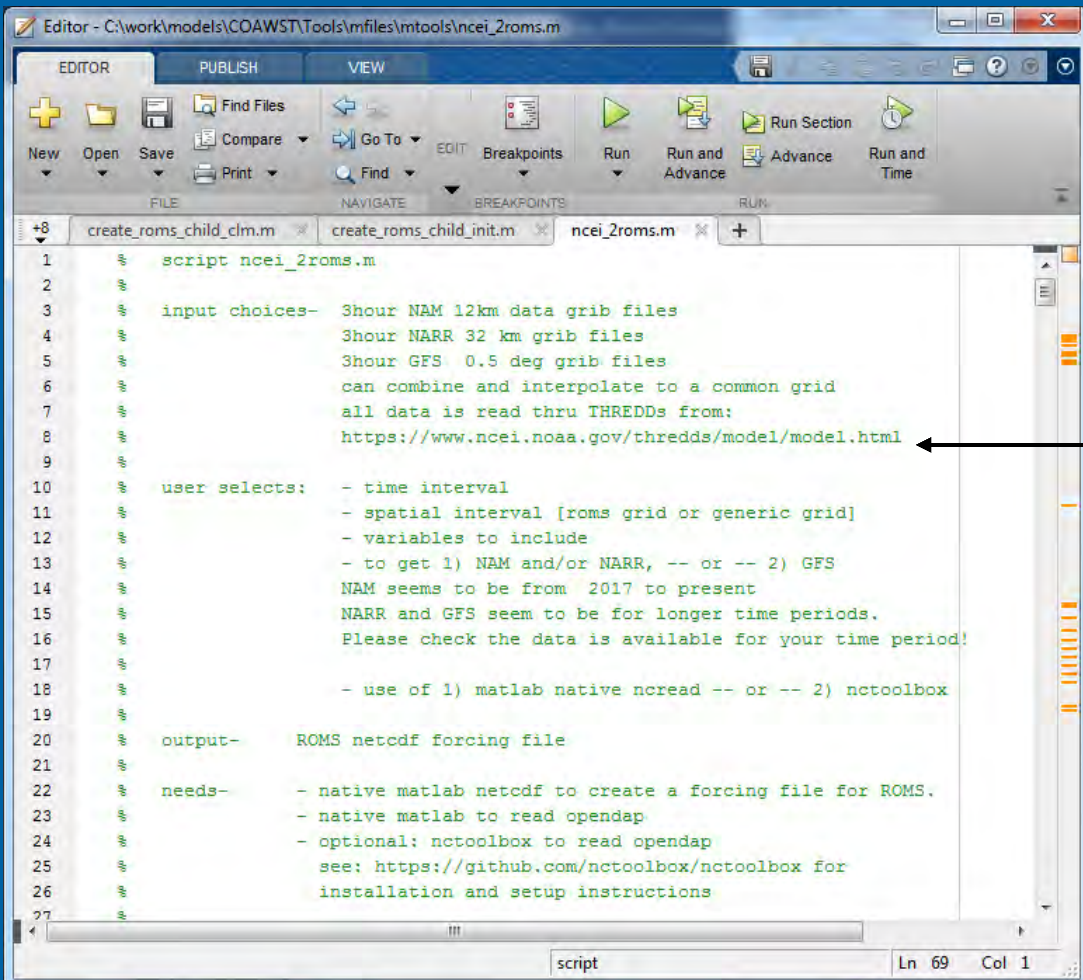


7. Surface forcings.
To create ROMS surface forcing files, there are several options such as:
- create_roms_forcings.m: converts data from matlab workspce to a netcdf forcing file.
- narrnc2roms.m : converts nc files from ftp.cdc.noaa.gov/Datasets/NARR/monolevel to netcdf.
- ncei_2roms.m : uses THREDDS to get data and create a forcing file.

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

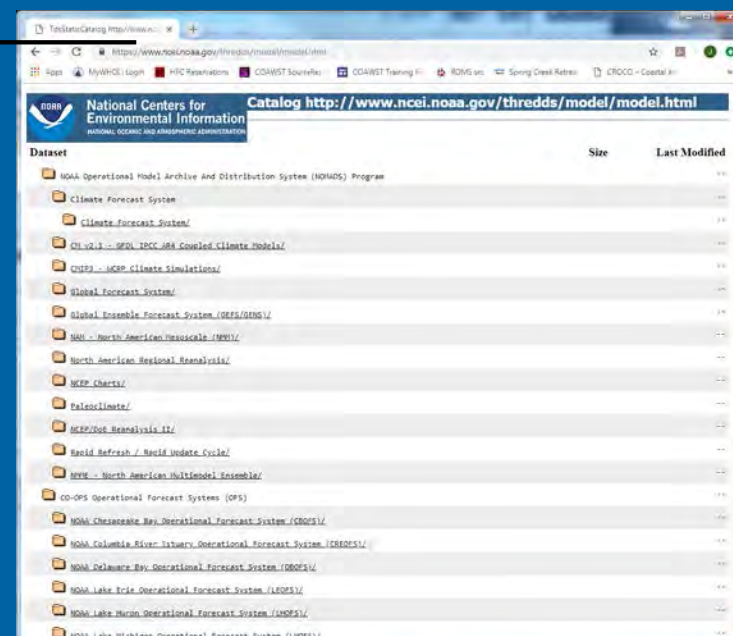
Many options. We will use ncei_2roms.m

7) Surface forcings – ncei_2roms.m

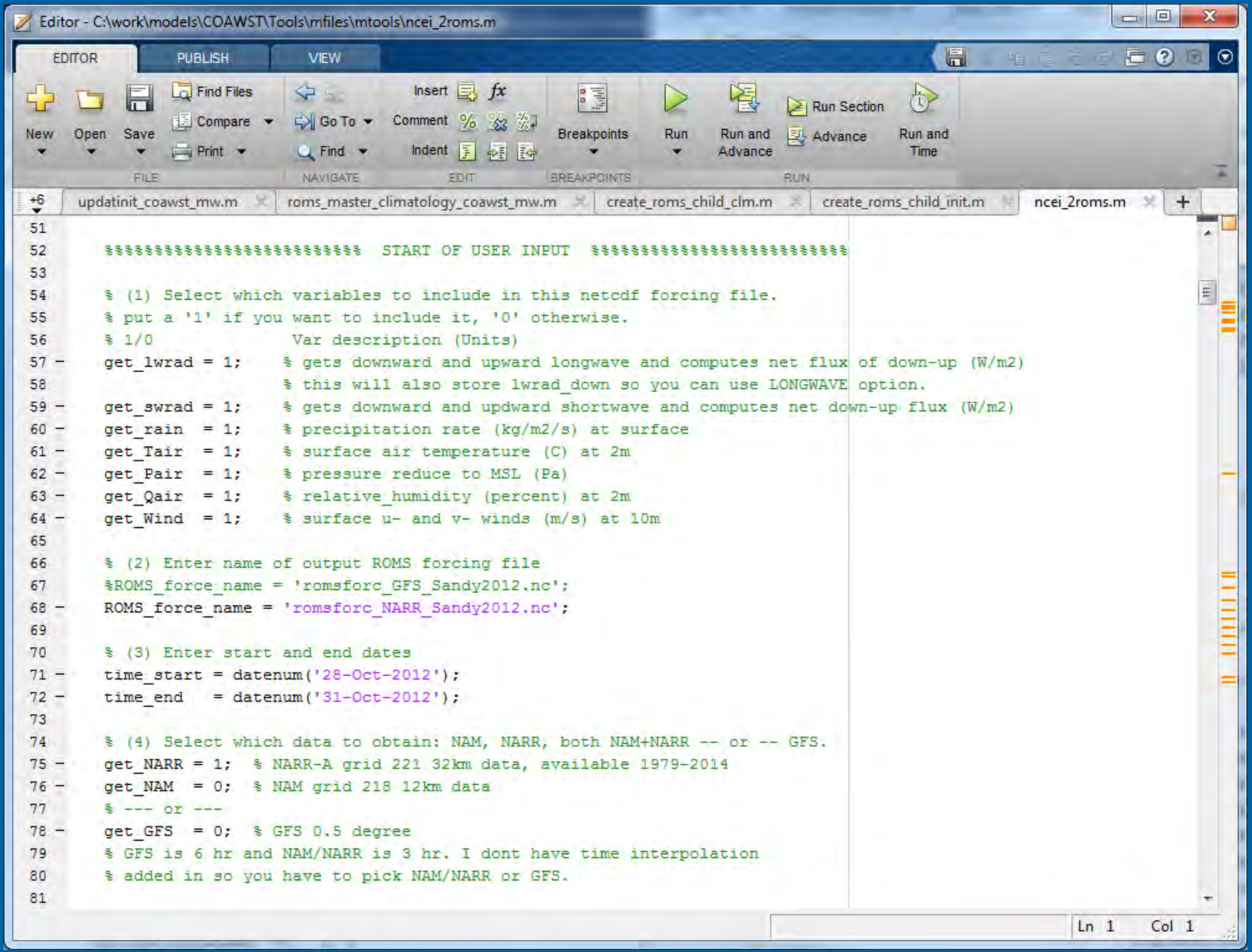


COAWST/Tools/mfiles/mtools/
ncei_2roms

creates netcdf forcing files for ROMS.



7) Surface forcings – ncei_2roms.m



1) Select variables to be downloaded.

2) Forcing file name

3) Times

4) Data sets

7) Surface forcings – ncei_2roms.m

Editor - F:\data\models\COAWST\Tools\mfiles\mtools\ncei_2roms.m

```matlab

% (4) Select which data to obtain: NAM, NARR, both NAM+NARR -- or -- GFS.
get_NARR = 1;  % NARR-A grid 221 32km data, available 1979-2014
get_NAM  = 0;  % NAM grid 218 12km data
% --- or ---
get_GFS  = 0;  % GFS 0.5 degree
% GFS is 6 hr and NAM/NARR is 3 hr. I dont have time interpolation
% added in so you have to pick NAM/NARR or GFS.

% (5) Select to interpolate to a roms grid or a user defined grid.
% Set one of these to a 1, the other to a 0.
interpto_roms_grid = 0;
interpto_user_grid = 1;

% The code has been written so that longitude is in -deg E.
if interpto_roms_grid
  model_grid = 'C:\models\grids\USEast_grd31.nc';
elseif interpto_user_grid
% Provide lon_rho, lat_rho, and angle_rho.
% NAM / NARR grids are centered at~ -100 deg lon; GFS = 0:360 lon
  if (get_NARR); offset=-360; end
  if (get_NAM);  offset=-360; end
  if (get_GFS);  offset=0;    end
% You Probably want to make a finer resolution from 0.2 to 0.1.
  lon_rho = [255:0.2:310]+offset;
  lat_rho = [ 10:0.2:50 ];
  lon_rho = repmat(lon_rho,size(lat_rho,2),1)';
  lat_rho = repmat(lat_rho',1,size(lon_rho,1))';
  angle_rho = lon_rho*0;
else
  disp('pick a grid')
end

% (6) Select which method to use: ncread or nctoolbox
use_matlab = 1;
use_nctoolbox = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  END OF USER INPUT  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

← user_gridを選ぶ → nesting gridに利用

コード中のurlの修正要

コード中の変数定義の修正要

} 設定する

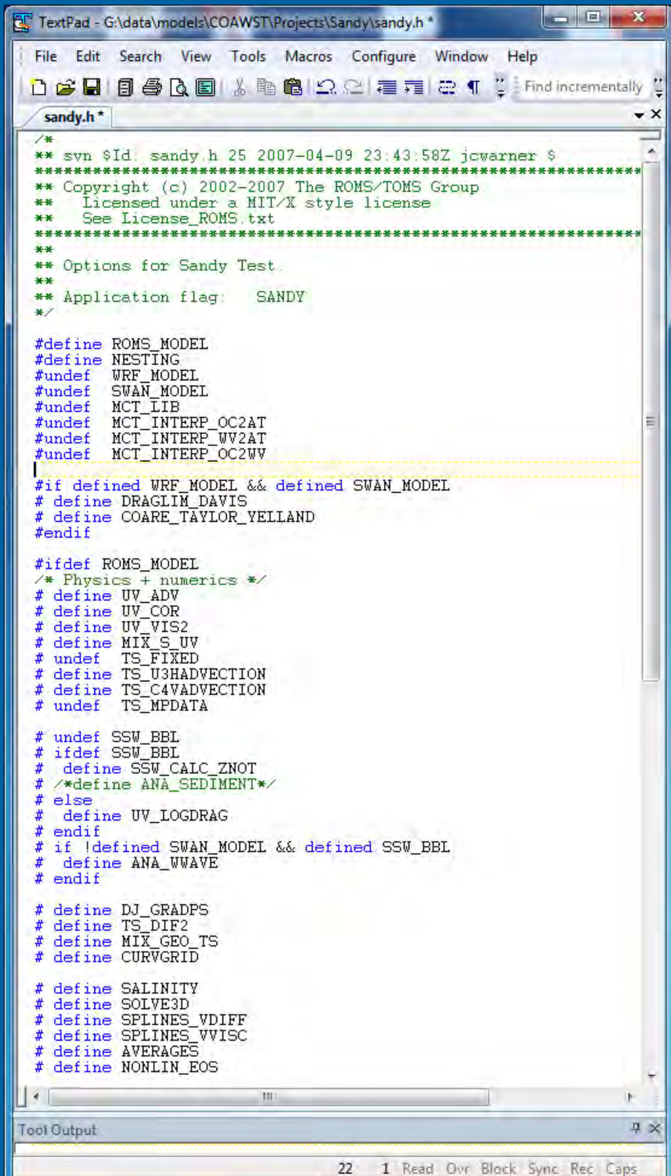
← 設定する

2 usages of "interpto_user_grid" found   script   Ln 90   Col 26

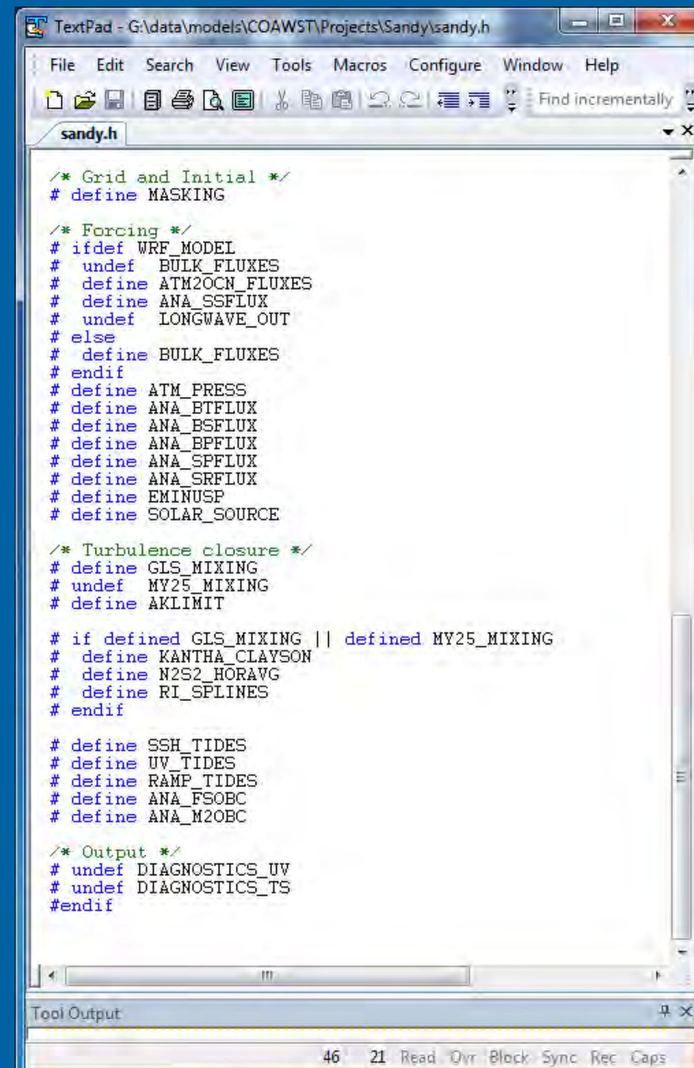
5) User or model grid

6) Native tools or outside toolbox

8) ROMS input files: sandy.h




```
/*
** svn $Id: sandy.h 25 2007-04-09 23:43:58Z jcwarner $
*******************************************************************
** Copyright (c) 2002-2007 The ROMS/TOMS Group
**   Licensed under a MIT/X style license
**   See License_ROMS.txt
*******************************************************************
**
** Options for Sandy Test.
**
** Application flag:   SANDY
*/

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

#if defined WRF_MODEL && defined SWAN_MODEL
# define DRAGLIM_DAVIS
# define COARE_TAYLOR_YELLAND
#endif

#ifdef ROMS_MODEL
/* Physics + numerics */
# define UV_ADV
# define UV_COR
# define UV_VIS2
# define MIX_S_UV
# undef  TS_FIXED
# define TS_U3HADVECTION
# define TS_C4VADVECTION
# undef  TS_MPDATA

# undef SSW_BBL
# ifdef SSW_BBL
#  define SSW_CALC_ZNOT
# /*define ANA_SEDIMENT*/
# else
#  define UV_LOGDRAG
# endif
# if !defined SWAN_MODEL && defined SSW_BBL
#  define ANA_WWAVE
# endif

# define DJ_GRADPS
# define TS_DIF2
# define MIX_GEO_TS
# define CURVGRID

# define SALINITY
# define SOLVE3D
# define SPLINES_VDIFF
# define SPLINES_VVISC
# define AVERAGES
# define NONLIN_EOS
```




```
/* Grid and Initial */
# define MASKING

/* Forcing */
# ifdef WRF_MODEL
#  undef  BULK_FLUXES
#  define ATM2OCN_FLUXES
#  define ANA_SSFLUX
#  undef  LONGWAVE_OUT
# else
#  define BULK_FLUXES
# endif
# define ATM_PRESS
# define ANA_BTFLUX
# define ANA_BSFLUX
# define ANA_BPFLUX
# define ANA_SPFLUX
# define ANA_SRFLUX
# define EMINUSP
# define SOLAR_SOURCE

/* Turbulence closure */
# define GLS_MIXING
# undef  MY25_MIXING
# define AKLIMIT

# if defined GLS_MIXING || defined MY25_MIXING
#  define KANTHA_CLAYSON
#  define N2S2_HORAVG
#  define RI_SPLINES
# endif

# define SSH_TIDES
# define UV_TIDES
# define RAMP_TIDES
# define ANA_FSOBC
# define ANA_M2OBC

/* Output */
# undef DIAGNOSTICS_UV
# undef DIAGNOSTICS_TS
#endif
```

8) ROMS input files: sandy_ocean.in

TextPad - F:\data\models\COAWST\Projects\Sandy\ocean_sandy.in

File Edit Search View Tools Macros Configure Window Help

ocean_sandy.in

```
!
! Application title.

       TITLE = Hurricane Sandy

! C-preprocessing Flag.

    MyAppCPP = SANDY

! Input variable information file name.  This file needs to be processed
! first so all information arrays can be initialized properly.

     VARNAME = ROMS/External/varinfo.dat

! Number of nested grids.

      Ngrids =  2

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

  NestLayers =  2

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

GridsInLayer =  1 1

! Grid dimension parameters. See notes below in the Glossary for how to set
! these parameters correctly.

          Lm == 85  114         ! Number of I-direction INTERIOR RHO-points
          Mm == 63   84         ! Number of J-direction INTERIOR RHO-points
           N == 16   16         ! Number of vertical levels
          ND ==  0              ! Number of wave directional bins

        Nbed =  0               ! Number of sediment bed layers

         NAT =  2               ! Number of active tracers (usually, 2)
         NPT =  0               ! Number of inactive passive tracers
         NCS =  0               ! Number of cohesive (mud) sediment tracers
         NNS =  0               ! Number of non-cohesive (sand) sediment tracers

! 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 == 1                               ! I-direction partition
      NtileJ == 1                               ! J-direction partition

! Set lateral boundary conditions keyword. Notice that a value is expected
! for each boundary segment per nested grid for each state variable.
!
```

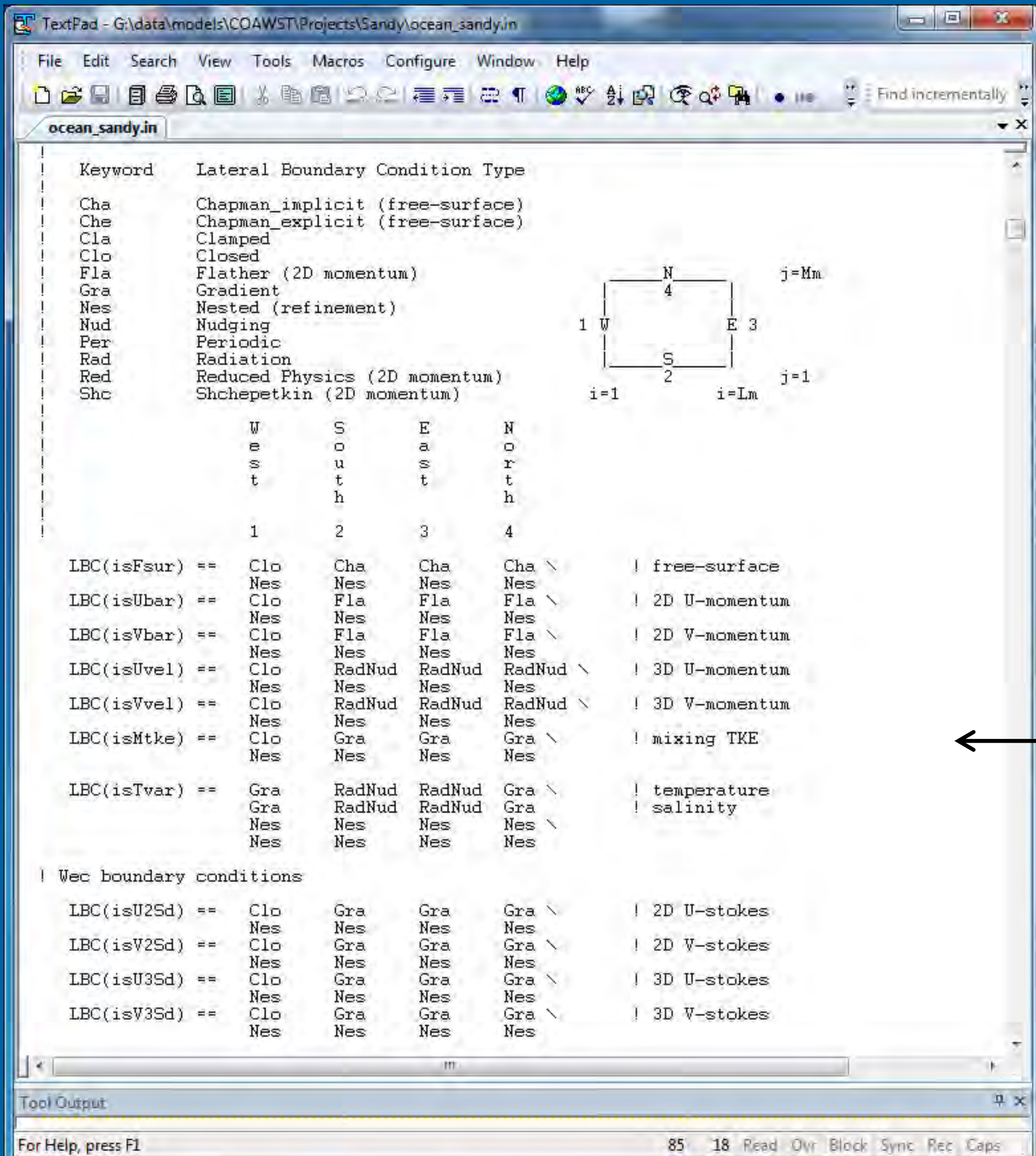
84 1 Read Ovr Block Sync Rec Caps

Nesting params

Grid sizes

NtileI NtileJ

8) ROMS input files: sandy_ocean.in



TextPad - G:\data\models\COAWST\Projects\Sandy\ocean_sandy.in

File Edit Search View Tools Macros Configure Window Help

ocean_sandy.in

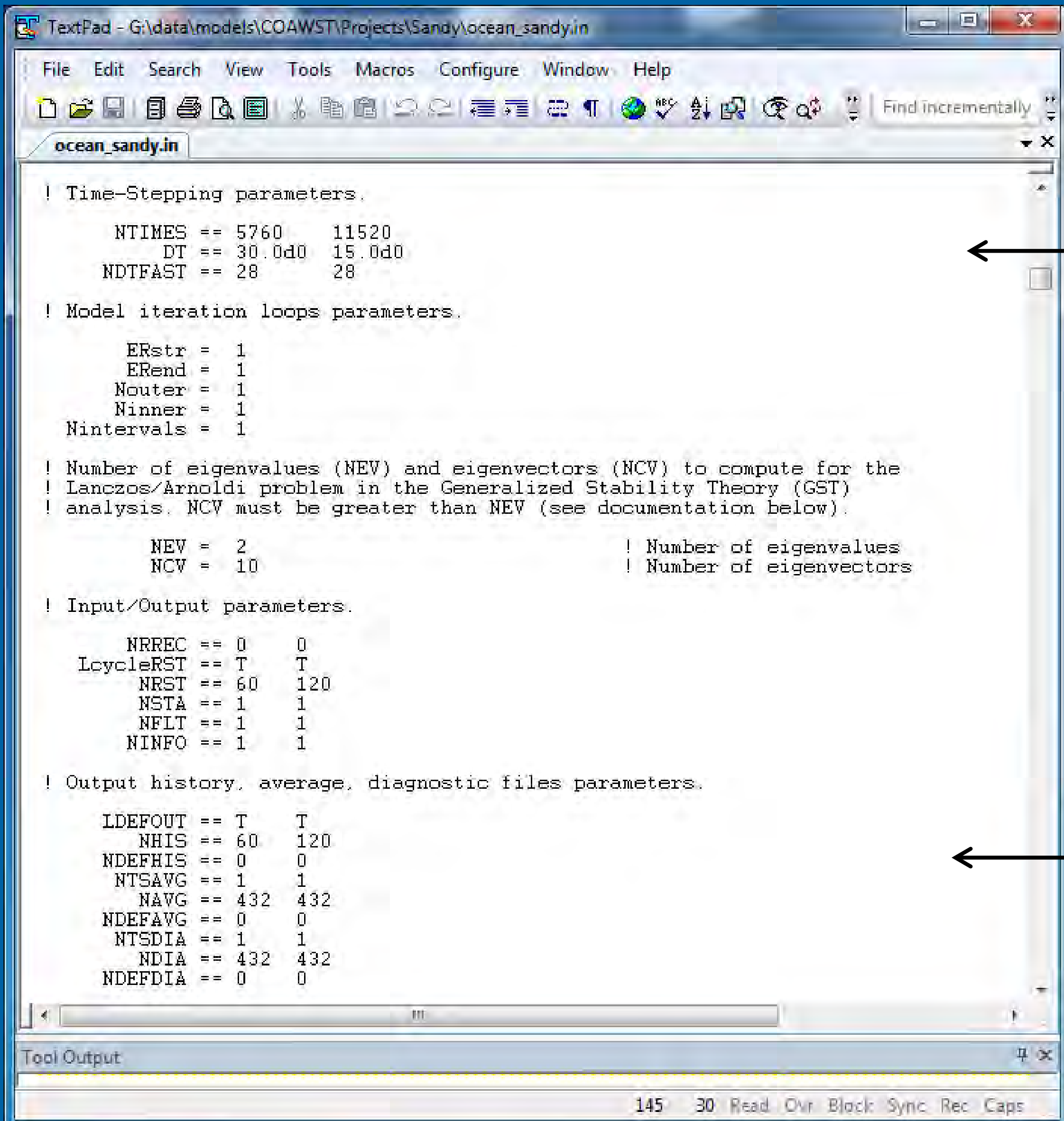
```
!
!   Keyword    Lateral Boundary Condition Type
!
!   Cha        Chapman_implicit (free-surface)
!   Che        Chapman_explicit (free-surface)
!   Cla        Clamped
!   Clo        Closed
!   Fla        Flather (2D momentum)                      _____N_____       j=Mm
!   Gra        Gradient                                  |     4     |
!   Nes        Nested (refinement)                       |           |
!   Nud        Nudging                                 1 W           E 3
!   Per        Periodic                                  |           |
!   Rad        Radiation                                 |_____S_____|
!   Red        Reduced Physics (2D momentum)                   2            j=1
!   Shc        Shchepetkin (2D momentum)                i=1         i=Lm
!
!                     W       S       E       N
!                     e       o       a       o
!                     s       u       s       r
!                     t       t       t       t
!                             h               h
!
!                     1       2       3       4

   LBC(isFsur) ==   Clo     Cha     Cha     Cha \        ! free-surface
                    Nes     Nes     Nes     Nes
   LBC(isUbar) ==   Clo     Fla     Fla     Fla \        ! 2D U-momentum
                    Nes     Nes     Nes     Nes
   LBC(isVbar) ==   Clo     Fla     Fla     Fla \        ! 2D V-momentum
                    Nes     Nes     Nes     Nes
   LBC(isUvel) ==   Clo     RadNud  RadNud  RadNud \     ! 3D U-momentum
                    Nes     Nes     Nes     Nes
   LBC(isVvel) ==   Clo     RadNud  RadNud  RadNud \     ! 3D V-momentum
                    Nes     Nes     Nes     Nes
   LBC(isMtke) ==   Clo     Gra     Gra     Gra \        ! mixing TKE
                    Nes     Nes     Nes     Nes

   LBC(isTvar) ==   Gra     RadNud  RadNud  Gra \        ! temperature
                    Gra     RadNud  RadNud  Gra          ! salinity
                    Nes     Nes     Nes     Nes \
                    Nes     Nes     Nes     Nes

! Wec boundary conditions

   LBC(isU2Sd) ==   Clo     Gra     Gra     Gra \        ! 2D U-stokes
                    Nes     Nes     Nes     Nes
   LBC(isV2Sd) ==   Clo     Gra     Gra     Gra \        ! 2D V-stokes
                    Nes     Nes     Nes     Nes
   LBC(isU3Sd) ==   Clo     Gra     Gra     Gra \        ! 3D U-stokes
                    Nes     Nes     Nes     Nes
   LBC(isV3Sd) ==   Clo     Gra     Gra     Gra \        ! 3D V-stokes
                    Nes     Nes     Nes     Nes
```

Tool Output

For Help, press F1 85 18 Read Ovr Block Sync Rec Caps

BC's

8) ROMS input files: sandy_ocean.in




```
! Time-Stepping parameters.

     NTIMES == 5760     11520
         DT == 30.0d0   15.0d0
    NDTFAST == 28       28

! Model iteration loops parameters.

      ERstr =  1
      ERend =  1
     Nouter =  1
     Ninner =  1
 Nintervals =  1

! Number of eigenvalues (NEV) and eigenvectors (NCV) to compute for the
! Lanczos/Arnoldi problem in the Generalized Stability Theory (GST)
! analysis. NCV must be greater than NEV (see documentation below).

        NEV =  2                               ! Number of eigenvalues
        NCV =  10                              ! Number of eigenvectors

! Input/Output parameters.

      NRREC == 0     0
  LcycleRST == T     T
       NRST == 60    120
       NSTA == 1     1
       NFLT == 1     1
      NINFO == 1     1

! Output history, average, diagnostic files parameters.

    LDEFOUT == T     T
       NHIS == 60    120
    NDEFHIS == 0     0
     NTSAVG == 1     1
       NAVG == 432   432
    NDEFAVG == 0     0
     NTSDIA == 1     1
       NDIA == 432   432
    NDEFDIA == 0     0
```

Time stepping

nhis, navg, etc

8) ROMS input files: sandy_ocean.in

TextPad - F:\data\models\COAWST\Projects\Sandy\ocean_sandy.in

File Edit Search View Tools Macros Configure Window Help

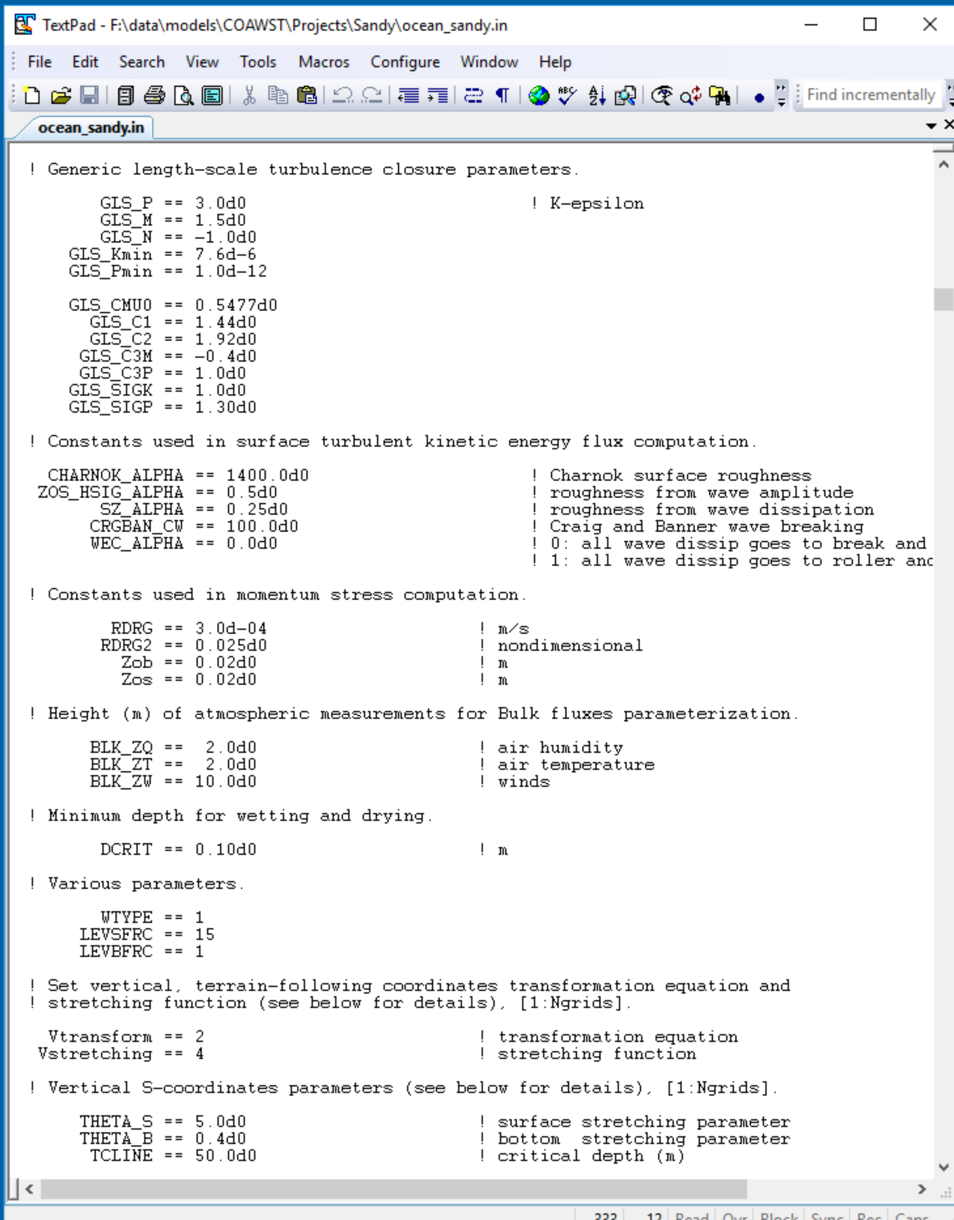
ocean_sandy.in

```
! Harmonic/biharmonic horizontal diffusion of tracer for nonlinear model
! and adjoint-based algorithms: [1:NAT+NPT,Ngrids].

        TNU2 == 0.2d0  0.2d0 0.2d0  0.2d0       ! m2/s
        TNU4 == 0.0d0  0.0d0                    ! m4/s

     ad_TNU2 == 0.0d0  0.0d0                    ! m2/s
     ad_TNU4 == 0.0d0  0.0d0                    ! m4/s

! Harmonic/biharmonic, horizontal viscosity coefficient for nonlinear model
! and adjoint-based algorithms: [Ngrids].

       VISC2 == 0.10d0 0.10d0                   ! m2/s
       VISC4 == 0.0d0                           ! m4/s

    ad_VISC2 == 0.0d0                           ! m2/s
    ad_VISC4 == 0.0d0                           ! m4/s

! Logical switches (TRUE/FALSE) to increase/decrease horizontal viscosity
! and/or diffusivity in specific areas of the application domain (like
! sponge areas) for the desired application grid.

    LuvSponge == F                              ! horizontal momentum
LtracerSponge == F F                            ! temperature, salinity, inert

! Vertical mixing coefficients for tracers in nonlinear model and
! basic state scale factor in adjoint-based algorithms: [1:NAT+NPT,Ngrids]

     AKT_BAK == 1.0d-6  1.0d-6 1.0d-6  1.0d-6   ! m2/s

  ad_AKT_fac == 1.0d0  1.0d0                    ! nondimensional

! Vertical mixing coefficient for momentum for nonlinear model and
! basic state scale factor in adjoint-based algorithms: [Ngrids].

     AKV_BAK == 1.0d-5  1.0d-5                  ! m2/s

  ad_AKV_fac == 1.0d0                           ! nondimensional

! Upper threshold values to limit vertical mixing coefficients computed
! from vertical mixing parameterizations. Although this is an engineering
! fix, the vertical mixing values inferred from ocean observations are
! rarely higher than this upper limit value.

   AKT_LIMIT == 1.0d-3 1.0d-3                   ! m2/s

   AKV_LIMIT == 1.0d-3 1.0d-3                   ! m2/s

! Turbulent closure parameters.

     AKK_BAK == 5.0d-6                          ! m2/s
     AKP_BAK == 5.0d-6                          ! m2/s
      TKENU2 == 0.0d0                           ! m2/s
      TKENU4 == 0.0d0                           ! m4/s
```

337 1 Read Ovr Block Sync Rec Caps

Limiters for mixing terms

8) ROMS input files: sandy_ocean.in




```
! Generic length-scale turbulence closure parameters.

       GLS_P == 3.0d0                          ! K-epsilon
       GLS_M == 1.5d0
       GLS_N == -1.0d0
    GLS_Kmin == 7.6d-6
    GLS_Pmin == 1.0d-12

    GLS_CMU0 == 0.5477d0
      GLS_C1 == 1.44d0
      GLS_C2 == 1.92d0
     GLS_C3M == -0.4d0
     GLS_C3P == 1.0d0
    GLS_SIGK == 1.0d0
    GLS_SIGP == 1.30d0

! Constants used in surface turbulent kinetic energy flux computation.

  CHARNOK_ALPHA == 1400.0d0                    ! Charnok surface roughness
 ZOS_HSIG_ALPHA == 0.5d0                       ! roughness from wave amplitude
       SZ_ALPHA == 0.25d0                      ! roughness from wave dissipation
      CRGBAN_CW == 100.0d0                     ! Craig and Banner wave breaking
      WEC_ALPHA == 0.0d0                       ! 0: all wave dissip goes to break and
                                               ! 1: all wave dissip goes to roller anc

! Constants used in momentum stress computation.

        RDRG == 3.0d-04                   ! m/s
       RDRG2 == 0.025d0                   ! nondimensional
         Zob == 0.02d0                    ! m
         Zos == 0.02d0                    ! m

! Height (m) of atmospheric measurements for Bulk fluxes parameterization.

      BLK_ZQ ==  2.0d0                    ! air humidity
      BLK_ZT ==  2.0d0                    ! air temperature
      BLK_ZW == 10.0d0                    ! winds

! Minimum depth for wetting and drying.

       DCRIT == 0.10d0                    ! m

! Various parameters.

       WTYPE == 1
     LEVSFRC == 15
     LEVBFRC == 1

! Set vertical, terrain-following coordinates transformation equation and
! stretching function (see below for details), [1:Ngrids].

  Vtransform == 2                         ! transformation equation
 Vstretching == 4                         ! stretching function

! Vertical S-coordinates parameters (see below for details), [1:Ngrids].

     THETA_S == 5.0d0                     ! surface stretching parameter
     THETA_B == 0.4d0                     ! bottom  stretching parameter
      TCLINE == 50.0d0                    ! critical depth (m)
```

GLS params

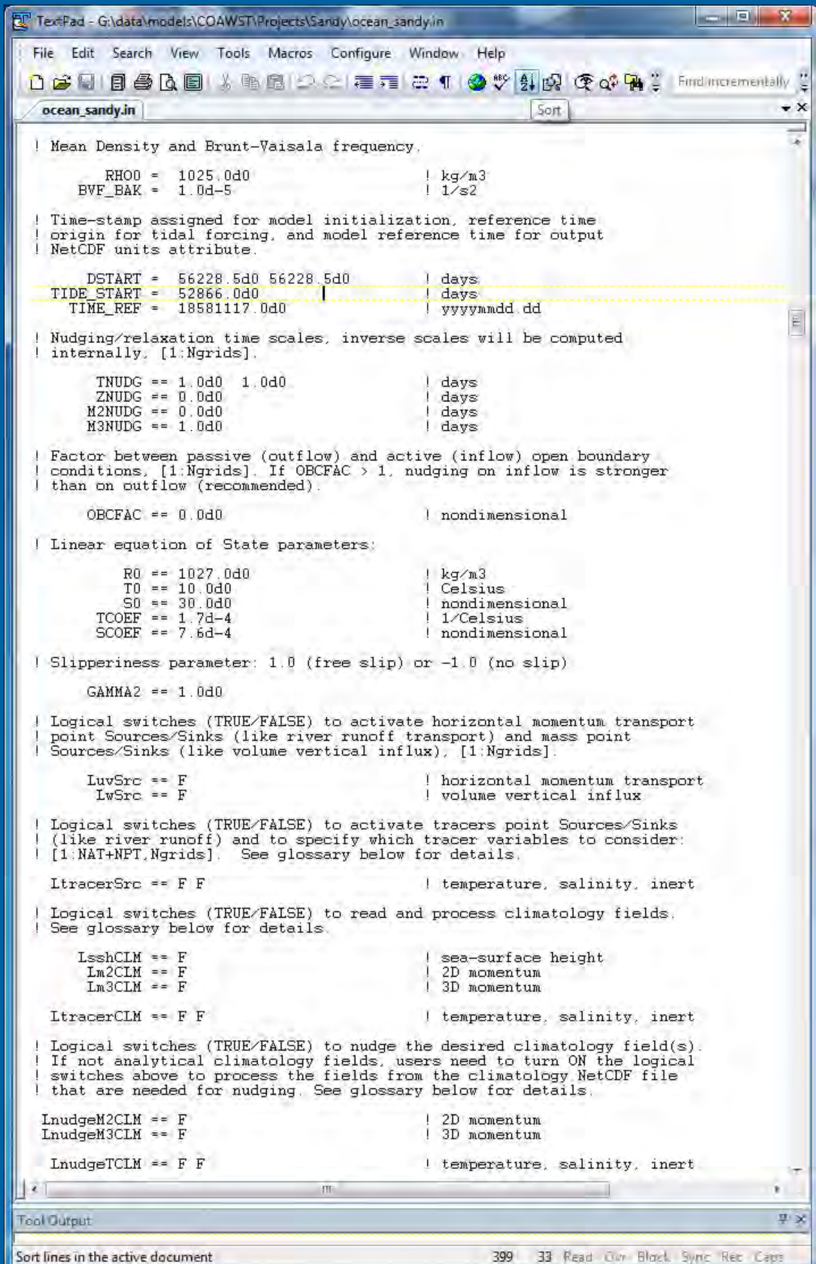
Wave breaking params

Bottom roughness

Bulk flux heights

Grid stretching params

8) ROMS input files: sandy_ocean.in




```
! Mean Density and Brunt-Vaisala frequency.

      RHO0 =  1025.0d0                   ! kg/m3
   BVF_BAK =  1.0d-5                     ! 1/s2

! Time-stamp assigned for model initialization, reference time
! origin for tidal forcing, and model reference time for output
! NetCDF units attribute.

    DSTART =  56228.5d0 56228.5d0        ! days
TIDE_START =  52866.0d0                  ! days
  TIME_REF =  18581117.0d0               ! yyyymmdd.dd

! Nudging/relaxation time scales, inverse scales will be computed
! internally, [1:Ngrids].

     TNUDG == 1.0d0  1.0d0               ! days
     ZNUDG == 0.0d0                      ! days
    M2NUDG == 0.0d0                      ! days
    M3NUDG == 1.0d0                      ! days

! Factor between passive (outflow) and active (inflow) open boundary
! conditions, [1:Ngrids]. If OBCFAC > 1, nudging on inflow is stronger
! than on outflow (recommended).

    OBCFAC == 0.0d0                      ! nondimensional

! Linear equation of State parameters:

        R0 == 1027.0d0                   ! kg/m3
        T0 == 10.0d0                     ! Celsius
        S0 == 30.0d0                     ! nondimensional
     TCOEF == 1.7d-4                     ! 1/Celsius
     SCOEF == 7.6d-4                     ! nondimensional

! Slipperiness parameter: 1.0 (free slip) or -1.0 (no slip)

    GAMMA2 == 1.0d0

! Logical switches (TRUE/FALSE) to activate horizontal momentum transport
! point Sources/Sinks (like river runoff transport) and mass point
! Sources/Sinks (like volume vertical influx), [1:Ngrids].

    LuvSrc == F                          ! horizontal momentum transport
     LwSrc == F                          ! volume vertical influx

! Logical switches (TRUE/FALSE) to activate tracers point Sources/Sinks
! (like river runoff) and to specify which tracer variables to consider:
! [1:NAT+NPT,Ngrids]. See glossary below for details.

LtracerSrc == F F                        ! temperature, salinity, inert

! Logical switches (TRUE/FALSE) to read and process climatology fields.
! See glossary below for details.

   LsshCLM == F                          ! sea-surface height
    Lm2CLM == F                          ! 2D momentum
    Lm3CLM == F                          ! 3D momentum

LtracerCLM == F F                        ! temperature, salinity, inert

! Logical switches (TRUE/FALSE) to nudge the desired climatology field(s).
! If not analytical climatology fields, users need to turn ON the logical
! switches above to process the fields from the climatology NetCDF file
! that are needed for nudging. See glossary below for details.

 LnudgeM2CLM == F                        ! 2D momentum
 LnudgeM3CLM == F                        ! 3D momentum

LnudgeTCLM == F F                        ! temperature, salinity, inert
```

Time starts, tide start

BC tnudge

8) ROMS input files: sandy_ocean.in

TextPad - G:\data\models\COAWST\Projects\Sandy\ocean_sandy.in *

File Edit Search View Tools Macros Configure Window Help

ocean_sandy.in *

```
! Logical switches (TRUE/FALSE) to activate writing of fields into
! HISTORY output file.

Hout(idUvel) == T T     ! u                  3D U-velocity
Hout(idVvel) == T T     ! v                  3D V-velocity
Hout(idu3dE) == F F     ! u_eastward         3D U-eastward  at RHO-points
Hout(idv3dN) == F F     ! v_northward        3D V-northward at RHO-points
Hout(idWvel) == T T     ! w                  3D W-velocity
Hout(idOvel) == T T     ! omega              omega vertical velocity
Hout(idUbar) == T T     ! ubar               2D U-velocity
Hout(idVbar) == T T     ! vbar               2D V-velocity
Hout(idu2dE) == F F     ! ubar_eastward      2D U-eastward  at RHO-points
Hout(idv2dN) == F F     ! vbar_northward     2D V-northward at RHO-points
Hout(idFsur) == T T     ! zeta               free-surface
Hout(idBath) == T T     ! bath               time-dependent bathymetry

Hout(idTvar) == T T T T ! temp, salt         temperature and salinity

Hout(idpthR) == F       ! z_rho              time-varying depths of RHO-points
Hout(idpthU) == F       ! z_u                time-varying depths of U-points
Hout(idpthV) == F       ! z_v                time-varying depths of V-points
Hout(idpthW) == F       ! z_w                time-varying depths of W-points

! Logical switches (TRUE/FALSE) to activate writing of time-averaged
! fields into AVERAGE output file.

Aout(idUvel) == F       ! u                  3D U-velocity
Aout(idVvel) == F       ! v                  3D V-velocity
Aout(idu3dE) == F       ! u_eastward         3D U-eastward  at RHO-points
Aout(idv3dN) == F       ! v_northward        3D V-northward at RHO-points
Aout(idWvel) == F       ! w                  3D W-velocity
Aout(idOvel) == F       ! omega              omega vertical velocity
Aout(idUbar) == F       ! ubar               2D U-velocity
Aout(idVbar) == F       ! vbar               2D V-velocity
Aout(idu2dE) == F       ! ubar_eastward      2D U-eastward  at RHO-points
Aout(idv2dN) == F       ! vbar_northward     2D V-northward at RHO-points
Aout(idFsur) == F       ! zeta               free-surface
Aout(idBath) == F       ! bath               time-dependent bathymetry

! Logical switches (TRUE/FALSE) to activate writing of time-averaged,
! 2D momentum (ubar,vbar) diagnostic terms into DIAGNOSTIC output file.

Dout(M2rate) == F       ! ubar_accel, ...    acceleration
Dout(M2pgrd) == F       ! ubar_prsgrd, ...   pressure gradient
Dout(M2fcor) == F       ! ubar_cor, ...      Coriolis force
Dout(M2hadv) == F       ! ubar_hadv, ...     horizontal total advection
Dout(M2xadv) == F       ! ubar_xadv, ...     horizontal XI-advection
Dout(M2yadv) == F       ! ubar_yadv, ...     horizontal ETA-advection
Dout(M2hrad) == F       ! ubar_hrad, ...     horizontal total wec_mellor radiation stress
Dout(M2hvis) == F       ! ubar_hvisc, ...    horizontal total viscosity
Dout(M2xvis) == F       ! ubar_xvisc, ...    horizontal XI-viscosity
Dout(M2yvis) == F       ! ubar_yvisc, ...    horizontal ETA-viscosity
Dout(M2sstr) == F       ! ubar_sstr, ...     surface stress
Dout(M2bstr) == F       ! ubar_bstr, ...     bottom stress
Dout(M2hjvf) == F       ! 2D wec_vf horizontal J vortex force
Dout(M2kvrf) == F       ! 2D wec_vf K vortex force
Dout(M2fsco) == F       ! 2D wec_vf coriolis-stokes
```

Tool Output

For Help, press F1   565   1   Read   Ovr   Block   Sync   Rec   Caps

History

Averages

Diagnostics

8) ROMS input files: sandy_ocean.in

TextPad - F:\data\models\COAWST\Projects\Sandy\ocean_sandy.in

File Edit Search View Tools Macros Configure Window Help

Find incrementally

ocean_sandy.in

入力ファイル

```
! Input NetCDF file names, [1:Ngrids].

    GRDNAME == Projects/Sandy/Sandy_roms_grid.nc \
               Projects/Sandy/Sandy_roms_grid_ref3.nc
    ININAME == Projects/Sandy/Sandy_ini.nc \
               Projects/Sandy/Sandy_ini_ref3.nc
    ITLNAME == ocean_itl.nc
    IRPNAME == ocean_irp.nc
    IADNAME == ocean_iad.nc
    FWDNAME == ocean_fwd.nc
    ADSNAME == ocean_ads.nc
    IWININAME == inwave_ini.nc
    IWSWNNAME == point1.spc2d

! Nesting grids connectivity data: contact points information. This
! NetCDF file is special and complex. It is currently generated using
! the script "matlab/grid/contact.m" from the Matlab repository.

    NGCNAME =  Projects/Sandy/Sandy_roms_contact.nc

! Input lateral boundary conditions and climatology file names.  The
! USER has the option to split input data time records into several
! NetCDF files (see prologue instructions above). If so, use a single
! line per entry with a vertical bar (|) symbol after each entry,
! except the last one.

    NCLMFILES == 1                          ! number of climate files
    CLMNAME == Projects/Sandy/Sandy_clm.nc \
               Projects/Sandy/Sandy_clm_ref3.nc

    NBCFILES == 1                           ! number of boundary files
!   BRYNAME == Projects/Sandy/Sandy_bdy.nc
    BRYNAME == Projects/Sandy/coawst_bdy_20121028.nc |
               Projects/Sandy/coawst_bdy_20121029.nc |
               Projects/Sandy/coawst_bdy_20121030.nc |
               Projects/Sandy/coawst_bdy_20121031.nc |
               Projects/Sandy/coawst_bdy_20121101.nc

! Input climatology nudging coefficients file name.

    NUDNAME == ocean_nud.nc

! Input Sources/Sinks forcing (like river runoff) file name.

    SSFNAME == ocean_rivers.nc
```

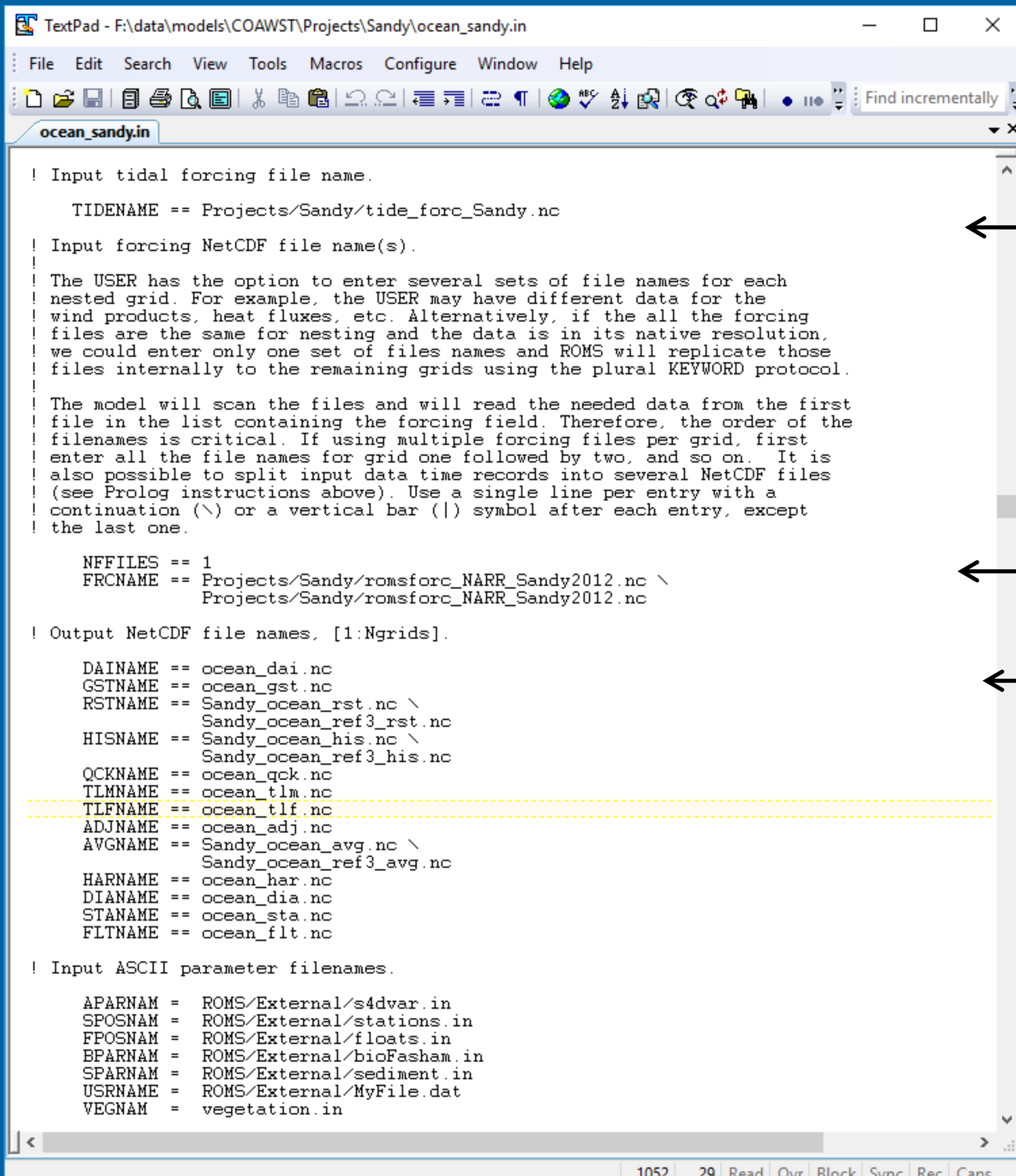
こちらをアクティベート

ディアクティベート

Different in COAWST

448 44 Read Ovr Block Sync Rec Caps

8) ROMS input files: sandy_ocean.in




```
! Input tidal forcing file name.

    TIDENAME == Projects/Sandy/tide_forc_Sandy.nc

! Input forcing NetCDF file name(s).
!
! The USER has the option to enter several sets of file names for each
! nested grid. For example, the USER may have different data for the
! wind products, heat fluxes, etc. Alternatively, if the all the forcing
! files are the same for nesting and the data is in its native resolution,
! we could enter only one set of files names and ROMS will replicate those
! files internally to the remaining grids using the plural KEYWORD protocol.
!
! The model will scan the files and will read the needed data from the first
! file in the list containing the forcing field. Therefore, the order of the
! filenames is critical. If using multiple forcing files per grid, first
! enter all the file names for grid one followed by two, and so on.  It is
! also possible to split input data time records into several NetCDF files
! (see Prolog instructions above). Use a single line per entry with a
! continuation (\) or a vertical bar (|) symbol after each entry, except
! the last one.

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

! Output NetCDF file names, [1:Ngrids].

     DAINAME == ocean_dai.nc
     GSTNAME == ocean_gst.nc
     RSTNAME == Sandy_ocean_rst.nc \
                Sandy_ocean_ref3_rst.nc
     HISNAME == Sandy_ocean_his.nc \
                Sandy_ocean_ref3_his.nc
     QCKNAME == ocean_qck.nc
     TLMNAME == ocean_tlm.nc
     TLFNAME == ocean_tlf.nc
     ADJNAME == ocean_adj.nc
     AVGNAME == Sandy_ocean_avg.nc \
                Sandy_ocean_ref3_avg.nc
     HARNAME == ocean_har.nc
     DIANAME == ocean_dia.nc
     STANAME == ocean_sta.nc
     FLTNAME == ocean_flt.nc

! Input ASCII parameter filenames.

     APARNAM =  ROMS/External/s4dvar.in
     SPOSNAM =  ROMS/External/stations.in
     FPOSNAM =  ROMS/External/floats.in
     BPARNAM =  ROMS/External/bioFasham.in
     SPARNAM =  ROMS/External/sediment.in
     USRNAME =  ROMS/External/MyFile.dat
     VEGNAM  =  vegetation.in
```

Only list tide forc for parent

You can use same forcing file for both grids

his and avg names, etc

9) Build and run: coawst.bash

TextPad - F:\data\models\COAWST\coawst.bash *

File Edit Search View Tools Macros Configure Window Help

Find incrementally

coawst.bash * | ocean_sandy.in

```
############################################################################
# Start of USER definitions area:
#
# Set the CPP option defining the particular application. This will
# determine the name of the ".h" header file with the application
# CPP definitions. Also this will activate the switch file for WW3.
export    COAWST_APPLICATION=SANDY

# Set the ROMS_APPLICATION to be the same as the COAWST_APP.
# Do not change this. We use the COAWST APP for other checks.
export    ROMS_APPLICATION=${COAWST_APPLICATION}

# Set a local environmental variable to define the path to the directories
# where all this project's files are kept.
export    MY_ROOT_DIR=/cygdrive/e/data/models/COAWST
export    MY_PROJECT_DIR=${MY_ROOT_DIR}

# The path to the user's local current ROMS source code.
# If using svn locally, this would be the user's Working Copy Path (WCPATH).
# Note that one advantage of maintaining your source code locally with svn
# is that when working simultaneously on multiple machines (e.g. a local
# workstation, a local cluster and a remote supercomputer) you can checkout
# the latest release and always get an up-to-date customized source on each
# machine. This script is designed to more easily allow for differing paths
# to the code and inputs on differing machines.
export    MY_ROMS_SRC=${MY_ROOT_DIR}/

############################################################################
# WRF : Needs to have the env variable NETCDF set.
#export   NETCDF=${NETCDF_INCDIR}/../
#
############################################################################
# Wave Watch 3: Here we provide 5 environment variables for WW3.
#
# 1) COAWST_WW3_DIR is a pointer to root WW3 code, do not change.
export    COAWST_WW3_DIR=${MY_ROOT_DIR}/WW3
#
# 2) WWATCH3_NETCDF can be NC3 or NC4. We need NC4 for COAWST. do not change.
export    WWATCH3_NETCDF=NC4
#
# 3) WWATCH_ENV points to WW3 environment listing. do not change.
export    WWATCH_ENV=${COAWST_WW3_DIR}/wwatch.env
#
# 4) NETCDF_CONFIG is needed by WW3. You need to set this:
#export    NETCDF_CONFIG=${NETCDF_LIBDIR}/../bin/nc-config
#     This may require nf-config, depending on your system.
export    NETCDF_CONFIG=/share/apps/netcdf-4.1.3_intel-2011.4.191/bin/nc-config
#
# 5) WW3_SWITCH_FILE is like cpp options for WW3. You need to create it and
#     list the name here.  You need to have COAWST listed in the switch file.
 export   WW3_SWITCH_FILE=switch_sandy

############################################################################
# Compiler selections.
#
# Set path of the directory containing makefile configuration (*.mk) files.
```

132 1 Read Ovr Block Sync Rec Caps

9) Build and run: coawst.bash

TextPad - F:\data\models\COAWST\coawst.bash *

```
########################################################################
# Compiler selections.
#
# Set path of the directory containing makefile configuration (*.mk) files.
# The user has the option to specify a customized version of these files
# in a different directory than the one distributed with the source code,
# ${MY_ROMS_SRC}/Compilers. If this is the case, the you need to keep
# these configurations files up-to-date.

#export          COMPILERS=${MY_ROMS_SRC}/Compilers

# Set tunable CPP options.
#
# Sometimes it is desirable to activate one or more CPP options to run
# different variants of the same application without modifying its header
# file. If this is the case, specify each options here using the -D syntax.
# Notice also that you need to use shell's quoting syntax to enclose the
# definition.  Both single or double quotes work. For example,
#
#export      MY_CPP_FLAGS="-DAVERAGES"
#export      MY_CPP_FLAGS="${MY_CPP_FLAGS} -DDEBUGGING"
#
# can be used to write time-averaged fields. Notice that you can have as
# many definitions as you want by appending values.

#export      MY_CPP_FLAGS="-D"

# Other user defined environmental variables. See the ROMS makefile for
# details on other options the user might want to set here. Be sure to
# leave the switches meant to be off set to an empty string or commented
# out. Any string value (including off) will evaluate to TRUE in
# conditional if-statements.

 export            USE_MPI=on             # distributed-memory parallelism
 export         USE_MPIF90=on             # compile with mpif90 script
#export          which_MPI=mpich          # compile with MPICH library
#export          which_MPI=mpich2         # compile with MPICH2 library
 export          which_MPI=openmpi        # compile with OpenMPI library

#export         USE_OpenMP=on             # shared-memory parallelism

 export               FORT=ifort
#export               FORT=gfortran
#export               FORT=pgi

 export          USE_DEBUG=               # use Fortran debugging flags
 export          USE_LARGE=               # activate 64-bit compilation
 export        USE_NETCDF4=               # compile with NetCDF-4 library
#export   USE_PARALLEL_IO=on              # Parallel I/O with Netcdf-4/HDF5

#export        USE_MY_LIBS=on             # use my library paths below

# There are several MPI libraries available. Here, we set the desired
# "mpif90" script to use during compilation. This only works if the make
# configuration file (say, Linux-pgi.mk) in the "Compilers" directory
```

Makefile determines the Compilers file from:

1) uname –s
2) the FORT value

For example, here it will be Compilers/Linux-ifort.mk

9) Build and run: coawst.bash

TextPad - F:\data\models\COAWST\coawst.bash *

File Edit Search View Tools Macros Configure Window Help

coawst.bash * | ocean_sandy.in

```
#############################################################################
# Header and other source directories selections.
#
# The rest of this script sets the path to the users header file and
# analytical source files, if any. See the templates in User/Functionals.
#
# If applicable, use the MY_ANALYTICAL_DIR directory to place your
# customized biology model header file (like fennel.h, nemuro.h, ecosim.h,
# etc).

#  export     MY_HEADER_DIR=${MY_PROJECT_DIR}/ROMS/Include
#  export MY_ANALYTICAL_DIR=${MY_PROJECT_DIR}/ROMS/Functionals
   export     MY_HEADER_DIR=${MY_PROJECT_DIR}/Projects/Inlet_test/Coupled
   export MY_ANALYTICAL_DIR=${MY_PROJECT_DIR}/Projects/Inlet_test/Coupled

# Put the binary to execute in the following directory.

# export            BINDIR=${MY_PROJECT_DIR}
  export            BINDIR=./

# Put the f90 files in a project specific Build directory to avoid conflict
# with other projects.

# export       SCRATCH_DIR=${MY_PROJECT_DIR}/Build
  export       SCRATCH_DIR=./Build

# Go to the users source directory to compile. The options set above will
# pick up the application-specific code from the appropriate place.

#############################################################################
# End of USER definitions area.  You really should not change anything
# down here.
#
 cd ${MY_ROMS_SRC}

#---------------------------------------------------------------------------
# Compile.
#---------------------------------------------------------------------------
# Remove build directory.

if [ $clean -eq 1 ]; then
  make clean
fi

# Compile (the binary will go to BINDIR set above).

  export WRF_DIR=${MY_ROMS_SRC}/WRF
if [ $cleanwrf -eq 1 ]; then
  make wrfclean
  cd ${MY_ROMS_SRC}
fi
make wrf

if [ $cleanww3 -eq 1 ]; then
  make ww3clean
  cd ${MY_ROMS_SRC}
fi
make ww3

if [ $dprint -eq 1 ]; then
  make $debug
else
  if [ $parallel -eq 1 ]; then
    make $NCPUS
  else
    make
  fi
fi
```

193 28 Read Ovr Block Sync Rec Caps

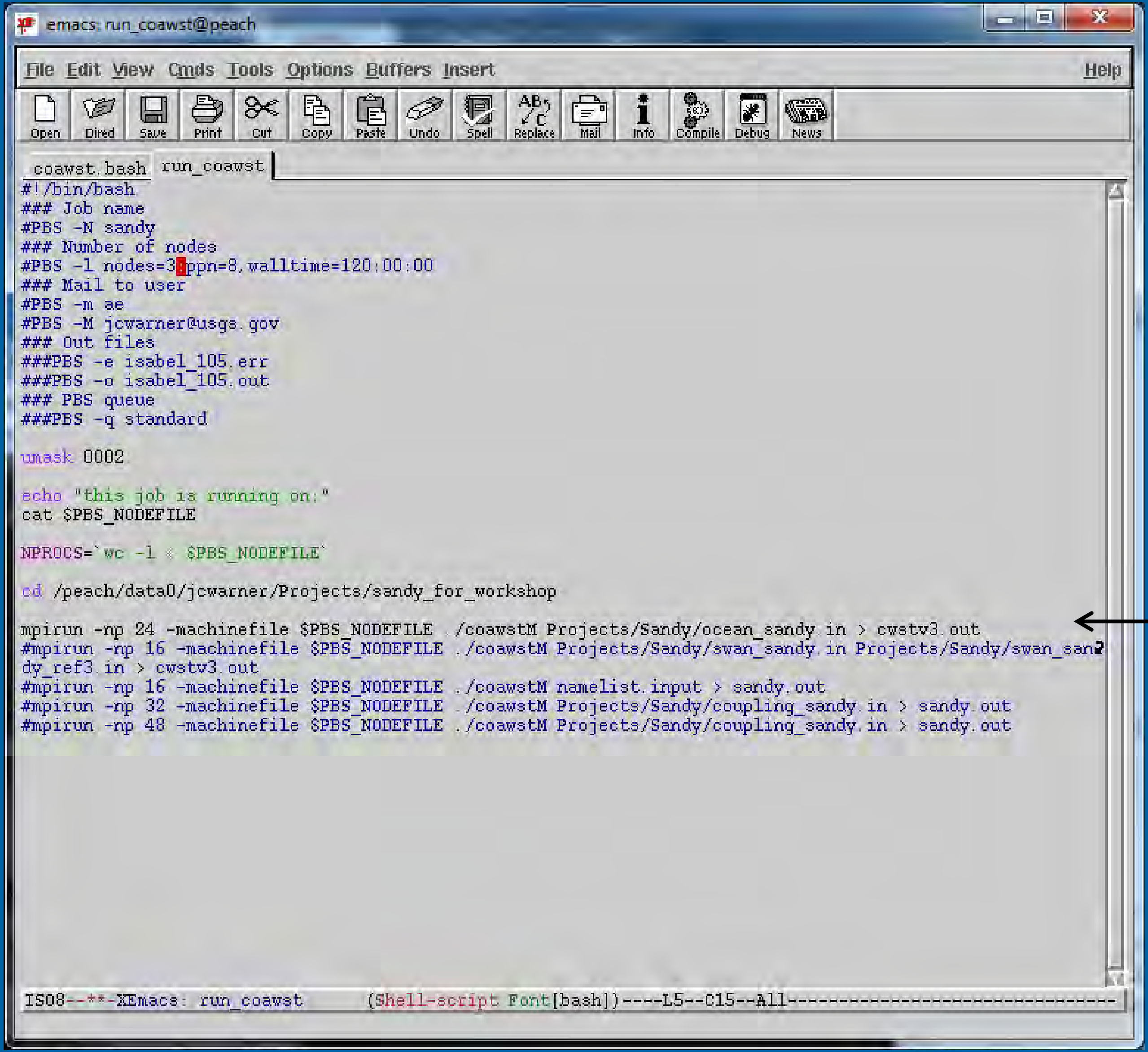
Set the header and ana paths

Customize the build here

build options

To build the code use
./coawst.bash –j N
this should make the
coawstM

9) Build and run: run_coawst




```
#!/bin/bash
### Job name
#PBS -N sandy
### Number of nodes
#PBS -l nodes=3:ppn=8,walltime=120:00:00
### Mail to user
#PBS -m ae
#PBS -M jcwarner@usgs.gov
### Out files
###PBS -e isabel_105.err
###PBS -o isabel_105.out
### PBS queue
###PBS -q standard

umask 0002

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

NPROCS=`wc -l < $PBS_NODEFILE`

cd /peach/data0/jcwarner/Projects/sandy_for_workshop

mpirun -np 24 -machinefile $PBS_NODEFILE ./coawstM Projects/Sandy/ocean_sandy.in > cwstv3.out
#mpirun -np 16 -machinefile $PBS_NODEFILE ./coawstM Projects/Sandy/swan_sandy.in Projects/Sandy/swan_san
dy_ref3.in > cwstv3.out
#mpirun -np 16 -machinefile $PBS_NODEFILE ./coawstM namelist.input > sandy.out
#mpirun -np 32 -machinefile $PBS_NODEFILE ./coawstM Projects/Sandy/coupling_sandy.in > sandy.out
#mpirun -np 48 -machinefile $PBS_NODEFILE ./coawstM Projects/Sandy/coupling_sandy.in > sandy.out
```

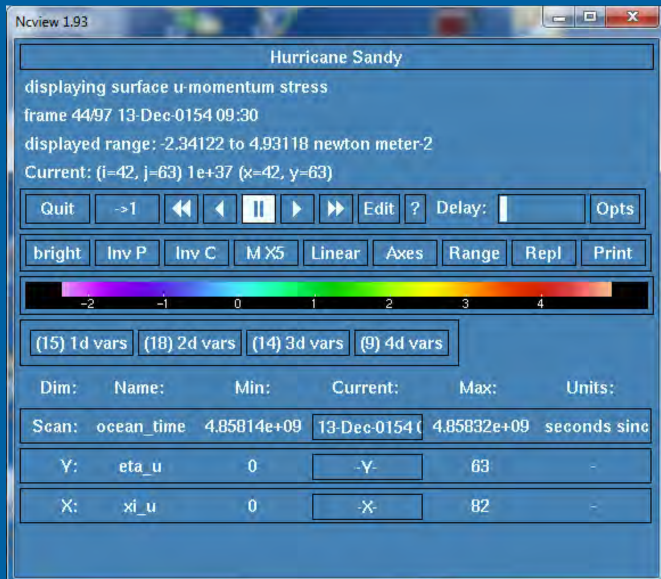
```
ISO8--**-XEmacs: run_coawst      (Shell-script Font[bash])----L5--C15--All-----------------------------
```

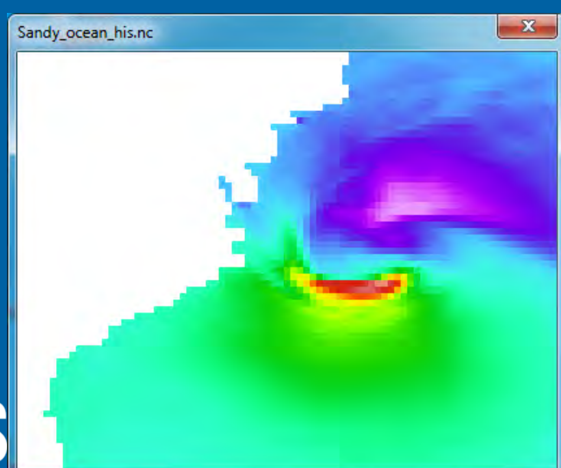
Make sure the –np X number of procs is equal to the NtileI * NtileJ in the ocean.in file.

To run just ROMS, the mpirun should point to the ocean.in

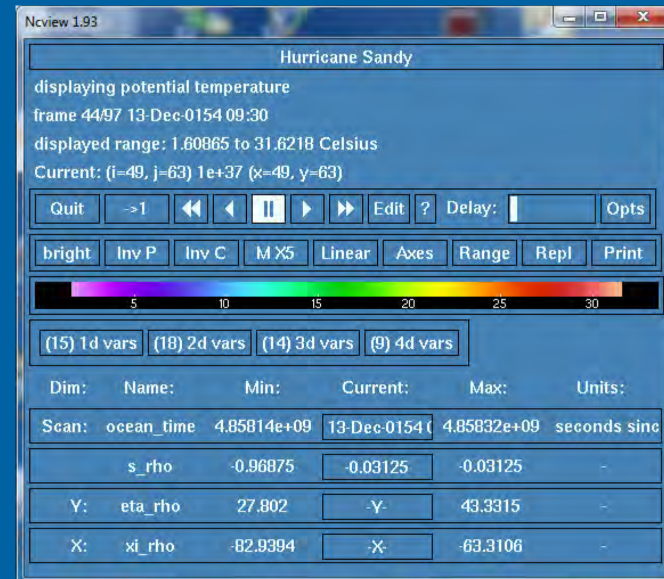
10) Output

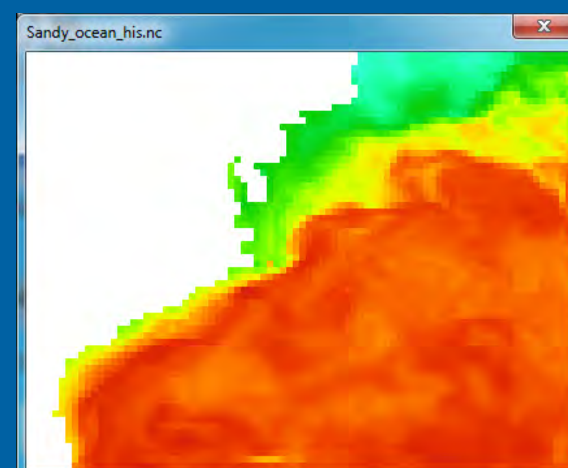
sustr





temp (surface)





It’s that easy. I don’t understand why people have problems.