Using SCHISM

Joseph Zhang

Primer

- Online manual and wiki resource: www.schism.wiki
  - There are beta_notes, sample files etc in each source code bundle
- It helps if you know a few programming/scripting languages: python, matlab, FORTRAN etc for pre- and post-processing needs
- Bare minimum
  - Generate a hgrid.gr3 with simplest possible b.c. (e.g., no open bnd)
  - Run SCHISM in 2D config: manning.gr3, param.in, bctides.in, vgrid.in (2 levels)
  - Pre-processing with ipre=1 first (with 1 CPU) to catch grid and other issues
  - Then proceed to more complex set-ups
  - **Establish a good work flow and be willing to revise the grid**

Sample run directory

- /sciclone/home10/yinglong/vims20/schism_verification_tests/Test_CORIE/TMP% ls -L

albedo.gr3 CORIE_TMP.o3660379 drag.gr3 fort.11 hotstart.nc postpros.pl run_bora show_schism_nc.m tmp.xyuv watertype.gr3
bctides.in coriolis.out err2.out fort.17 mirror.out pschism_WHIRLWIND_Intel_VL run_whirlwind_openmpi sidecenters.gr3 total.dat
windrot_geo2proj.gr3 bnd.xy date.in estuary.gr3 fort.33 SAL_nudge.gr3 TEM_1.th total_TR.dat
centers.gr3 diffmax.gr3 flux.dat hgrid.gr3 **outputs/** read.in SAL_nu.nc TEM_nudge.gr3 tvd.prop
CORIE_TMP.e3660379 diffmin.gr3 flux.th hgrid.ll param.in README **sflux/** TEM_nu.nc vgrid.in

- /sciclone/home10/yinglong/vims20/schism_verification_tests/Test_CORIE/TMP% ls -L *.gr3 *.ll *.ic

albedo.gr3 diffmax.gr3 drag.gr3 hgrid.gr3 SAL_nudge.gr3 TEM_nudge.gr3 windrot_geo2proj.gr3
centers.gr3 diffmin.gr3 estuary.gr3 hgrid.ll sidecenters.gr3 watertype.gr3

- /sciclone/home10/yinglong/vims20/schism_verification_tests/Test_CORIE/TMP% ls -L *.in

bctides.in param.in vgrid.in

- /sciclone/home10/yinglong/vims20/schism_verification_tests/Test_CORIE/TMP% ls -L *.th*

flux.th TEM_1.th <- ASCII

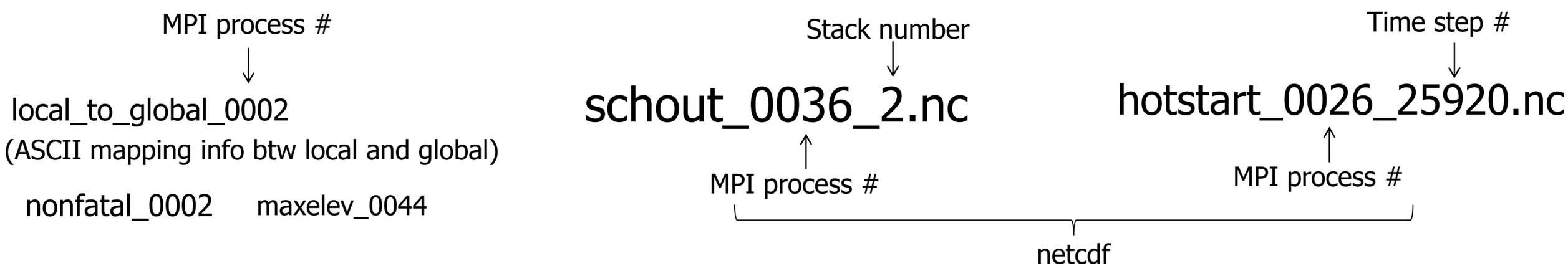
- /sciclone/home10/yinglong/vims20/schism_verification_tests/Test_CORIE/TMP% ls -L *.nc

hotstart.nc SAL_nu.nc TEM_nu.nc

Sample output directory


```
/sciclone/home10/yinglong/vims20/schism_verification_tests/Test_CORIE/TMP% ls outputs/
hotstart_0013_14400.nc  hotstart_0026_20160.nc  hotstart_0039_26880.nc  maxelev_0041        schout_0010_5.nc   schout_0023_27.nc  schout_0036_21.nc
hotstart_0013_15360.nc  hotstart_0026_21120.nc  hotstart_0039_2880.nc   maxelev_0042        schout_0010_6.nc   schout_0023_28.nc  schout_0036_22.nc
global_to_local.prop    hotstart_0013_16320.nc  hotstart_0026_22080.nc  hotstart_0039_3840.nc  maxelev_0043     schout_0010_7.nc   schout_0023_29.nc  schout_0036_23.nc
hotstart_0000_10560.nc  hotstart_0013_17280.nc  hotstart_0026_23040.nc  hotstart_0039_4800.nc  maxelev_0044     schout_0010_8.nc   schout_0023_2.nc   schout_0036_24.nc
hotstart_0000_11520.nc  hotstart_0013_18240.nc  hotstart_0026_24000.nc  hotstart_0039_5760.nc  maxelev_0045     schout_0010_9.nc   schout_0023_3.nc   schout_0036_25.nc
hotstart_0000_12480.nc  hotstart_0013_19200.nc  hotstart_0026_24960.nc  hotstart_0039_6720.nc  maxelev_0046     schout_0011_10.nc  schout_0023_4.nc   schout_0036_26.nc
hotstart_0000_13440.nc  hotstart_0013_1920.nc   hotstart_0026_25920.nc  hotstart_0039_7680.nc  maxelev_0047     schout_0011_11.nc  schout_0023_5.nc   schout_0036_27.nc
hotstart_0000_14400.nc  hotstart_0013_20160.nc  hotstart_0026_26880.nc  hotstart_0039_8640.nc  nonfatal_0000    schout_0011_12.nc  schout_0023_6.nc   schout_0036_28.nc
hotstart_0000_15360.nc  hotstart_0013_21120.nc  hotstart_0026_2880.nc   hotstart_0039_9600.nc  nonfatal_0001    schout_0011_13.nc  schout_0023_7.nc   schout_0036_29.nc
hotstart_0000_16320.nc  hotstart_0013_22080.nc  hotstart_0026_3840.nc   hotstart_0039_960.nc   nonfatal_0002    schout_0011_14.nc  schout_0023_8.nc   schout_0036_2.nc
hotstart_0000_17280.nc  hotstart_0013_23040.nc  hotstart_0026_4800.nc   hotstart_0040_10560.nc nonfatal_0003    schout_0011_15.nc  schout_0023_9.nc   schout_0036_3.nc
hotstart_0000_18240.nc  hotstart_0013_24000.nc  hotstart_0026_5760.nc   hotstart_0040_11520.nc nonfatal_0004    schout_0011_16.nc  schout_0024_10.nc  schout_0036_4.nc
local_to_global_0000 ....
```

- SCHISM code generally outputs results per MPI rank (not per openMP thread)
  - Exception: station outputs
- Post-proc scripts are used to 'gather' all ranks into a 'global' output (either ASCII or netcdf4)
  - V5.5.0 needs netcdf 4.4* and newer



Pre- and post-combined outputs

- Netcdf outputs

schout_0000_2.nc schout_0001_2.nc …… schout_0399_2.nc

combine_output10.f90

schout_2.nc (global output)

Similarly for hotstart

- ASCII

maxelev_0000 maxelev_0001 …… maxelev_0399

combine_gr4.f90

maxelev.gr3 (max elevation over all time steps)

Main diagnostic output (mirror.out, by rank 0)


```
Run begins at 20171026, 082037.556
 You are using baroclinic model
 # of tracers in each module:          1          1          0          0
           0          0          0          0          0          0
           0
 Total # of tracers=           2
 Index ranges of each module:          1          1          2          2
           3          2          3          2          3          2
           3          2          3          2          3          2
           3          2          3          2          3          2
 # of global outputs=          29
 done reading param.in; s2_mxnbt in param.in =   3.00000000000000
 lhas_quad= F
 mnei, mnei_p =          10          11
 lhas_quad= F

Global Grid Size (ne,np,ns,nvrt):      38960     20641     59615        54

**********Augmented Subdomain Sizes**********
 rank     nea      ne     neg     nea2     neg2     npa      np     npg     npa2     npg2     nsa      ns     nsg     nsa2     nsg2
    0    1071     821     250     1071        0     634     521     113      634        0    1701    1345     356     1701        0
    1    1055     828     227     1055        0     611     514      97      611        0    1664    1349     315     1664        0
    2     981     794     187      981        0     588     500      88      588        0    1566    1293     273     1566        0
.....................
```

```
 heat budge model completes...
 time stepping begins...           1       26880
 heat budge model completes...
 done adjusting wind stress ...
 done flow b.c.
 done MYG-UB...
 done hvis...
 done backtracking
 done 1st preparation
 done 2nd preparation
 done solver; etatot=  0.939833361180156     ; average |eta|= 4.553235604767967E-005
 done solving momentum eq...
 done solving w
 done tracer transport...
 done solving transport equation
 done recomputing levels...
 done density and flux calculation...
TIME STEP=           1;  TIME=           90.000000
 heat budge model completes...
 done adjusting wind stress ...
 done flow b.c.
```

Sample netcdf output file (pre-combine)


```
netcdf schout_0011_4 {
dimensions:
        nSCHISM_hgrid_node = 516 ;
        nSCHISM_hgrid_face = 813 ;
        nSCHISM_hgrid_edge = 1326 ;
        nMaxSCHISM_hgrid_face_nodes = 4 ;
        nSCHISM_vgrid_layers = 54 ;
        one = 1 ;
        two = 2 ;
        time = UNLIMITED ; // (96 currently)
variables:
        float time(time) ;
                time:i23d = 0 ;
        float wetdry_elem(time, nSCHISM_hgrid_face) ;
                wetdry_elem:i23d = 4 ;
                wetdry_elem:ivs = 1 ;
        float zcor(time, nSCHISM_hgrid_node, nSCHISM_vgrid_layers) ;
                zcor:i23d = 2 ;
                zcor:ivs = 1 ;
        float elev(time, nSCHISM_hgrid_node) ;
                elev:i23d = 1 ;
                elev:ivs = 1 ;
        float wind_speed(time, nSCHISM_hgrid_node, two) ;
                wind_speed:i23d = 1 ;
                wind_speed:ivs = 2 ;
        float vertical_velocity(time, nSCHISM_hgrid_node, nSCHISM_vgrid_layers) ;
                vertical_velocity:i23d = 2 ;
                vertical_velocity:ivs = 1 ;
        float temp(time, nSCHISM_hgrid_node, nSCHISM_vgrid_layers) ;
                temp:i23d = 2 ;
                temp:ivs = 1 ;
        float salt(time, nSCHISM_hgrid_node, nSCHISM_vgrid_layers) ;
                salt:i23d = 2 ;
                salt:ivs = 1 ;
        float diffusivity(time, nSCHISM_hgrid_node, nSCHISM_vgrid_layers) ;
                diffusivity:i23d = 2 ;
                diffusivity:ivs = 1 ;
        float hvel(time, nSCHISM_hgrid_node, nSCHISM_vgrid_layers, two) ;
                hvel:i23d = 2 ;
                hvel:ivs = 2 ;
data:

 time = 260100, 261000, 261900, 262800, 263700, 264600, 265500, 266400,
    267300, 268200, 269100, 270000, 270900, 271800, 272700, 273600, 274500,
    275400, 276300, 277200, 278100, 279000, 279900, 280800, 281700, 282600,
    283500, 284400, 285300, 286200, 287100, 288000, 288900, 289800, 290700,
    291600, 292500, 293400, 294300, 295200, 296100, 297000, 297900, 298800,
    299700, 300600, 301500, 302400, 303300, 304200, 305100, 306000, 306900,
    307800, 308700, 309600, 310500, 311400, 312300, 313200, 314100, 315000,
```

Connectivity table is found in local_to_global_*

Sample netcdf output file (post-combine)


```
netcdf schout_3 {
dimensions:
        nSCHISM_hgrid_node = 20641 ;
        nSCHISM_hgrid_face = 38960 ;
        nSCHISM_hgrid_edge = 59615 ;
        nMaxSCHISM_hgrid_face_nodes = 4 ;
        nSCHISM_vgrid_layers = 54 ;
        one = 1 ;
        two = 2 ;
        sigma = 37 ;
        nz = 17 ;
        time = UNLIMITED ; // (96 currently)
variables:
        double time(time) ;
                time:long_name = "Time" ;
                time:units = "seconds since 2002-30-04  00:00:00 PST" ;
                time:base_date = "04/30/2002 00:00:00 PST" ;
                time:standard_name = "time" ;
        int SCHISM_hgrid(one) ;
                SCHISM_hgrid:long_name = "Topology data of 2d unstructured mesh" ;
                SCHISM_hgrid:topology_dimension = 2 ;
                SCHISM_hgrid:cf_role = "mesh_topology" ;
                SCHISM_hgrid:node_coordinates = "SCHISM_hgrid_node_x
SCHISM_hgrid_node_y" ;
                SCHISM_hgrid:face_node_connectivity = "SCHISM_hgrid_face_nodes" ;
                SCHISM_hgrid:edge_coordinates = "SCHISM_hgrid_edge_x
SCHISM_hgrid_edge_y" ;
                SCHISM_hgrid:face_coordinates = "SCHISM_hgrid_face_x
SCHISM_hgrid_face_y" ;
                SCHISM_hgrid:edge_node_connectivity = "SCHISM_hgrid_edge_nodes" ;
        int Mesh3D(one) ;
```

Connectivity table

```
        int SCHISM_hgrid_face_nodes(nSCHISM_hgrid_face,
nMaxSCHISM_hgrid_face_nodes) ;
                SCHISM_hgrid_face_nodes:long_name = "Horizontal Element Table" ;
                SCHISM_hgrid_face_nodes:units = "non-dimensional" ;
                SCHISM_hgrid_face_nodes:cf_role = "face_node_connectivity" ;
                SCHISM_hgrid_face_nodes:start_index = 1 ;
                SCHISM_hgrid_face_nodes:_FillValue = -99999 ;
        int SCHISM_hgrid_edge_nodes(nSCHISM_hgrid_edge, two) ;
                SCHISM_hgrid_edge_nodes:long_name = "Map every edge to the two
nodes that it connects" ;
                SCHISM_hgrid_edge_nodes:units = "non-dimensional" ;
                SCHISM_hgrid_edge_nodes:cf_role = "edge_node_connectivity" ;
                SCHISM_hgrid_edge_nodes:start_index = 1 ;
        float SCHISM_hgrid_node_x(nSCHISM_hgrid_node) ;
                SCHISM_hgrid_node_x:long_name = "node x-coordinate" ;
                SCHISM_hgrid_node_x:standard_name = "projection_x_coordinate" ;
                SCHISM_hgrid_node_x:units = "m or degrees" ;
                SCHISM_hgrid_node_x:mesh = "SCHISM_hgrid" ;
        float SCHISM_hgrid_node_y(nSCHISM_hgrid_node) ;
                SCHISM_hgrid_node_y:long_name = "node y-coordinate" ;
                SCHISM_hgrid_node_y:standard_name = "projection_y_coordinate" ;
                SCHISM_hgrid_node_y:units = "m or degrees" ;
                SCHISM_hgrid_node_y:mesh = "SCHISM_hgrid" ;
```

*We are still tweaking the arrays for UGRID standard and also for visIT so final format may be slightly different

Sample netcdf output file (post-combine)


```
int node_bottom_index(nSCHISM_hgrid_node) ;
        node_bottom_index:long_name = "bottom level index at each node" ;
        node_bottom_index:units = "non-dimensional" ;
        node_bottom_index:mesh = "SCHISM_hgrid" ;
        node_bottom_index:location = "node" ;
        node_bottom_index:start_index = 1 ;
float SCHISM_hgrid_face_x(nSCHISM_hgrid_face) ;
        SCHISM_hgrid_face_x:long_name = "x_coordinate of 2D mesh face" ;
        SCHISM_hgrid_face_x:standard_name = 391266304 ;
        SCHISM_hgrid_face_x:units = "m" ;
        SCHISM_hgrid_face_x:mesh = "SCHISM_hgrid" ;
float SCHISM_hgrid_face_y(nSCHISM_hgrid_face) ;
        SCHISM_hgrid_face_y:long_name = "y_coordinate of 2D mesh face" ;
        SCHISM_hgrid_face_y:standard_name = "projection_y_coordinate" ;
        SCHISM_hgrid_face_y:units = "m" ;
        SCHISM_hgrid_face_y:mesh = "SCHISM_hgrid" ;
int ele_bottom_index(nSCHISM_hgrid_face) ;
        ele_bottom_index:long_name = "bottom level index at each element" ;
        ele_bottom_index:units = "non-dimensional" ;
        ele_bottom_index:mesh = "SCHISM_hgrid" ;
        ele_bottom_index:location = "elem" ;
        ele_bottom_index:start_index = 1 ;
float SCHISM_hgrid_edge_x(nSCHISM_hgrid_edge) ;
        SCHISM_hgrid_edge_x:long_name = "x_coordinate of 2D mesh edge" ;
        SCHISM_hgrid_edge_x:standard_name = "projection_x_coordinate" ;
        SCHISM_hgrid_edge_x:units = "m" ;
        SCHISM_hgrid_edge_x:mesh = "SCHISM_hgrid" ;
```

```
float SCHISM_hgrid_edge_y(nSCHISM_hgrid_edge) ;
        SCHISM_hgrid_edge_y:long_name = "y_coordinate of 2D mesh edge" ;
        SCHISM_hgrid_edge_y:standard_name = "projection_y_coordinate" ;
        SCHISM_hgrid_edge_y:units = "m" ;
        SCHISM_hgrid_edge_y:mesh = "SCHISM_hgrid" ;
 int edge_bottom_index(nSCHISM_hgrid_edge) ;
        edge_bottom_index:long_name = "bottom level index at each edge" ;
        edge_bottom_index:units = "non-dimensional" ;
        edge_bottom_index:mesh = "SCHISM_hgrid" ;
        edge_bottom_index:location = "edge" ;
        edge_bottom_index:start_index = 1 ;
 float depth(nSCHISM_hgrid_node) ;
        depth:long_name = "Bathymetry" ;
        depth:units = "meters" ;
        depth:positive = "down" ;
        depth:mesh = "SCHISM_hgrid" ;
        depth:location = "node" ;
 float sigma(sigma) ;
        sigma:long_name = "S coordinates at whole levels" ;
        sigma:units = "1" ;
        sigma:standard_name = "ocean_s_coordinate" ;
        sigma:positive = "up" ;
```

*These arrays are static

Sample netcdf output file (post-combine)


```
int coordinate_system_flag(one) ;
float minimum_depth(one) ;
float sigma_h_c(one) ;
        sigma_h_c:long_name = "ocean_s_coordinate h_c constant" ;
        sigma_h_c:units = "meters" ;
        sigma_h_c:positive = "down" ;
float sigma_theta_b(one) ;
        sigma_theta_b:long_name = "ocean_s_coordinate theta_b
constant" ;
float sigma_theta_f(one) ;
        sigma_theta_f:long_name = "ocean_s_coordinate theta_f
constant" ;
float sigma_maxdepth(one) ;
        sigma_maxdepth:long_name = "ocean_s_coordinate
maximum depth cutoff (mixed s over z boundary)" ;
        sigma_maxdepth:units = "meters" ;
        sigma_maxdepth:positive = "down" ;
float Cs(sigma) ;
        Cs:long_name = "Function C(s) at whole levels" ;
        Cs:units = "non-dimensional" ;
        Cs:positive = "up" ;
float z(nz) ;
        z:long_name = "Z coordinates at whole levels" ;
        z:units = "meters" ;
        z:positive = "up" ;
```

```
float wetdry_elem(time, nSCHISM_hgrid_face) ;
        wetdry_elem:missing_value = 9.96921e+36f ;
        wetdry_elem:mesh = "SCHISM_hgrid" ;
        wetdry_elem:data_horizontal_center = "elem" ;
        wetdry_elem:data_vertical_center = "full" ;
        wetdry_elem:i23d = 4 ;
        wetdry_elem:ivs = 1 ;
float zcor(time, nSCHISM_hgrid_node, nSCHISM_vgrid_layers) ;
        zcor:missing_value = 9.96921e+36f ;
        zcor:mesh = "SCHISM_hgrid" ;
        zcor:data_horizontal_center = "node" ;
        zcor:data_vertical_center = "full" ;
        zcor:i23d = 2 ;
        zcor:ivs = 1 ;
float elev(time, nSCHISM_hgrid_node) ;
        elev:missing_value = 9.96921e+36f ;
        elev:mesh = "SCHISM_hgrid" ;
        elev:data_horizontal_center = "node" ;
        elev:data_vertical_center = "full" ;
        elev:i23d = 1 ;
        elev:ivs = 1 ;
float wind_speed(time, nSCHISM_hgrid_node, two) ;
```

↑
Time series

*zcor: z-coordinates at nodes and each time step; junk values if below bottom. Also used to infer wetting and drying

Pre- & post-processing and visualization

- Combine binary outputs (MPI): FORTRAN & perl script in Utility/Combining_Scripts/)
- Visualization
  - xmgredit: for inputs; can handle mixed grids
  - python: for in/outputs; see DWR tools
  - VisIT: the best & most comprehensive tool for visualizing outputs; up to date and can viz shaved cells (**combine_outputs10.f90 plugin still has some issues, mostly related to side & elem centered arrays. These will be fixed soon**)
  - matlab: for in/outputs; not as efficient as VisIT
  - GIS and google earth based: specialty tools; needs some customization
- Extracting time series: vertical profiles, horizontal slabs, transects
  - station outputs (easiest): be careful not to abuse; only 'online'
  - FORTRAN scripts: must wait until outputs are combined, but more flexible ('offline')
  - Linear interpolation in 3D (and time for some)
  - calculation of *z*-coordinates consistent with SCHISM code
  - beware wet/dry interpolation
- Residuals, harmonics analysis

SCHISM input types: glossary

- *.gr3: grid-like files (node centered): only hgrid.gr3 has b.c. info at the end
- hgrid.ll: lon/lat form of horizontal grid (.gr3 format)
- *.th*: time history (b.c.); ASCII or nc
- *.ic: initial condition for elev, tracers (ASCII; most of them actually of .gr3 format)
- *.prop: element centered property (ASCII)
- sflux/: atmos. Forcings (netcdf)
- *.in: vgrid.in (vertical grid); param.in (parameters); bctides.in (b.c.) [ASCII]
- *.nc: hotstart; tracer nudging inputs; b.c. time history inputs (type '±4' or '5')
- Modules: gotmturb.inp, wwminput.nml, ice.nml ….
- Bare minimum (mandatory) for pre-processing (ipre=1)
  - hgrid.gr3 (and hgrid.ll if you are using some modules): no need to have correct bathymetry & boundaries yet
  - vgrid.in (1 layer for 2D for simplicity)
  - param.in
  - bctides.in (easiest is to use no open bnd)
  - drag.gr3/rough.gr3/manning.gr3 (to specify bottom friction, depending on the *bfric* specified in param.in)
- Pre-processing
  - Prepare the mandatory input files, and set ipre=1 in param.in
  - Run SCHISM with 1 CPU only
  - Check fort.11 for fatal errors (e.g. grid issues)
  - Rectify grid/inputs errors and repeat until pre-processing is successful

Details of model setup

.gr3 and .ll files

Your comments

44343 24025

nodes

1	-90.4293	30.1689	0.30
2	-90.4313	30.1625	0.30
3	-90.4327	30.1559	0.20
4	-90.4320	30.1498	0.20

.....

elements

1	3	1	2	3
2	3	2	4	5
3	3	15943	16197	15942

.....

Boundary info (hgrid.gr3 only)

2 = Number of open boundaries
69 = Total number of open boundary nodes
4 = Number of nodes for open boundary 1
1
2
3
4
.....
12 = number of land boundaries
1756 = Total number of land boundary nodes
737 0 = Number of nodes for land boundary 1
29368
29421
29420
29467

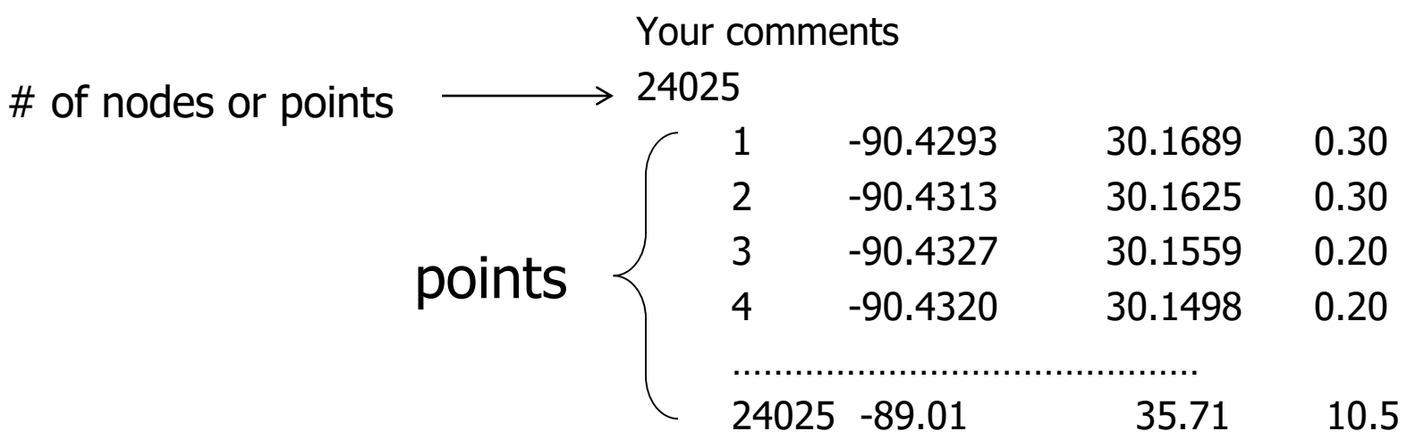
.....

Except in hgrid.gr3 and hgrid.ll, only depth info is used

*There is a simple perl converter between .2dm and .gr3

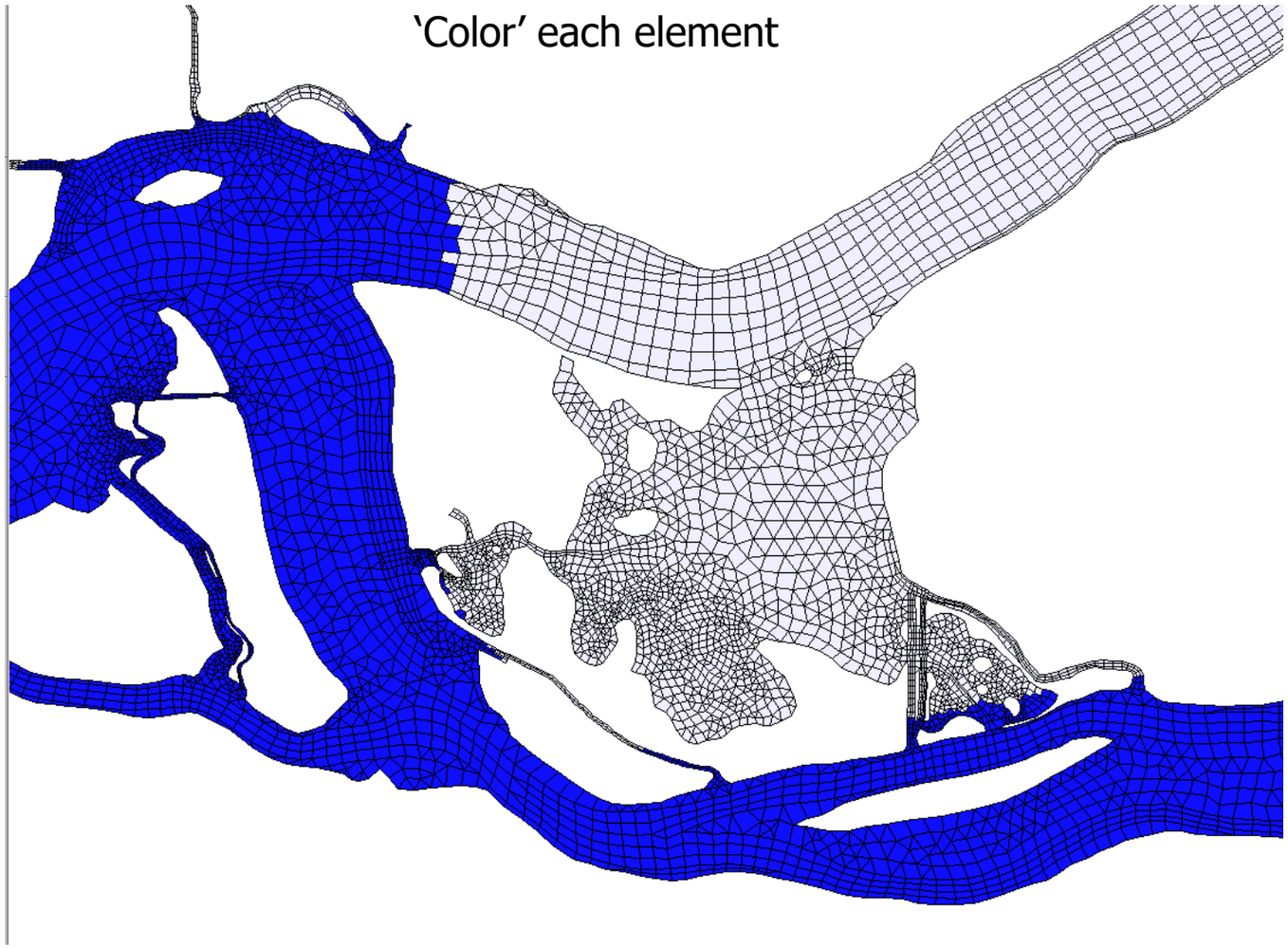
.bp format

- Not used in SCHISM per se, but used in some scripts
- Basically a .gr3 file without connectivity
- Alternative: .sta



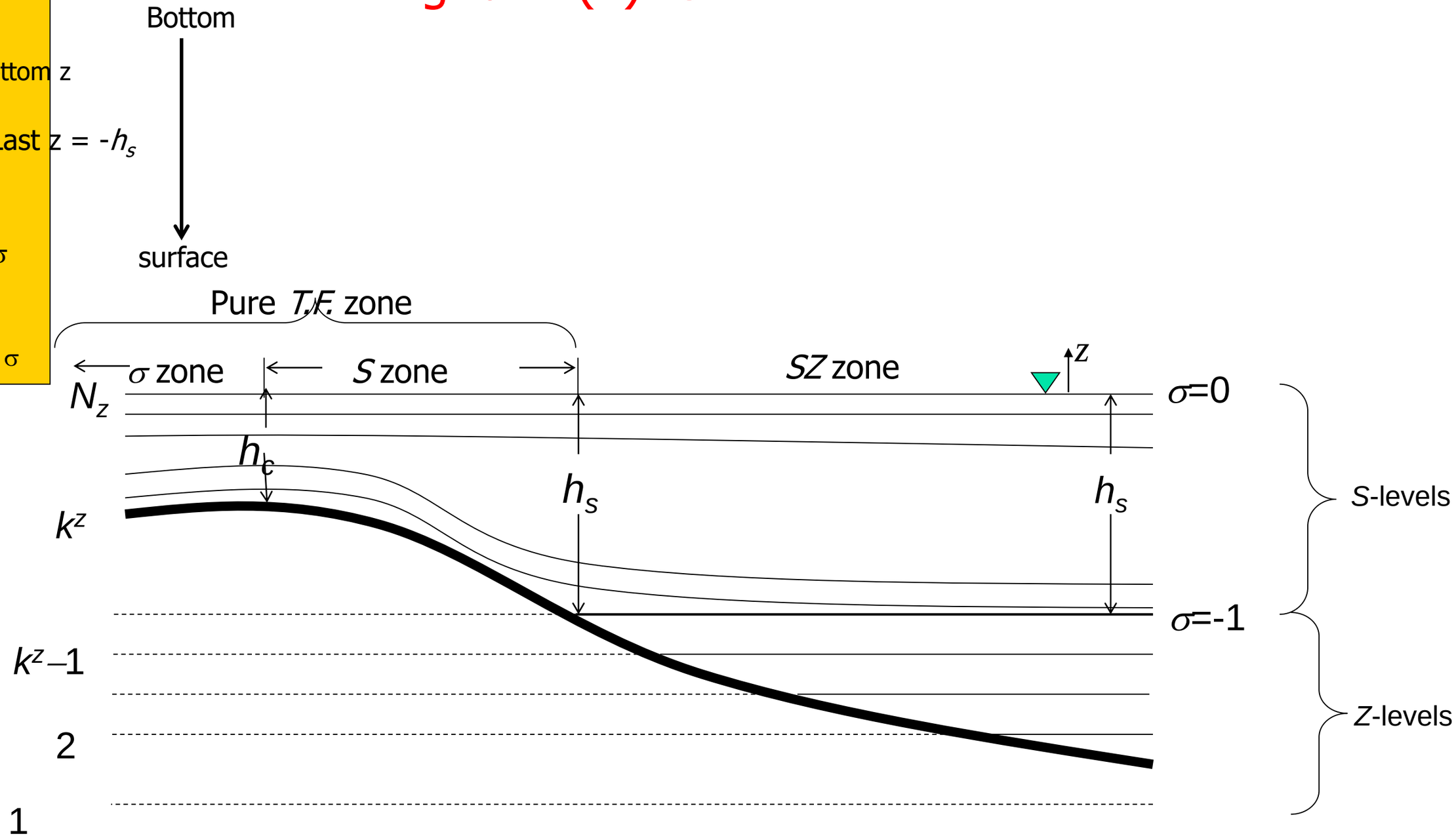
.prop format


```
 1  0.0000000E+00
 2  0.0000000E+00
 3  0.0000000E+00
 4  0.0000000E+00
 5  0.0000000E+00
 6  0.0000000E+00
 7  0.0000000E+00
 8  0.0000000E+00
 9  1.0000000E+00
10  1.0000000E+00
11  0.0000000E+00
12  0.0000000E+00
13  0.0000000E+00
14  0.0000000E+00
15  0.0000000E+00
16  0.0000000E+00
17  0.0000000E+00
18  0.0000000E+00
19  1.0000000E+00
20  0.0000000E+00
21  0.0000000E+00
22  1.0000000E+00
23  0.0000000E+00
 ..................
```



vgrid.in (1): SZ

```
2  ivcor
28 18   100. !h_s
Z levels
1       -5000.  <-- Bottom z
........
18      -100.00 <-- Last z = -h_s
S levels
30. 0.9 3. h_c, θ_b, θ_f
18    -1        <-- 1st σ
19    -0.9
........
28    0.  <-- Surface σ
```



vgrid.in (2): LSC^2

- Needs pre-proc script to generate (e.g. gen_vqs*.f90 in Utility)
- *Always visualize a few transects using plot_vqs.m*

```
 1 ivcor
42 nvrt
        1       37    -1.000000    -0.703456    -0.665960    -0.369236    -0.167103     0.000000
        2       37    -1.000000    -0.703456    -0.665960    -0.369236    -0.167103     0.000000
      ..............................

      233       37    -1.000000    -0.703456    -0.665960    -0.369236    -0.167103     0.000000
      234       37    -1.000000    -0.703456    -0.665960    -0.369236    -0.167103     0.000000
      235       37    -1.000000    -0.703456    -0.665960    -0.369236    -0.167103     0.000000
      236       34    -1.000000    -0.749747    -0.562022    -0.419538    -0.309461    -0.222147    -0.136640    -0.065220     0.000000
      237       34    -1.000000    -0.754543    -0.570416    -0.430663    -0.322695    -0.237054    -0.144453    -0.068642     0.000000
.....................................
```

Node # | **Bottom index** | **Bottom σ** | **Surface σ**

bctides.in: a simple example

04/30/2002 00:00:00 PST
0 40. ntip
0 nbfr
0 nope*

*Note that there is no open boundary here. This is convenient for ipre=1

bctides.in: more examples

A simpler example

```
12/01/2013 00:00:00 GMT
0 50. ntip
0  nbfr
2 nope
1190 4 4 4 4 3 Pacific
5.e-1
5.e-1
1.
139 4 4 4 4 3 South China Sea
5.e-1
5.e-1
1.
```

Relaxation constants for each tracer module

b.c. flags for **enabled** tracer modules (SED in this case)

A more complex example

```
04/15/2004 00:00:00 PST
0 40. ntip
2  nbfr
Z0
 0. 1. 0.
O1
 6.759775e-05  1.15343 118.92151
4 nope
3 3 0 4 0  Georgia
 Z0
  0.1299680       0.000000E+00
  0.1299680       0.000000E+00
  0.1299680       0.000000E+00
O1
  0.361010246  281.862898
  0.36065857  281.918305
  0.360767938  281.968364
1.e-3 T relaxation
88 3 0 0 0 ocean
 Z0
   8.892417E-02   0.000000E+00
   8.631096E-02   0.000000E+00
.....................
O1
.....................
```

Extra b.c. flags for tracer models if invoked

Phases don't matter for Z0

- Use bi-section method if you really cannot find the cause for crash….
- B.C. flags drives additional b.c. inputs .th*
- Order of tracer modules: T, S, GEN, AGE, SED3D, EcoSim, ICM, CoSINE, Feco, TIMOR, FABM

param.in

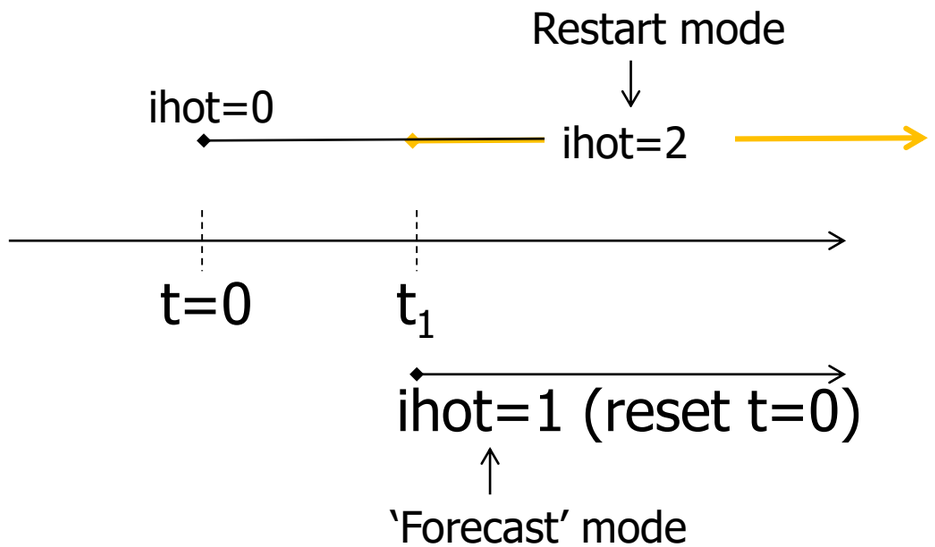
- Free format rules:
  - Lines beginning with "" are comments; blank lines are ignored;
  - one line for each parameter in the format: keywords= value;
    keywords are case sensitive; spaces allowed between keywords and "=" and value; comments starting with "" allowed after value;
  - value is an integer, double, or 2-char string; for double, any of the format is acceptable: 40 40. 4.e1; use of decimal point in integers is OK but discouraged;
  - Order of parameters not important

```
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Model configuration parameters
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
!---------------------------------------------------------------------
! Pre-processing option. Useful for checking grid violations and get sample z coordinates (sample_z.out)
!---------------------------------------------------------------------
  ipre = 0 !Pre-processor flag (1: on; 0: off)


!---------------------------------------------------------------------
! Bed deformation option (1: on; 0: off). If imm=1, bdef.gr3 is needed.
!---------------------------------------------------------------------
  imm = 0
! ibdef = 10 !needed if imm=1; # of steps used in deformation
```

param.in


```
!------------------------------------------------------------------------
! Coordinate option: 1: Cartesian; 2: lon/lat (hgrid.gr3=hgrid.ll in this case,
! and orientation of triangles is outward of earth)
!------------------------------------------------------------------------
 ics = 1 !Coordinate option
 cpp_lon = -124  !CPP projection centers: lon; not used if ics=2
 cpp_lat = 46.25 !CPP projection centers: lat

!------------------------------------------------------------------------
! Baroclinic/barotropic option. If ibcc=0 (baroclinic model), itransport is not used.
!------------------------------------------------------------------------
 ibcc = 0 !Baroclinic option
 itransport = 1
 nrampbc = 1 !ramp-up flag for baroclinic force
 drampbc = 1. !not used if nrampbc=0

!------------------------------------------------------------------------
! Hotstart option. 0: cold start; 1: hotstart with time reset to 0; 2:
! continue from the step in hotstart.in
!------------------------------------------------------------------------
 ihot = 1

!------------------------------------------------------------------------
! Hydraulic model option. If ihydraulics/=0, hydraulics.in
! is required. This option cannot be used with non-hydrostatic model.
!------------------------------------------------------------------------
 ihydraulics = 0

!------------------------------------------------------------------------
! Point sources/sinks option (0: no; 1: on). If =1, needs source_sink.in,
! vsource,th, vsink.th, and msource.th
!------------------------------------------------------------------------
 if_source = 0
```

You can use lon/lat even for high-resolution inundation study!



* Most parts of the code do not use the calendar time
  * Exceptions: heat exchange, WWM etc
* The code basically starts from t=0 and marches thru time

Initial condition flags

```
!------------------------------------------------------------------------
! Define # of tracers in each module (if enabled)
!------------------------------------------------------------------------
  ntracer_gen = 2 !user defined module
  ntracer_age = 4 !age calculation. Must be =2*N where N is # of age tracers
  sed_class = 5 !SED3D
  eco_class = 27 !EcoSim: must be between [25,60]
!------------------------------------------------------------------------
! Initial condition for T,S. This value only matters for ihot=0 (cold start).
! If ic_*=1, the initial T,S field is read in from temp.ic and salt.ic (horizontally varying).
! If ic_*=2, the initial T,S field is read in from ts.ic (vertical varying).
! If ihot=0 && ic_*=2 || ibcc_mean=1, ts.ic is used for removing mean density profile.
!------------------------------------------------------------------------
   ic_TEM = 1
   ic_SAL = 1 !must be same as ic_TEM

! initial conditions for other tracers.
! 1: needs inputs [MOD]_hvar_[1,2,...].ic ('1...' is tracer id); format of each file is similar to salt.ic;
!    i.e. horizontally varying i.c. is used for each tracer.
! 2: needs [MOD]_vvar_[1,2,...].ic. Format of each file (for each tracer in tis MOD) is similar to ts.ic
!    (i.e. level #, z-coord., tracer value). Verically varying i.c. is used for each tracer.
! 0: model sets own i.c. (EcoSim; TIMOR)
   ic_GEN = 1 !user defined module
   ic_AGE = 1 !Age
   ic_SED = 1 !SED3D
   ic_ECO = 1 !EcoSim
   ic_ICM = 1 !ICM
   ic_COS = 1 !CoSINE
   ic_FIB = 1 !FIB
```

Control of dissipation in momentum


```
!------------------------------------------------------------------------
! Methods for computing velocity at nodes.
! If indvel=0, conformal linear shape function is used; if indvel=1, averaging method is used.
! For indvel=0, a stabilization method is needed (see below).
!------------------------------------------------------------------------
 indvel = 0

!------------------------------------------------------------------------
! 2 stabilization methods, mostly for indvel=0.
! (1) Horizontal viscosity option. ihorcon=0: no viscosity is used; =1: Lapacian;
! =2: bi-harmonic. If ihorcon=1, horizontal viscosity _coefficient_ (<=1/8, related
! to diffusion number) is given in hvis_coef0, and the diffusion #
! is problem dependent; [0.001-1/8] seems to work well.
! If ihorcon=2, diffusion number is given by hvis_coef0 (<=0.025).
! If indvel=1, no horizontal viscosity is needed.
! (2) Shapiro filter (see below)
!
! For non-eddying regime applications (nearshore, estuary, river), two easiest options are:
! (1) indvel=1, ishapiro=ihorcon=0, and any choices of inter_mom;
! (2) indvel=0, ishapiro=1 (shapiro=0.5), ihorcon=inter_mom=0.
! For applications that include eddying regime, refer to the manual.
!------------------------------------------------------------------------
 ihorcon = 0
 hvis_coef0 = 0.025 !const. diffusion # if ihorcon/=0; <=0.025 for ihorcon=2, <=0.125 for ihorcon=1
!  cdh = 0.01 !needed only if ihorcon/=0; land friction coefficient - not active yet

!------------------------------------------------------------------------
! 2nd stabilization method via Shapiro filter. This should normally be used
! if indvel=ihorcon=0. To transition between eddying/non-eddying regimes, use
! indvel=0, ihorcon/=0, and ishapiro=-1 (shapiro.gr3).
!------------------------------------------------------------------------
 ishapiro = 1 !on/off flag
 shapiro = 0.5 !Shapiro filter strength, needed only if ishapiro=1; max is 0.5

!------------------------------------------------------------------------
! Horizontal diffusivity option. if ihdif=1, horizontal diffusivity is given in hdif.gr3
!------------------------------------------------------------------------
 ihdif = 0
```

- Eddying or regime or cross scale: indvel=0, ihorcon=2, hvis_coef0 = 0.025 ishapiro=0 or
  - We are working on combined bi-harmonic and Laplacian viscosity (to replace ishapiro=-1)
- indvel=1, ishapiro=ihorcon=0 occasionally useful for b-tropic applications

param.in


```
!------------------------------------------------------------------
! Bottom friction.
!       bfric=0: drag coefficients specified in drag.gr3; bfric=-1: Manning's
!       formulation (even for 3D prisms).
!       bfric=1: bottom roughness (in meters) specified in rough.gr3 (and in this case, negative
!       or 0 depths in rough.gr3 indicate time-independent Cd, not roughness!).
!       Cd is calculated using the log law, when dzb>=dzb_min; when dzb<dzb_min,
!       Cd=Cdmax*exp[dzb_decay*(1-dzb/dzb_min)], where Cdmax=Cd(dzb=dzb_min),
!       and dzb_decay (<=0) is a decay const specified below.
!       If iwbl=1, bfric must =1.
!------------------------------------------------------------------
 bfric = 0 !nchi in code
 !dzb_min = 0.5 !needed if bfric==1; min. bottom boundary layer thickness [m].
 !dzb_decay = 0. !needed if bfric=1; a decay const. [-]
 hmin_man = 1. !needed if bfric=-1: min. depth in Manning's formulation [m]

!------------------------------------------------------------------
! Coriolis. If ncor=-1, specify "latitude" (in degrees); if ncor=0,
! specify Coriolis parameter in "coriolis"; if ncor=1, model uses
! lat/lon in hgrid.ll for beta-plane approximation if ics=1, and in this case,
! the latitude specified in CPP projection ('cpp_lat') is used. If ncor=1 and ics=2,
! Coriolis is calculated from local latitude, and 'cpp_lat' is not used.
!------------------------------------------------------------------
 ncor = 1 !must be 1 if ics=2
 !latitude = 46 !if ncor=-1
 !coriolis = 1.e-4 !if ncor=0
```

If bfric=1

We always use dzb_decay=0, so Cd<=Cdmax

param.in


```
!------------------------------------------------------------------
! Wetting and drying. If ihhat=1, \hat{H} is made non-negative to enhance
! robustness near wetting and drying; if ihhat=0, no retriction is imposed for
! this quantity.
! inunfl=0 is used for normal cases and inunfl=1 (not available yet) is used for more accurate wetting
! and drying if grid resolution is sufficiently fine.
!------------------------------------------------------------------
 ihhat = 1 !not used for 2D model
 inunfl = 0
 h0 = 0.01 !min. water depth for wetting/drying

!------------------------------------------------------------------
! Implicitness factor (0.5<thetai<=1).
!------------------------------------------------------------------
 thetai = 0.6

!------------------------------------------------------------------
! Run time and ramp option
!------------------------------------------------------------------
 rnday = 42 !total run time in days
 nramp = 1 !ramp-up option (1: on; 0: off)
 dramp = 2. !needed if nramp=1; ramp-up period in days
 dt = 150. !Time step in sec

!------------------------------------------------------------------
! Solver option. JCG is used presently.
!------------------------------------------------------------------
 slvr_output_spool = 50 !output spool for solver info (fort.33)
 mxitn = 1000 !max. iteration allowed
 tolerance = 1.e-12 !error tolerance
```

θ ⟶

param.in


```
!--------------------------------------------------------------------
! Advection (ELM) option. If nadv=1, backtracking is done using Euler method;
! nadv=2, using 2nd order Runge-Kutta; if nadv=0, advection in momentum
! is turned off/on in adv.gr3 (the depths=0,1, or 2 also control methods
! in backtracking as above). dtb_max/min are the max/min steps allowed -
! actual step is calculated adaptively based on local flow gradient.
!--------------------------------------------------------------------
  nadv = 1
  dtb_max = 50. !in sec
  dtb_min = 15.


!--------------------------------------------------------------------
! If inter_mom=0, linear interpolation is used for velocity at foot of char. line.
! If inter_mom=1 or -1, Kriging is used, and the choice of covariance function is
! specified in 'kr_co'. If inter_mom=1, Kriging is applied to whole domain;
! if inter_mom=-1, the regions where Kriging is used is specified in krvel.gr3
! (depth=0: no kriging; depth=1: with kriging).
! For velocity, additional controls are available in 'blend_internal' and 'blend_bnd',
! two parameters specifying how continuous and discontinuous velocities are blended
! for internal and boundary sides. If indvel=1, code resets blend_internal=blend_bnd=0.
!--------------------------------------------------------------------
  inter_mom = 0
  kr_co = 1 !not used if inter_mom=0 (-h, h^2log(h), h^3, -h^5)
```

Choice of inter_mom depends on *indvel.* If indvel=0, inter_mom=1, then kr_co should be 1 (otherwise dispersion is too strong)

Transport solver

- Use itr_met=3 or 4 for most b-clinic applications

```
!------------------------------------------------------------------
! Transport method.
! If itr_met=1, upwind method is used. If itr_met>=2, TVD or WENO method is used
! on an element/prism if the total depth (at all nodes of the elem.)>=h_tvd and the flag in
! tvd.prop = 1 for the elem. (tvd.prop is required in this case);
! otherwise upwind is used for efficiency.
! itr_met=3 (horizontal TVD) or 4 (horizontal WENO): implicit TVD in the vertical dimension.
! Also if itr_met==3 and h_tvd>=1.e5, some parts of the code are bypassed for efficiency
! Controls for WENO are not yet in place
!------------------------------------------------------------------
 itr_met = 4
 h_tvd = 5. !used only if itr_met>=2; cut-off depth (m)
 !If itr_met=3 or 4, need the following 2 tolerances of convergence. The convergence
 !is achieved when sqrt[\sum_i(T_i^s+1-T_i^s)^2]<=eps1_tvd_imp*sqrt[\sum_i(T_i^s)^2]+eps2_tvd_imp
 eps1_tvd_imp = 1.e-4 !suggested value is 1.e-4, but for large suspended load, need to use a smaller value (e.g. 1.e-9)
 eps2_tvd_imp = 1.e-14

 !if itr_met = 4, the following parameters are needed
 !if itr_met=4 and ipre=1, diagnostic outputs are generated for weno accuracy and stencil quality,
 !  see subroutine weno_diag in src/Hydro/misc_subs.F90 for details
 ip_weno = 2   !order of accuracy: 0- upwind; 1- linear polynomial, 2nd order; 2- quadratic polynomial, 3rd order
 courant_weno=0.5 !Courant number for weno transport
 epsilon1 = 1.e-8   !coefficient for 2nd order weno smoother
 epsilon2 = 1.e-10  !1st coefficient for 3rd order weno smoother
 epsilon3 = 1.e-25  !2nd coefficient for 3rd order weno smoother (not used)
 !Elad filter has not been implemented yet; preliminary tests showed it might not be necessary
 ielad_weno = 0      !ielad, if ielad=1, use ELAD method to suppress dispersion
 small_elad = 1.e-4  !small
```

Atmospheric forcing


```
!--------------------------------------------------------------------
! Atmos. option. If nws=0, no atmos. forcing is applied. If nws=1, atmos.
! variables are read in from wind.th. If nws=2, atmos. variables are
! read in from sflux_ files.
! If nws=4, ascii format is used for wind and atmos. pressure at each node (see source code).
! If nws>0, 'iwindoff' can be used to scale wind speed (with windfactor.gr3).
!
! Stress calculation:
! If nws=1 or >=4, or nws=2 and ihconsv=0, or nws=2 and iwind_form=-1,
! the stress is calculated from Pond & Pichard formulation
! If nws=2, ihconsv=1 and iwind_form=0, the stress is calculated from heat exchange
! routine;
! If WWM is enabled and icou_elfe_wwm>0 and iwind_form=-2, stress is calculated by WWM;
! otherwise the formulations above are used.
!--------------------------------------------------------------------
  nws = 2
  wtiminc = 150. !time step for atmos. forcing
  nrampwind = 1 !ramp-up option for atmos. forcing
  drampwind = 2. !needed if nrampwind/=0; ramp-up period in days
  iwindoff = 0 !needed only if nws/=0; '1': needs windfactor.gr3
  iwind_form = -1

!--------------------------------------------------------------------
! Heat and salt exchange. isconsv=1 needs ihconsv=1; ihconsv=1 needs nws=2.
! If isconsv=1, need to compile with precip/evap module turned on.
!--------------------------------------------------------------------
  ihconsv = 1 !heat exchange option
  isconsv = 0 !evaporation/precipitation model
```

nws=2 is recommended. There are m-lab scripts to help you

Turbulence closure


```
!--------------------------------------------------------------------
! Turbulence closure.
!--------------------------------------------------------------------
  itur = 3
!  dfv0 = 1.e-6 !needed if itur=0
!  dfh0 = 1.e-6 !needed if itur=0
  turb_met = KE !needed if itur=3,5. Use KE if itur=5
  turb_stab = KC !needed if itur=3 or 5. Use 'GA' if turb_met='MY'; otherwise use 'KC'.
  xlsc0 = 0.1 !needed if itur=3 or 5. Scale for surface & bottom mixing length (>0)
```

3 or 4: needs diffmax.gr3,diffmin.gr3

Nudging options


```
!------------------------------------------------------------------
! Sponge layer for elevation and vel.
! If inu_elev=0, no relaxation is applied to elev.
! If inu_elev=1, relax. constants are specified in elev_nudge.gr3
! and applied to eta=0 (thus a depth=0 means no relaxation).
! Similarly for inu_uv (with input uv_nudge.gr3)
!------------------------------------------------------------------
  inu_elev = 0
  inu_uv = 0

!------------------------------------------------------------------
! Nudging options for tracers. If inu_[MOD]=0, no nudging is used. If inu_[MOD]=1,
! nudge to initial condition according to relaxation constants specified
! in [MOD]_nudge.gr3. If inu_[MOD]=2, nudge to values in [MOD]_nu,in
! (with step 'step_nu_tr') according to [MOD]_nudge.gr3.
! The final relaxation = horizontal relax (specified in [MOD]_nudge.gr3) times dt.
! [MOD] are tracer model names.
!------------------------------------------------------------------
  inu_TEM = 0
  inu_SAL = 0
  inu_GEN = 0 !user defined
  inu_AGE = 0 !Age
  inu_SED = 0 !SED3D
  inu_ECO = 0 !EcoSim
  inu_ICM = 0 !ICM
  inu_COS = 0 !CoSINE
  inu_FIB = 0 !FIB

  step_nu_tr = 86400. !shared time step [sec] in all [MOD]_nu.in (for inu_[MOD]=2)
```

For each tracer module; needs [MOD]_nu.nc if '2'

Backtracking


```
!---------------------------------------------------------------------
! Dimensioning parameters for inter-subdomain btrack.
! If error occurs like 'bktrk_subs: overflow' or 'MAIN: nbtrk > mxnbt'
! gradually increasing these will solve the problem
!---------------------------------------------------------------------
  s1_mxnbt = 0.5
  s2_mxnbt = 3.5
```

No effects on accuracy; only affect memory consumption

param.in


```
!------------------------------------------------------------------
!Option for hotstart outputs (use combine_hotstart7.f90 to combine)
!------------------------------------------------------------------
  hotout = 1 !1: output *_hotstart every 'hotout_write' steps
  hotout_write = 4032 !divisible by ihfskip

!------------------------------------------------------------------
!Global output options (node centered)
!------------------------------------------------------------------
  nspool = 6 !output step spool
  ihfskip = 576 !stack spool; every ihfskip steps will be put into 1_*, 2_*, etc...
  elev.61 = 1 !0: off; 1: on
  pres.61 = 1
  airt.61 = 1
  shum.61 = 1
  srad.61 = 1
  flsu.61 = 1
  fllu.61 = 1
  radu.61 = 1
  radd.61 = 1
  flux.61 = 1
  evap.61 = 0
  prcp.61 = 0
  wind.62 = 1
  wist.62 = 0
  dahv.62 = 0
  vert.63 = 1
  temp.63 = 1
  salt.63 = 1
  conc.63 = 0
  tdff.63 = 1
  vdff.63 = 0
  kine.63 = 0
```

```
!------------------------------------------------------------------
!Outputs for (age)
!------------------------------------------------------------------
  AGE_1.63 = 0 !indices from "1" to "ntr/2"; [days]
  AGE_2.63 = 0

!------------------------------------------------------------------
!Specific outputs in SED3D (USE_SED must be on in Makefile;
!otherwise these are not needed)
!------------------------------------------------------------------
  SED_1.63 = 0 !conc. of 1st class (one output need by each class) [g/L]
  SED_2.63 = 0
  SED_bfrac_1.61 = 0  !Bed fraction 1st tracer (one output need by each class) [-]
  SED_bfrac_2.61 = 0
  SED_qbdl_1.62 = 0  !Bedload transport rate (kg.m-1.s-1) for 1st tracer (one output need by tracer)
  SED_qbdl_2.62 = 0
  ……
```

```
!------------------------------------------------------------------
!Non-node centered outputs section. Some of these need corresponding cpp flags
!to be on in order to be active
!------------------------------------------------------------------
  hvel.67 = 0 !horizontal vel. defined at side [m/s]
  vert.69 = 0 !vertical vel. at centroids [m/s]
  temp.70 = 0 !T at prism centers [C]
  salt.70 = 0 S !at prism centers [PSU]
  z0st.66 = 0  !Sediment transport roughness length (m) {module: SED}
  ……
```

param.in

```
!------------------------------------------------------------------------
 !Station output option. If iout_sta/=0, need output skip (nspool_sta) and
 !a station.in. If ics=2, the cordinates in station.in must be in lon., lat,
 !and z (measured from MSL; not used for 2D variables).
!------------------------------------------------------------------------
  iout_sta = 0
  nspool_sta = 10 !needed if iout_sta/=0

!------------------------------------------------------------------------
! Flag for harmonic analysis for elevation. If used , need to turn on cpp flags
! in Makefile first. Otherwise set it to 0.
!------------------------------------------------------------------------
  iharind = 0


!------------------------------------------------------------------------
! Conservation check option. If consv_check=1, some fluxes are computed
! in regions specified in fluxflag.prop (regional number from -1 to an arbitrary !integer).
!------------------------------------------------------------------------
  consv_check = 0
```

Most important parameters are…

- Momentum dissipation: indvel, ihorcon, ishapiro, inter_mom
  - Appropriate combination of these can be used for different applications (dispersion vs diffusion)
  - May start with: indvel= ihorcon=inter_mom=0, ishapiro=1 ('MB-LI' scheme)
  - If dissipation is still too high, use indvel= inter_mom=ishapiro=0, ihorcon=2, hvis_coef0=0.025 (or smaller) – the 'eddying' option. If oscillation is found in non-eddying part, use ishapiro=-1 and set the filter strength in shapiro.gr3 (eventually we will replace this with a new option of using both Laplacian and bi-harmonic viscosities)
  - indvel=1, ihorcon=ishapiro=inter_mom=0 is the most diffusive option ('MA-LI' scheme), but can be useful for 2D runs etc
- dt: appropriate time step
  - Another way to control dispersion/diffusion
  - Reducing dt will increase diffusion and decrease dispersion
- nws: use '2' as much as possible (and there are scripts for generating sflux*.nc)
- itr_met: use 3 or 4
  - Use 'h_tvd' and tvd.prop to control efficiency
  - 'Upwind' zone: upstream river where there is no stratification

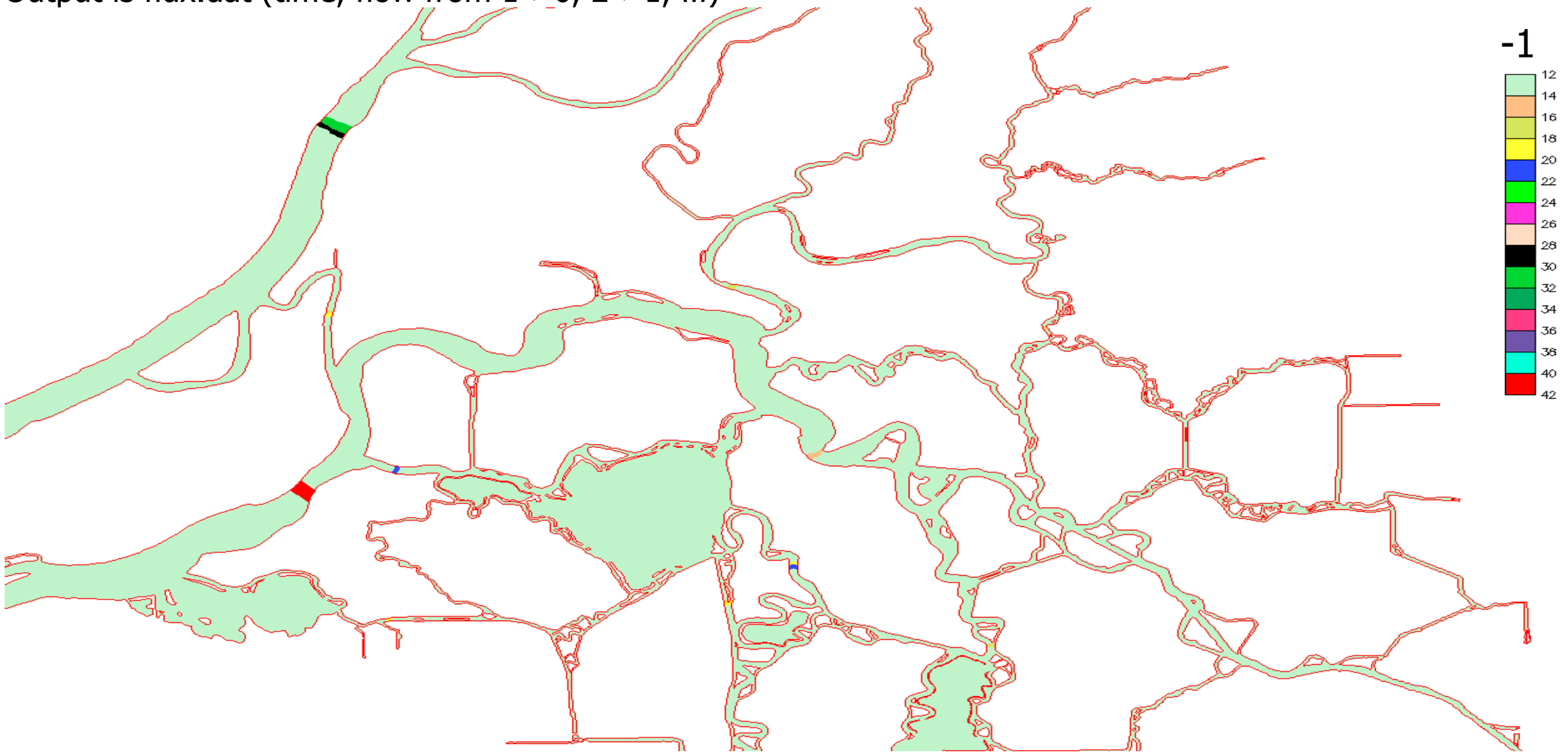
Station.in


```
1 1 1 1 1 1 1 1 1 !on (1)|off(0) flags for elev, air pressure, windx, windy, T, S, u, v, w
4        !# of stations
1 6.5833 54.0000 0      !Format: station #,x,y,z; z is z-coord. 2 7.1583 55.195 0
2 7.1583 55.195 -1.       !Fino3
3 6.35  54.1667 -5.       !Ems
4 8.4514 54.7942 0      !PHoern
```

- If ics=2, x,y are degrees in lon/lat.
- z is z-coordinates, not distance from surface! So z<0 is below the datum
- If you want to extract series at multiple depths, use multiple stations
- For 3D variables, code will extrapolate above surface/below bottom if necessary

Fluxflag.prop

- Compute various fluxes (from ‘high’ to ‘low’), if the difference of flags=1 and neither are -1
- Otherwise ignored
- Output is flux.dat (time, flow from 1->0, 2->1, …)



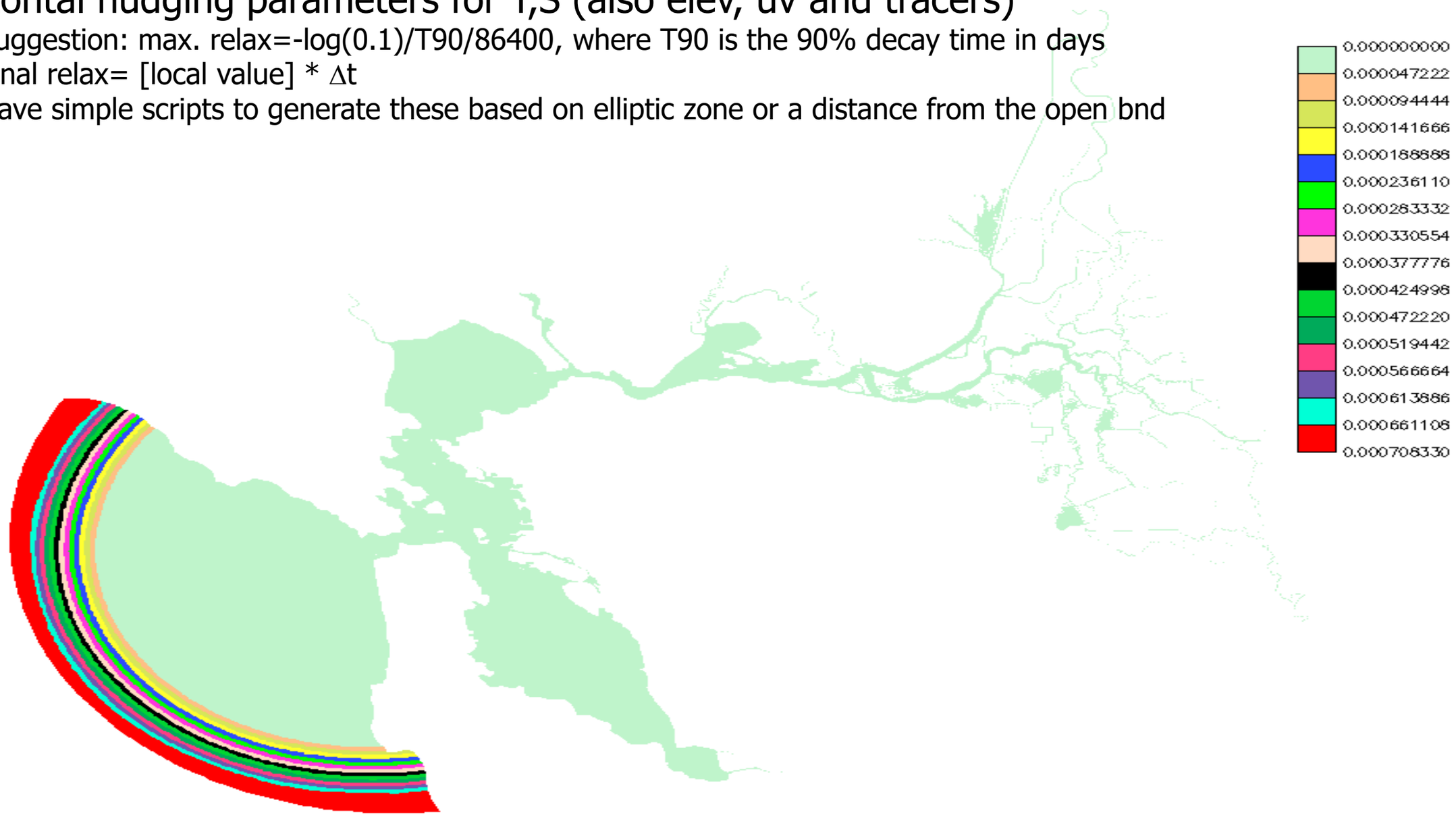
Hotstart and nuding inputs


```
netcdf hotstart {
dimensions:
        node = 20641 ;
        elem = 38960 ;
        side = 59615 ;
        nVert = 54 ;
        ntracers = 2 ;
        one = 1 ;
        three = 3 ;
variables:
        double time(one) ;
        int iths(one) ;
        int ifile(one) ;
        int idry_e(elem) ;
        int idry_s(side) ;
        int idry(node) ;
        double eta2(node) ;
        double we(elem, nVert) ;
        double tr_el(elem, nVert, ntracers) ;
        double su2(side, nVert) ;
        double sv2(side, nVert) ;
        double tr_nd(node, nVert, ntracers) ;
        double tr_nd0(node, nVert, ntracers) ;
        double q2(node, nVert) ;
        double xl(node, nVert) ;
        double dfv(node, nVert) ;
        double dfh(node, nVert) ;
        double dfq1(node, nVert) ;
        double dfq2(node, nVert) ;
data:

 time = 0 ;

 iths = 0 ;

 ifile = 1 ;

 idry_e = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
netcdf SAL_nu {
dimensions:
        node = 20641 ;
        nVert = 54 ;
        ntracers = 1 ;
        time = UNLIMITED ; // (29 currently)
variables:
        double time(time) ;
                time:long_name = "simulation time in days" ;
        float tracer_concentration(time, node, nVert, ntracers) ;
data:

 time = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
    20, 21, 22, 23, 24, 25, 26, 27, 28 ;

 tracer_concentration =
  34.63076,
  34.62345,
  34.57667,
  34.50737,
  34.4041,
  34.30476,
  34.17599,
  34.11315,
  34.06581,
  34.03587,
  33.99743,
```

[TEM,SAL, TR]_nudge.gr3

- Horizontal nudging parameters for T,S (also elev, uv and tracers)
  - Suggestion: max. relax=-log(0.1)/T90/86400, where T90 is the 90% decay time in days
  - Final relax= [local value] * Δt
  - Have simple scripts to generate these based on elliptic zone or a distance from the open bnd



*.th: ASCII

flux.th (**negative for inflow**), TEM_1.th, SAL_1.th, elev.th, wind.th

- Corresponds to b.c. flag=1
- Easy to plot with xmgr5

Time (sec) value 1 (1st bnd) value 2 (2nd bnd) ..... value N (Nth bnd)

```
0.      -0.084950529 0. -0.283168435 -0.226534754 -3.19980335 -16.367136
86400 -0.084950529 0. -0.283168435 -0.226534754 -3.19980335 -16.367136
172800 -0.084950529 0. -0.283168435 -0.226534754 -0.906139016 -22.2853565
259200 -0.084950529 0. -0.25485158 -0.226534754 -0.764554799 -16.367136
345600 -0.084950529 0. -0.25485158 -0.226534754 -0.764554799 -16.367136
432000 -0.084950529 0. -0.25485158 -0.226534754 -0.877822161 -16.367136
518400 -0.084950529 0. -0.25485158 -0.226534754 -3.17148638 -16.367136
604800 -0.084950529 0. -0.25485158 -0.226534754 -0.906139016 -16.367136
.....
```

- Time steps ≥ main time step in param.in; record must start from 0
- Time step in wind.th specified in param.in (*wtiminc*)

*[23]D.th.nc

elev2D.th.nc, TEM_3D.th.nc, SAL_3D.th.nc, [tracer]_3D.th.nc, uv3D.th.nc (not discharge)

- Corresponds to b.c. flag=±4 or 5
- Make sure you check values after creation

```
netcdf uv3D.th {
dimensions:
        nOpenBndNodes = 1329 ;
        nLevels = 48 ;
        nComponents = 2 ;
        one = 1 ;
        time = UNLIMITED ; // (2881 currently)
variables:
        float time_step(one) ;
                time_step:long_name = "time step in seconds" ;
        double time(time) ;
                time:long_name = "simulation time in seconds" ;
        float time_series(time, nOpenBndNodes, nLevels, nComponents) ;

// global attributes:
                :_NCProperties = "version=1|netcdflibversion=4.4.1.1|hdf5libversion=1.8.18" ;
data:

 time_step = 3600 ;

 time = 0, 3600, 7200, 10800, 14400, 18000, 21600, 25200, 28800, 32400,
    36000, 39600, 43200, 46800, 50400, 54000, 57600, 61200, 64800, 68400,
    72000, 75600, 79200, 82800, 86400, 90000, 93600, 97200, 100800, 104400,
    108000, 111600, 115200, 118800, 122400, 126000, 129600, 133200, 136800,
    140400, 144000, 147600, 151200, 154800, 158400, 162000, 165600, 169200,
```

Time steps may be different from each other and ≥ dt in param.in; record must start from t=0

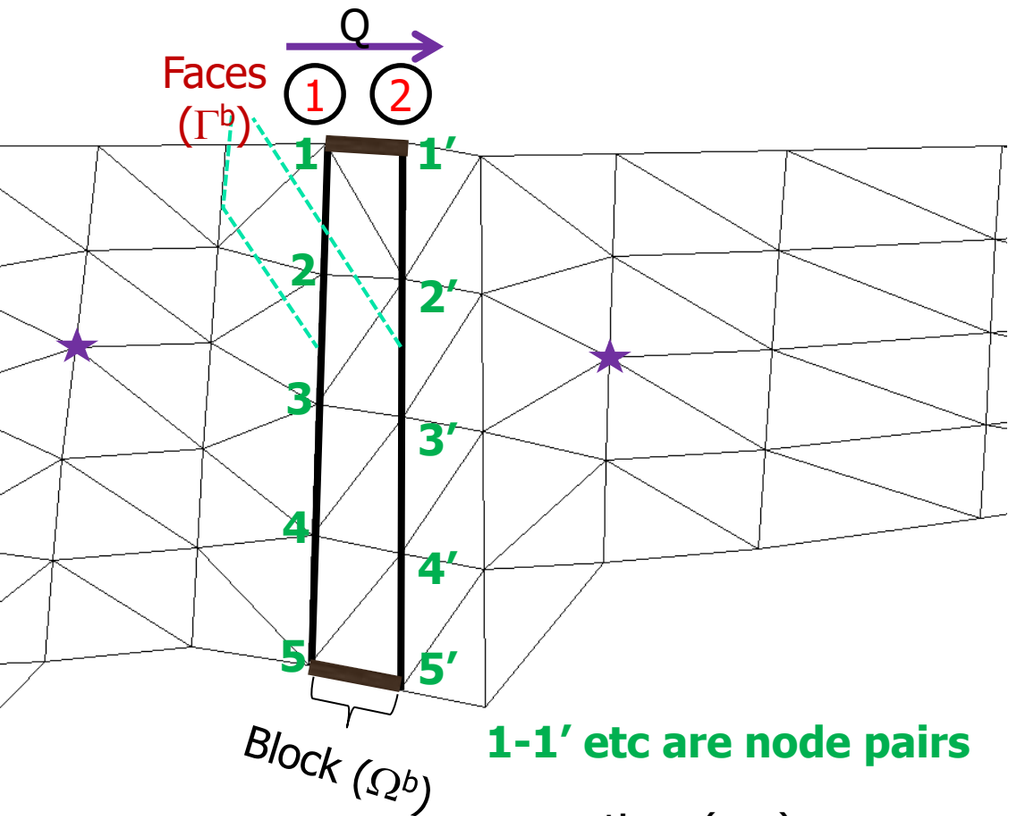
*.ic: initial condition for T,S, tracers

- ic_TEM(=ic_SAL)=1 in param.in: needs temp.ic and salt.ic (.gr3 format)
- ic_TEM=2 or ibcc_mean=1 in param.in: needs ts.ic (mean density removed in the code automatically)

Sample ts.ic


```
43 total # of vertical levels;
1 -2000. 4. 34. level #, z-coordinates, temperature, salinity
2 -1000. 5. 34.
3 -500.   6. 33.
4 -200.   7. 32.
5 -100.   8. 31.
.....
```

- Constant extrapolation is used if necessary below bottom or above surface
- For more generic 3D i.c., use hotstart.in
- For tracer modules, the files names are [tr]_hvar_[tn].ic (if ic_[tr]=1), or [tr]_vvar_[tn].ic (if ic_[tr]=2), where 'tr' is the tracer module name (e.g. 'SED'), and 'tn' is the tracer index inside each module (e.g., class 1,2,…); e.g. SED_hvar_1.ic. The format is similar to temp.ic or ts.ic above



1-1′ etc are node pairs

```
14 # of structures
0.1  Nudging factor
1 delta_cross_channel
3 48402 48977   # of node-pairs, 2 ref. nodes (global indices) for 2 faces
48403 48589 node pair
48591 48782
48783 48978
radial  struct type
2  n_duplicates
-3.420000 18.290000 10.000000  elevation, width, height or radius
1.000000 1.000000 1.000000  coef, op_downstream, op_upstream
1  time series enabled
2 ccfb_gate
3 163413 163248   # of node-pairs, 2 ref.nodes (global indices) for 2 faces
```

delta_cross_channel.th

time (sec) | op_up | elevation, width, height

```
-315420.0     1     2     1     1     -3.42   18.29   0
17748180.0    1     2     1     1     -3.42   18.29   10
28290600.0    1     2     1     1     -3.42   18.29   0
28297800.0    1     2     1     1     -3.42   18.29   10
```

install | nduplicates | op_down

Consult hydraulics manual for more details

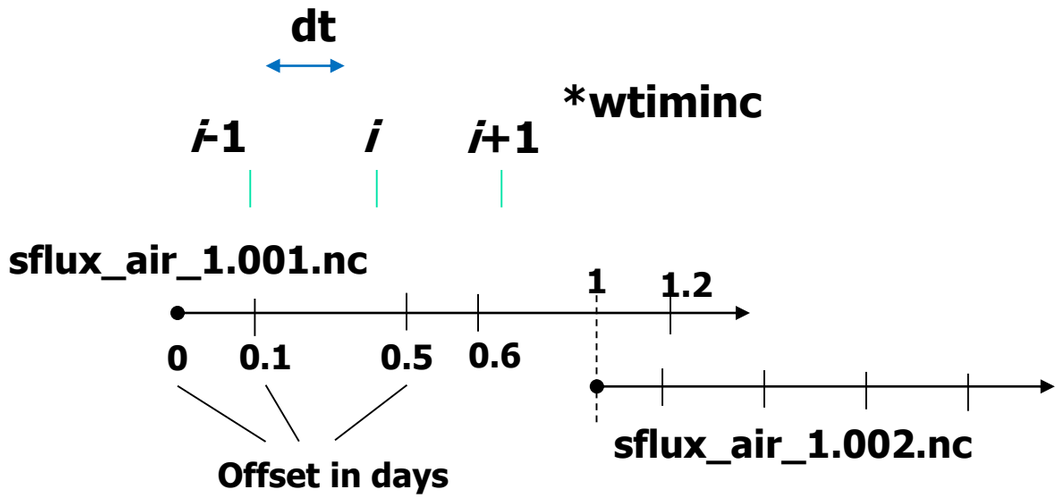
sflux/ (nws=2)

- Requires hgrid.ll for interpolation
- netcdf files (CF-1.0) reformatted from CFSR, NARR, NAM, WRF, MM5… (structured grid)
- Three types
  - sflux_air_[12]_*.nc: wind speed (u,v) (10m above MSL), air pressure (MSL), surface air T (2m above MSL), and *specific* humidity (2m above MSL)
  - sflux_rad_[12]_*.nc: *downward* long (infrared) and short (solar) wave radiation fluxes – used in heat exchange only (ihconsv=1)
  - sflux_prc_[12]_*.nc: surface precipitation rate (kg/m^2/s) – used in salt exchange only (isconsv=1)
- Sample files: NARR (we have 1979-present on Sciclone)
  - Download NARR files for your application period. Each NARR file covers ~ 1 day (e.g., narr_air.1981_01_28.nc is for Jan. 28, 1981).
  - In your run directory, mkdir sflux and inside it, create symbolic links to the NARR files. e.g., if you run starts from June 10, 2004 and ends June 20, 2004, then

```
ln –s   narr_air.2004_06_10.nc    sflux_air_1.001.nc (OR: earlier date to account for time zone)
ln –s   narr_air.2004_06_11.nc    sflux_air_1.002.nc
…
ln –s   narr_air.2004_06_21.nc    sflux_air_1.012.nc (extra day to account for time zone difference)
```

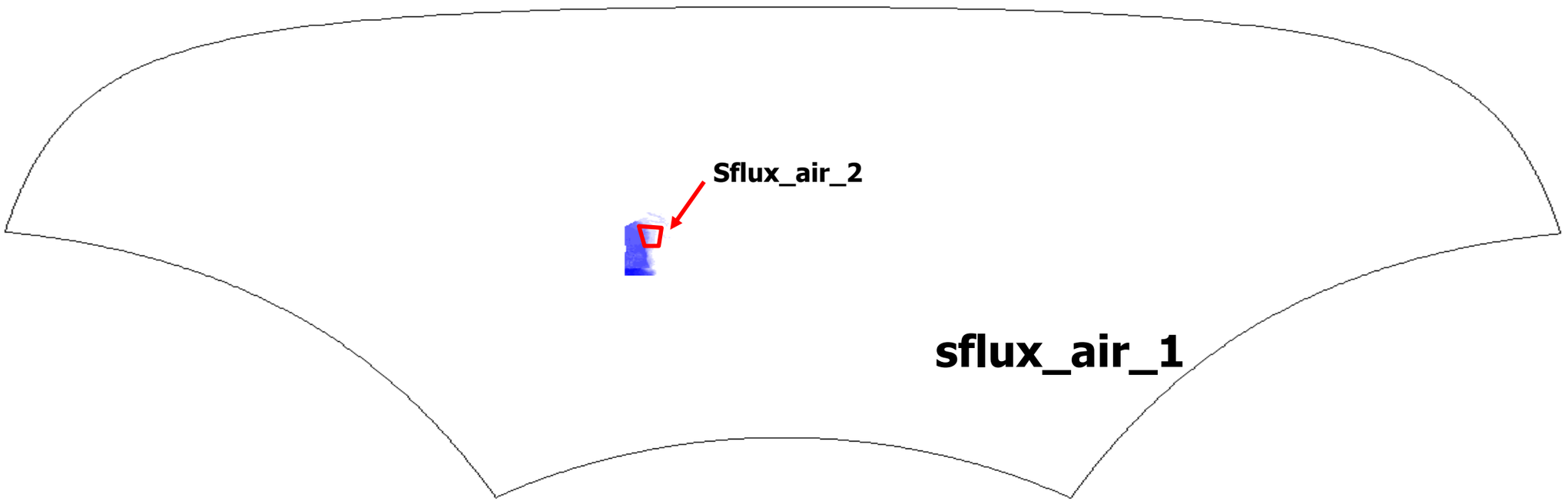
  - Similarly for sflux_rad_*.nc and sflux_prc_*.nc. The number "1" or "2" after "air_" denotes data set used
  - In sflux, copy the file *sflux_inputs.txt* and edit it to reflect the start time (time origin). The field "**utc_start**" is hours *behind* UTC (for application in eastern hemisphere, you may need to add an extra stack to cover *t*=0)
  - The time records in *.nc can be in irregular time steps. **The code will first dovetail all time records (giving preference to later deck e.g. hindcast)**
  - The time interval at which the main routine interpolates the atmos. info from *.nc is specified as *wtiminc* in param.in
  - Make sure the dovetailed time records from all .nc cover the entire simulation period

```
netcdf sflux_air_1.001 {
dimensions:
        nx_grid = 349 ;
        ny_grid = 277 ;
        time = UNLIMITED ; // (8 currently)
variables:
        float time(time) ;
                time:long_name = "Time" ;
                time:standard_name = "time" ;
                time:units = "days since 2001-01-01" ;
                time:base_date = 2001, 1, 1, 0 ;
        float lon(ny_grid, nx_grid) ;
                lon:long_name = "Longitude" ;
                lon:standard_name = "longitude" ;
                lon:units = "degrees_east" ;
        float lat(ny_grid, nx_grid) ;
                lat:long_name = "Latitude" ;
                lat:standard_name = "latitude" ;
                lat:units = "degrees_north" ;
        float uwind(time, ny_grid, nx_grid) ;
                uwind:long_name = "Surface Eastward Air Velocity (10m AGL)" ;
                uwind:standard_name = "eastward_wind" ;
                uwind:units = "m/s" ;
        float vwind(time, ny_grid, nx_grid) ;
                vwind:long_name = "Surface Northward Air Velocity (10m AGL)"
;
                vwind:standard_name = "northward_wind" ;
                vwind:units = "m/s" ;
        float prmsl(time, ny_grid, nx_grid) ;
                prmsl:long_name = "Pressure reduced to MSL" ;
                prmsl:standard_name = "air_pressure_at_sea_level" ;
                prmsl:units = "Pa" ;
        float stmp(time, ny_grid, nx_grid) ;
                stmp:long_name = "Surface Air Temperature (2m AGL)" ;
                stmp:standard_name = "air_temperature" ;
                stmp:units = "K" ;
```



- The code will dovetail time records from all .nc
- What happens if you choose a large wtiminc?
- Actually we almost always set wtiminc=dt for nws=2

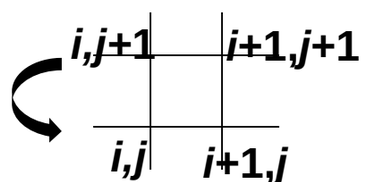
Details: interpolation



- In case of 2 sources/grids for a variable, use "1" as larger grid (i.e. encompassing hgrid.ll) and "2" as smaller grid
- Both grids must start from stack 1 and have same # of stacks for each variable. However, within each nc file # of time steps may vary
- In the overlapping area btw the 2 grids, the blending ratio is specified by *relative_weight* in netcdf_io of sflux_9c.F90 (default 1:99) to smooth out the transition
- The code will calculate interpolation weights associated with the 2 grids, and if some nodes in hgrid.ll fall outside grid "2" the interpolation will be done on grid "1" only (see combine_sflux_data, in particular, bad_node_2 based on area coordinates outside [0,1])

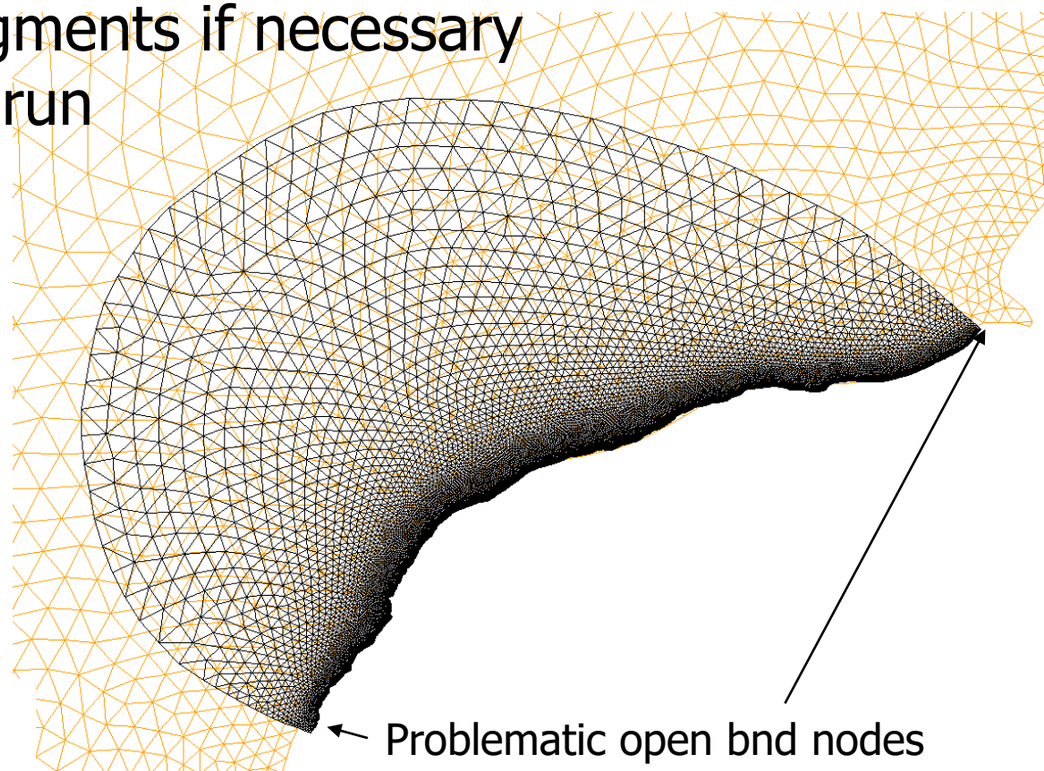
Creating your own sflux/

- Sample m-lab scripts: readnc*.m
- The grids used in atmospheric forcings are structured grids, but atmos grid "1" must encompass hgrid.ll for successful interpolation, which is done in SCHISM at run time (both in space and in time)
  - The collective time window from all .nc files MUST cover the simulation period
- Of all attributes in nc file, only 'base_date' is required (hour is not used)
- Grids for air, rad and prc can be different (but must be the same within each type and each source).
  - Additional requirements for the structured grid in .nc: [lon,lat](nx,ny) give x,y coord. (nx is # of pts in lon). Suppose a node in the grid is given by (i,j) (1<=i<=nx), then the quad (i,j), (i+1,j), (i+1,j+1,i,j+1) must be along **counter-clockwise** direction (check signs in .m scripts)
  - Beware the m-lab function meshgrid when you transform the 1D lon/lat arrays to 2D.
- Some constants are hard-wired in sflux_9c.F90
  - The maximum numbers of input files and time records are set in the module netcdf_io as *max_files* and *max_times*. If your application requires larger values simply increase them in the module
  - **_[12]_max_window_hours* are set in netcdf_io to define the max. time stamp (offset from start time in each file) within each nc file. Also, *max_file_times* (max. # of time records in each nc file) in routine get_times_etc() may be adjusted as well
- Be careful if you use a huge *wtiminc*



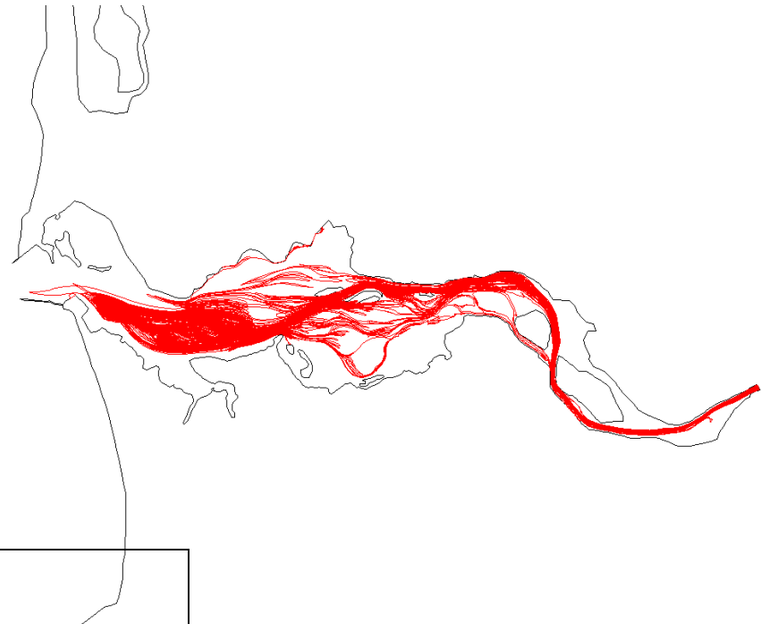
One-way nesting

- Useful as a way to generate uv3D.th etc (when no other sources are available)
- Use interpolate_variables_selfe7.f90 to generate *[23]D.th.nc for the small-domain run
  - Make sure the larger grid covers the smaller grid)
  - Since SCHISM does not allow drying at any open boundary nodes and at any time, make sure these nodes never become dry in the large-domain results – move them or redo open boundary segments if necessary
  - Use the new *[23]D.th.nc for the small-domain run



Particle tracking

- 3D tracking on unstructured grid
- Adapted from backtracking portion of SCHISM (currently only Euler)
  - Can do both forward and backward tracking
  - Oil spill option supports only forward tracking
- Inputs: hgrid.gr3, vgrid.in, particle.bp, schout_*.nc (need to contain elev, hvel, vertical_velocity; also wind and diffusivity for oil spill)
- Output: particle.pth (drogue format that can be viz'ed with xmvis6)



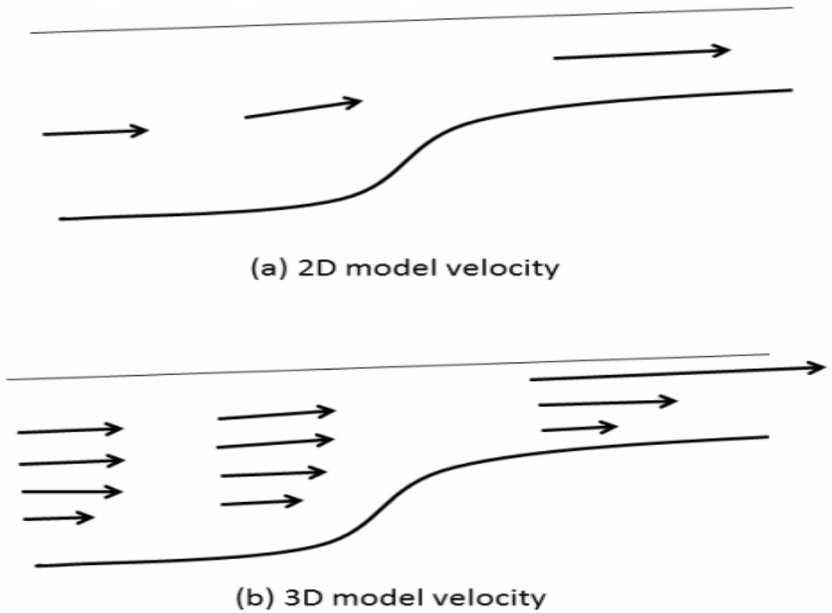
particle.bp

```
Input for ptrack
1 nscreen
0 mod_part (0: passive; 1: oil spill)
1 ibf (forward (=1) or backward (=-1) tracking)
1 istiff (1: from F.S.)
1 -124 46.25  ics slam0 sfea0 (ics=2 still uses CPP)
0.1 8. 90. 10 960 9 h0,rnday,dtm,nspool,ihfskip,ndeltp (same as param.in except for the last one, which is # of divisions)
16 nparticle
1 84600.        385000  510000 -5.    particle id, start time (s), starting x,y, and z relative to the instant f.s. (<=0)
2 84600.        300000  450000 -5.
3 84600.        200000  450000 -5.
4 84600.        120000  450000 -5.
5 84600.        380000  292500 -5.
6 84600.        360000  290000 -5.
7 84600.        320000  290000 -5.
.....
[Oil spill parameters]
```

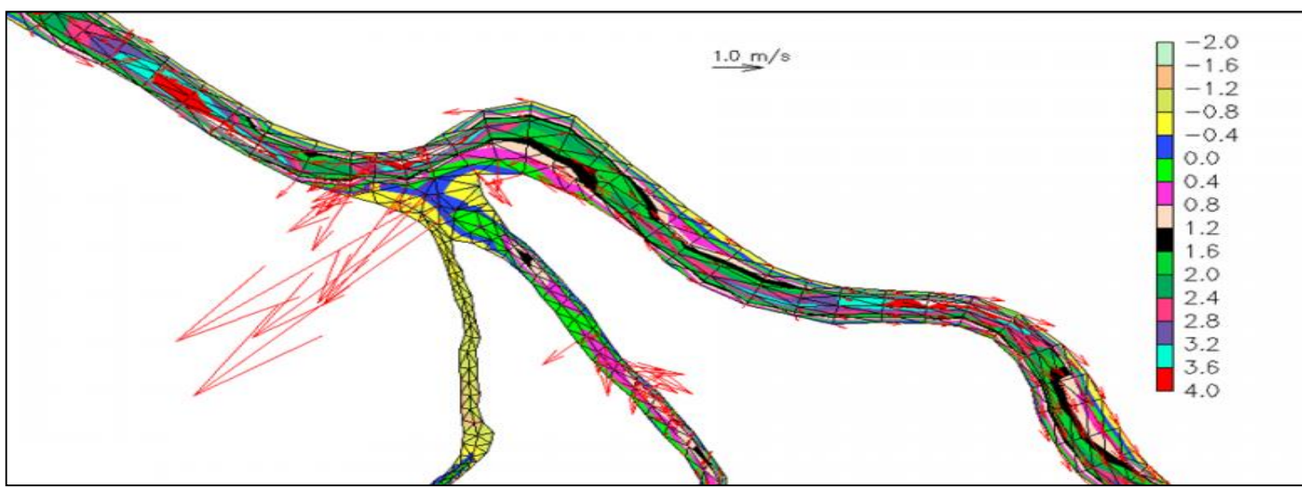
Advanced topics

Wet/dry in shallow region (3D model)

- In 3D, the roughness formulation leads to large Cd in shallow area
- Limiting Cd in the shallow helps: 3D model prefers smaller Cd in shallows
- Locally 2D configuration helps



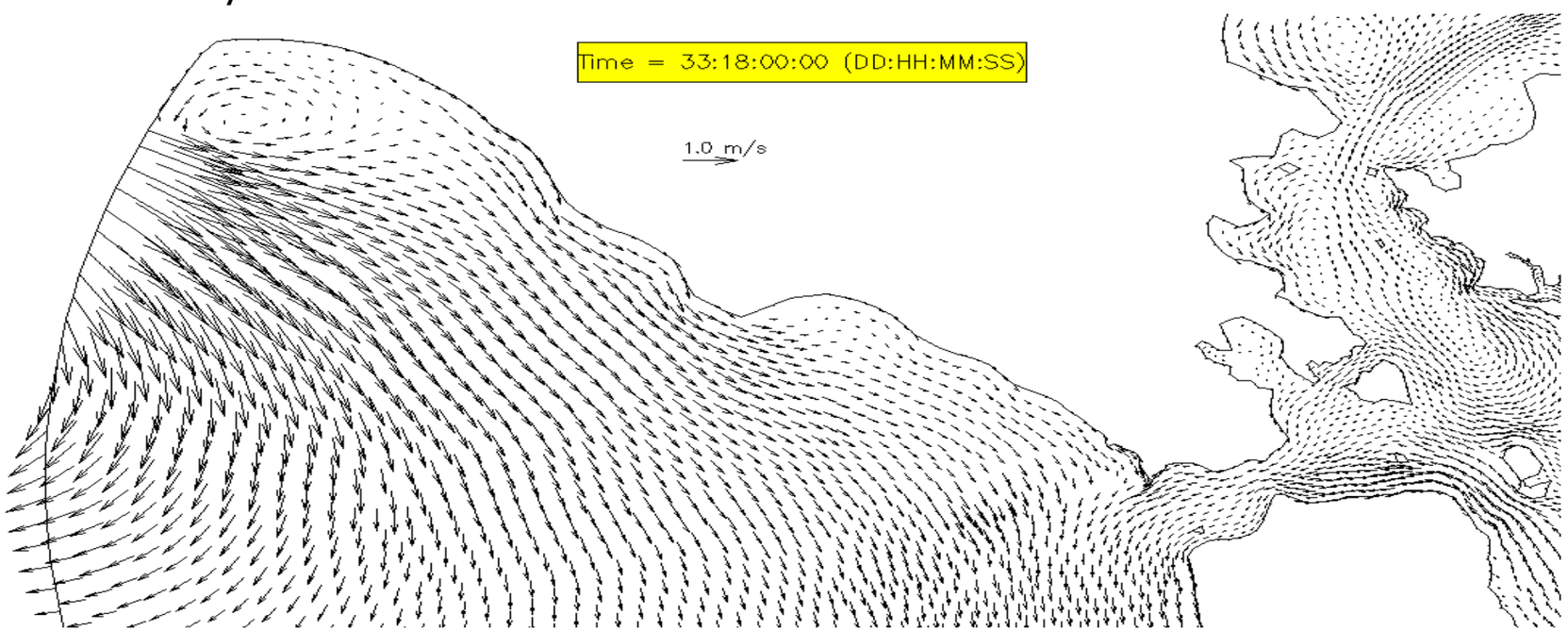
Make sure that channels are not 'blocked' by using at least 1 row of *always-wet* elements



Horizontal b.c. for velocity: uv3D.th.nc

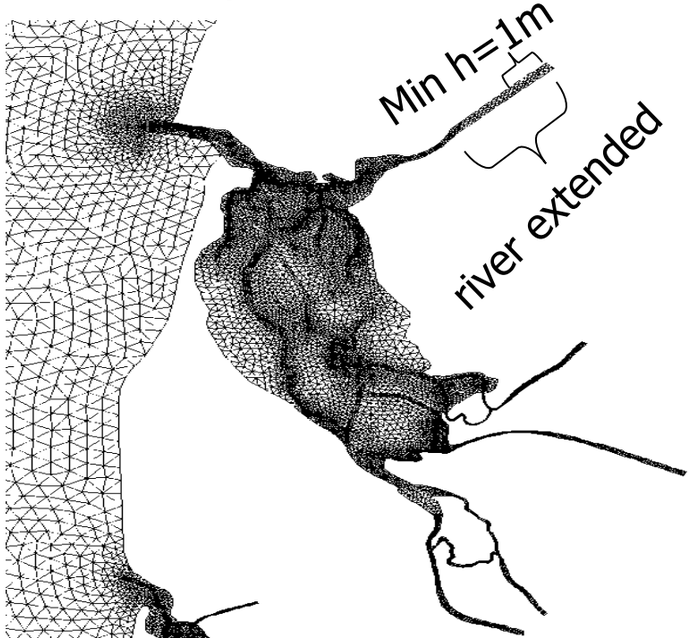
What's wrong with my open boundary ?

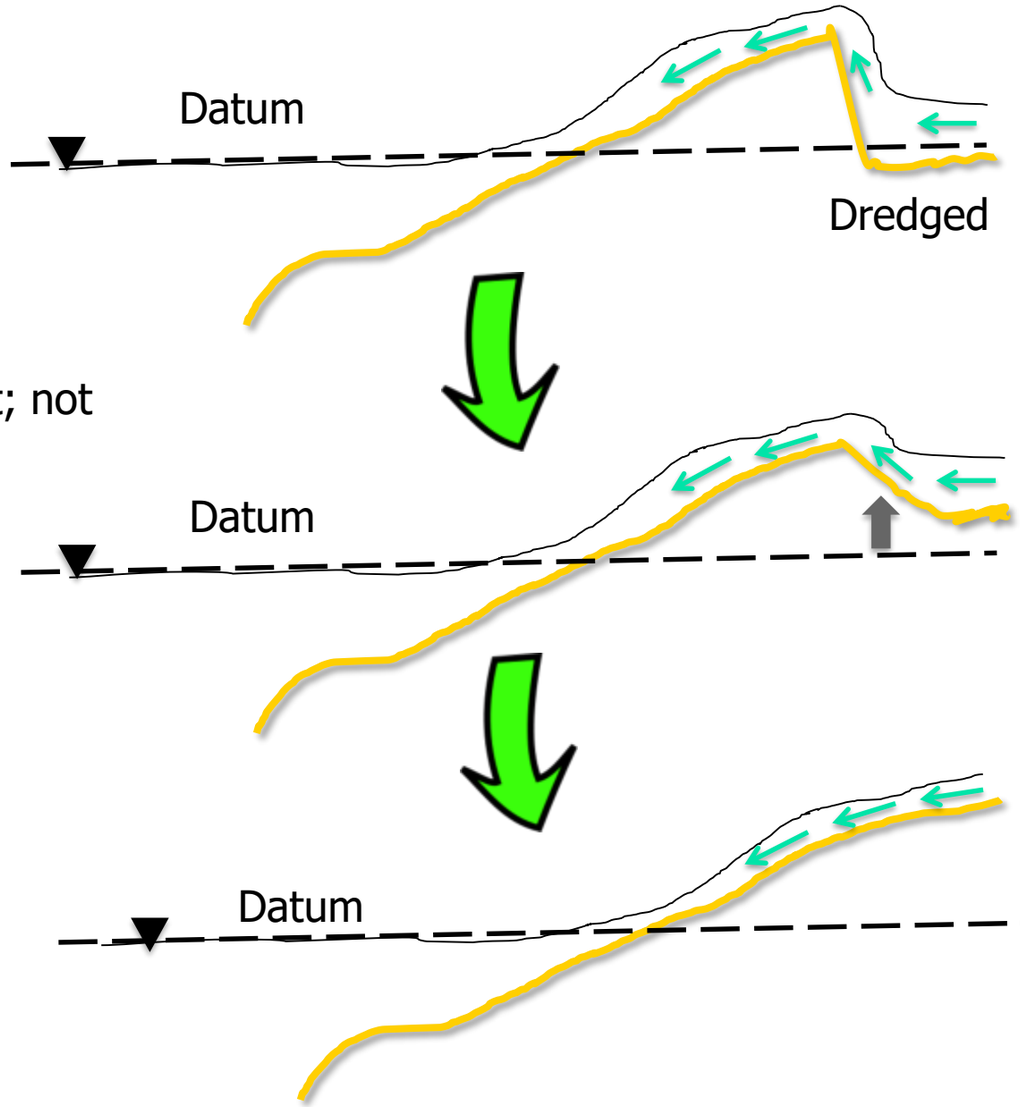
- Incoming velocity is required but often unavailable
- Use 1-way nesting to generate uv3D.th.nc
  - Can be combined with a non-tidal model (e.g. HYCOM)
- Use 2D model for the 'larger-domain', with most dissipative settings
  - θ=1
  - Large Manning's *n* near bnd
  - indvel=1; inunfl=0
  - Often you can 'cheat' by using the same grid for 'large domain' and 'small domain' if you don't care too much about accuracy near the boundary



River boundary

- To prevent drying near the river boundary, impose a min depth in ~2 rows of elements
- Datum issue in upstream rivers is severe
  - Flow b.c. alone should suffice
  - Initially "dry" river
    - Option 1: dredged inflow boundary
    - Option 2: elev.ic (with nramp_elev=1)
    - Option 3: bed deformation module
    - Option 4: point sources (combined with elev.ic)
  - Implicitness factor: θ=1 stabilizes for channel flooding
  - Damping of tides near river end: depth, friction
  - elev+flow b.c.: over-specification to get stages right; must be consistent; not usually recommended as essential b.c. overwrites Neumann b.c.





Tuning TVD transport for performance

$$\Delta t' \leq \frac{V_i}{\sum_{j \in S^-} |Q_j| (1 - \frac{\varphi_j}{2} + \delta)}$$

A workflow for adjusting the grid for TVD:

1. Generate a 1st grid with reasonable resolution and test-run it with TVD and an appropriate h_tvd (e.g. 6m); use average flow condition (it is expected that high-flow condition will lead to less efficiency);
2. *After a dynamic equilibrium is established for some time*, plot out fort.17 (format: time_step, nitr, where nitr is # of sub-cycles in transport) to see if on average the # of sub-cycles is reasonable (Δt_b = Δt/nitr). Resist the temptation to do this too soon, as initial re-organization of the salt often leads to small time steps but sometimes the steps become larger later;
3. If sub-step is too small in general, use **TVD_analyzer.pl** (run in same dir as hgrid.gr3) to find "culprit" elements; the outputs from this script is TVD_upwind.prop which can be loaded into xmgredit5 for visualization. Specify a tolerance time step (dtb0) for this script. Remember: this script will write out cumulative max up to the most current time step, and so start with a small dtb0 (e.g. 2 sec) and you'll likely to see only a few violating elements. Re-generate the grid to coarsen these elements to relieve the bottleneck.
4. Iterate between steps 2 and 3 until a reasonable grid is obtained

Additional control in TVD

- h_tvd: threshold depth where TVD is invoked; should be min depth of stratification
- tvd.prop: turn on/off TVD/upwind

Other considerations:

- **Vgrid: LSC² helps alleviate sub-cycling in shallows**
- **Use upwind in very high resolution areas or in regions of no stratification**

LSC² script: gen_vqs.f90


```
      !m_vqs: # of master grids
      m_vqs=18
      dz_bot_min=0.3 !min. bottom layer thickness [m]
      allocate(hsm(m_vqs),nv_vqs(m_vqs),a_vqs(m_vqs))
      hsm(1:7)=(/(7.+3*(i-1),i=1,7)/)
      hsm(8:18)=(/28.,32.,37.,43.,50.,58.,67.,77.,88.,100.,120./)
      nv_vqs(1:m_vqs)=(/(14+2*(i-1),i=1,m_vqs)/) !# of levels for each master grid (increasing with depth)
...
     !Other consts.
     !Stretching const. for the 1st master grid and also for depth <= hsm(1)
     !|a_vqs0|<=1 (1: skew toward bottom; -1: toward surface; 0: no bias)
      a_vqs0=-0.5
...
 do m=1,m_vqs
       do k=1,nv_vqs(m)
         sigma=(k-1.0)/(1.0-nv_vqs(m))

        !Alternative transformations below
        !Option 1: quadratic
         !a_vqs(m)=max(-1.d0,a_vqs0-(m-1)*0.03)
         !tmp=a_vqs(m)*sigma*sigma+(1+a_vqs(m))*sigma !transformed sigma
         !z_mas(k,m)=tmp*(etal+hsm(m))+etal

         !Option 2: S
         theta_b=0
         theta_f=1.5+0.03*(m-1)
         cs=(1-theta_b)*sinh(theta_f*sigma)/sinh(theta_f)+ &
     &theta_b*(tanh(theta_f*(sigma+0.5))-tanh(theta_f*0.5))/2/tanh(theta_f*0.5)
         z_mas(k,m)=etal*(1+sigma)+hsm(1)*sigma+(hsm(m)-hsm(1))*cs
       enddo k
     enddo m
```

Controls near bottom resolution (used to consolidation of near-bottom levels)

$z=Hs+\eta$

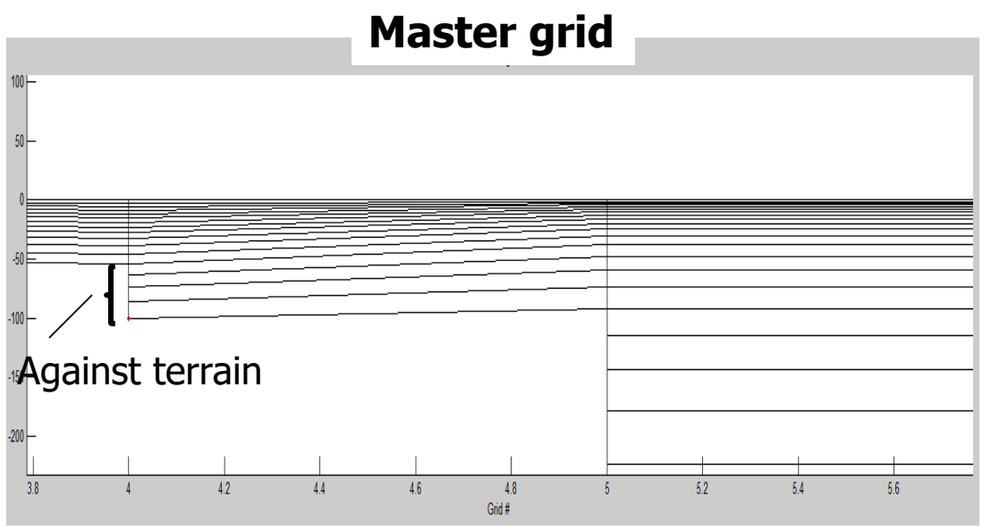
$s=a\sigma^2+(1+a)\,\sigma \qquad (-1 \le \sigma \le 0)$

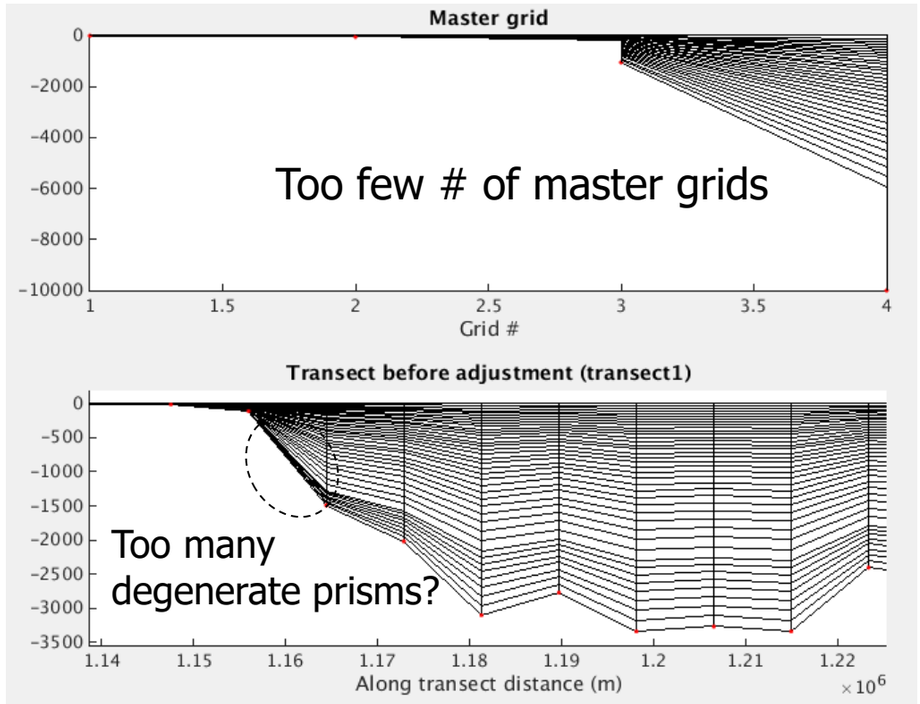
a_vqs0

S transformation (or you favorite coordinates here)
(There are more options also; see gen_vqs_Rutgers.f90)

LSC²: tuning

- Generally, the master grid and the final transect grid should be 'pleasing to the eye', with resolution applied where you wanted
  - ✓ Use many transects to catch potential issues (tip: assign different depths to each 'segment' for better viz)
  - ✓ Prefer to resolve surface instead of bottom but…
- Symptom #1: going against terrain
  - ✓ Solution: adjust # of levels, stretching constants…
- Symptom #2: too many degenerate (shaved) cells near bottom
  - ✓ Solution: adjust spacing in some master grids (dz_bot_min, too few # of master grids, stretching constants, large differences in # of levels between adjacent master grids, etc)



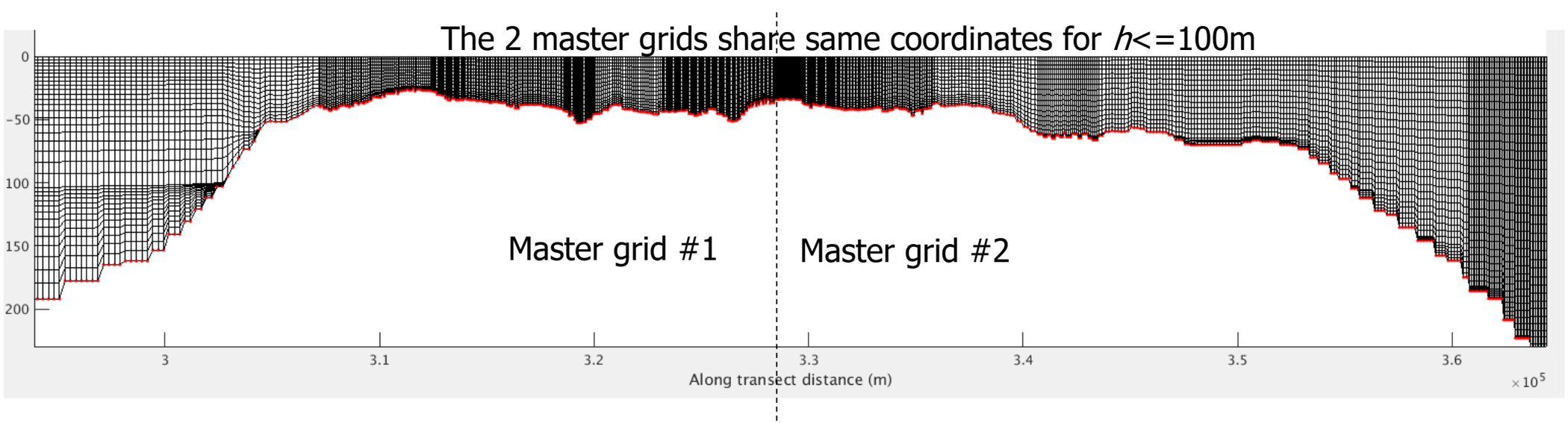


LSC^2: performance consideration

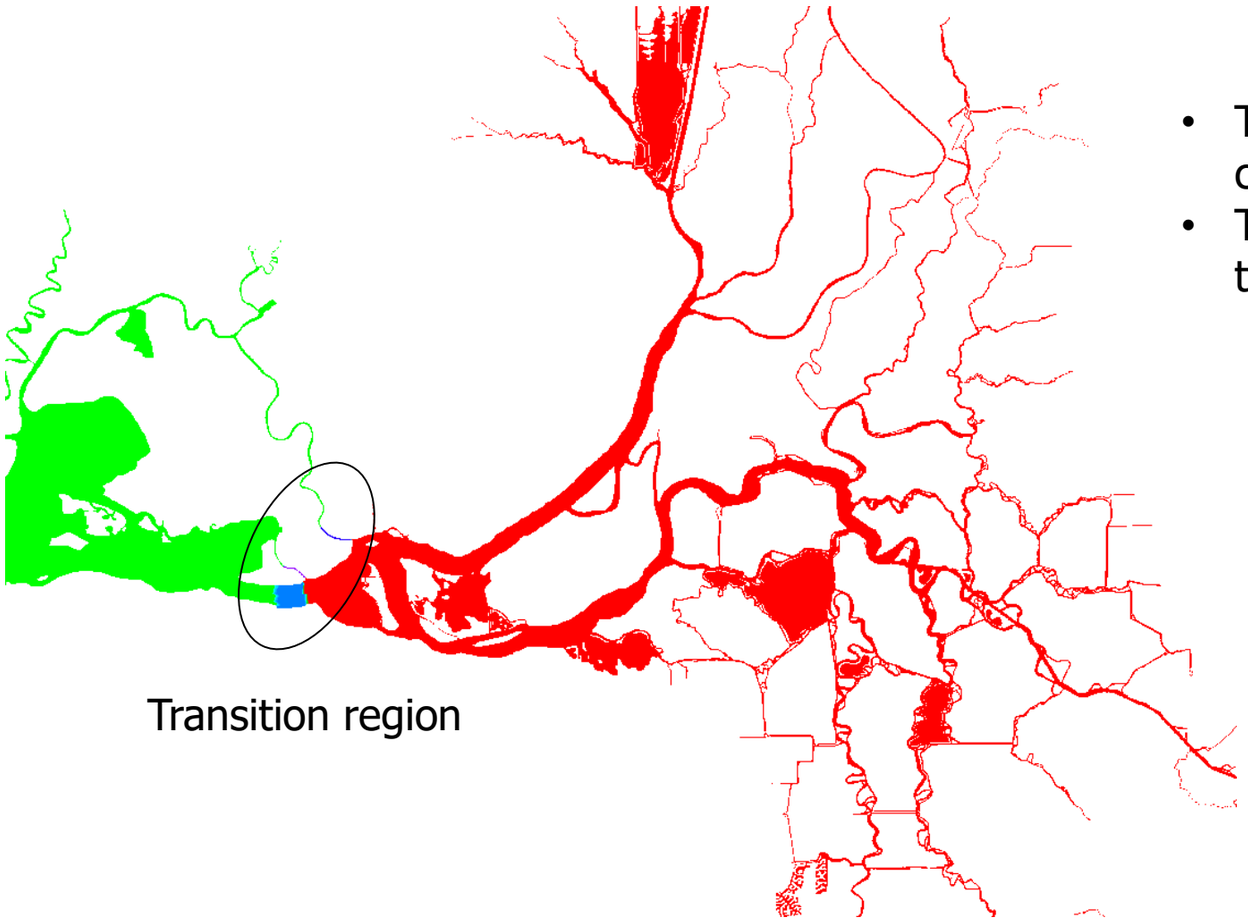
- Can use constant z after certain depths to save cost (esp. in deep ocean)
- Use multiple sets of master grids in different parts of domain (take care of transition)
- Optimize on a regional basis

Last 3 master grids

```
z_mas(1:nv_vqs(5),6)=z_mas(1:nv_vqs(5),5)
z_mas(1:nv_vqs(6),7)=z_mas(1:nv_vqs(5),5)
z_mas(1:nv_vqs(6),8)=z_mas(1:nv_vqs(5),5)
z_mas(1+nv_vqs(5):nv_vqs(6),6)=(/-400,-460,-520,-590,-660,-740,-830,-930,-1050/)
z_mas(1+nv_vqs(5):nv_vqs(7),7)=(/-400,-460,-520,-590,-660,-740,-830,-930,-1050,-1200,-1400,-1600,-1800,-2000/)
z_mas(1+nv_vqs(5):nv_vqs(8),8)=(/-400,-460,-520,-590,-660,-740,-830,-930,-1050,-1200,-1400,-1600,-1800,-2000,-2400,-2900,-3500,-4200,-5000.,-7000,-1.e4/)
```



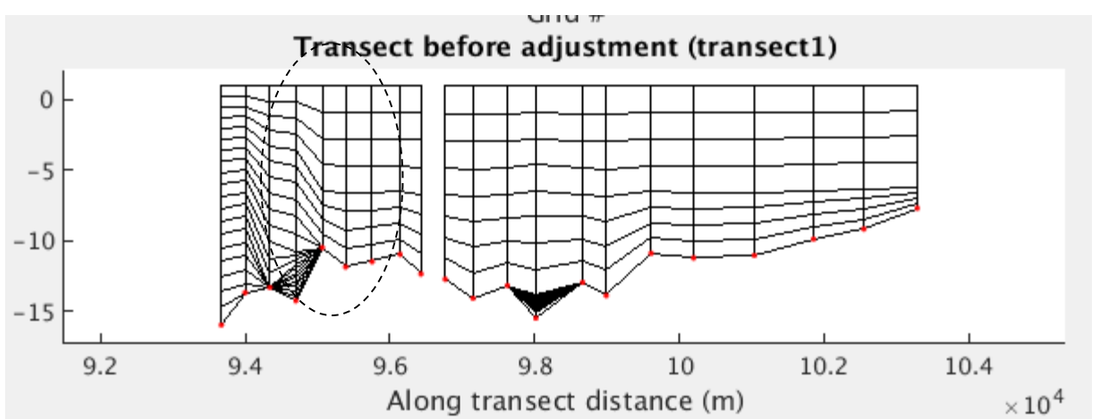
Regionally based LSC² grid



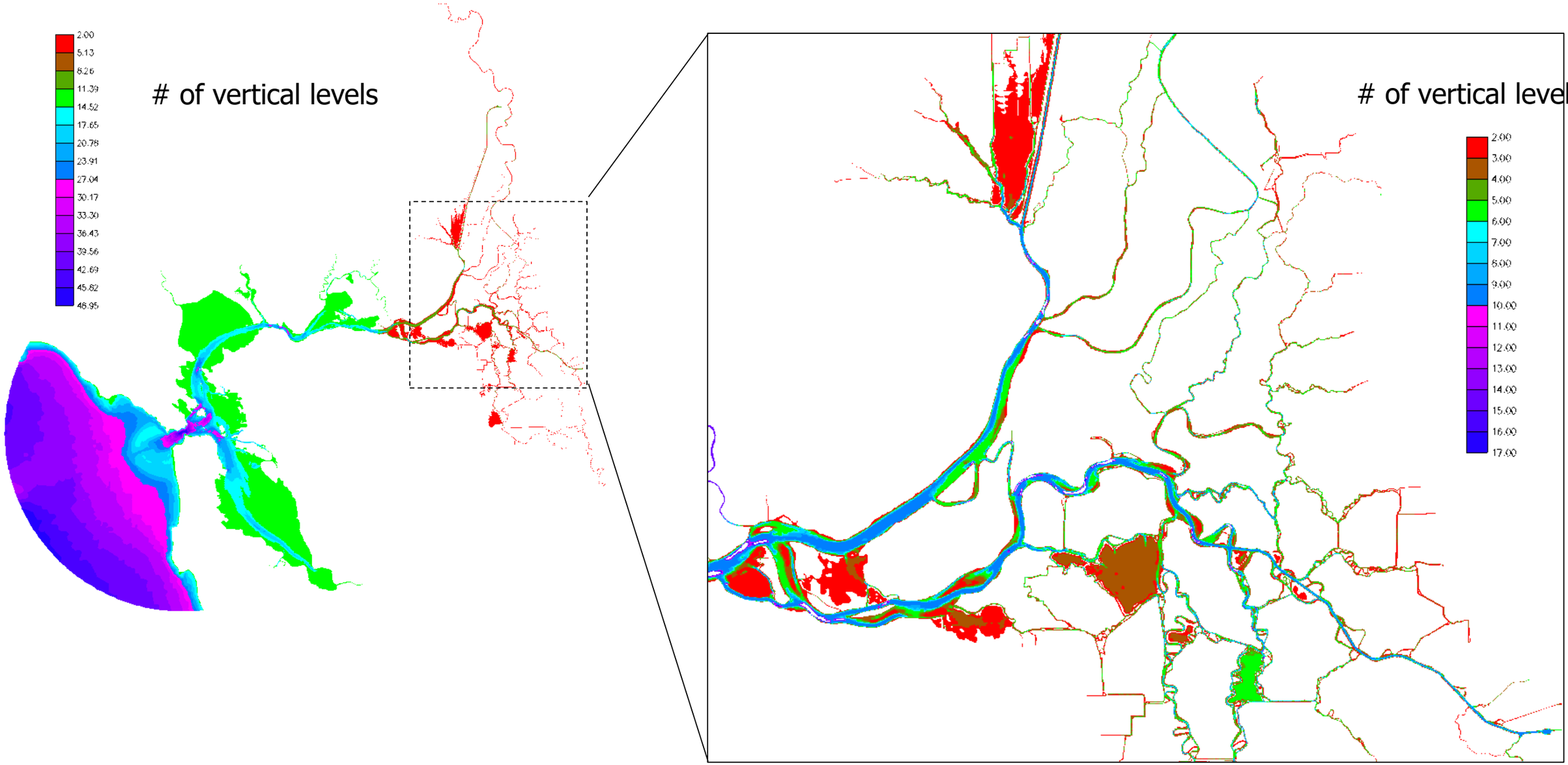
Transition region

- Transition the master grid from Bay into shallow Delta to save cost
- Tip: use Cd=0 in the non-smooth transition (to effectively turn the 3D model into 2D) to avoid momentum attenuation

Non-smooth vgrid (set Cd=0)

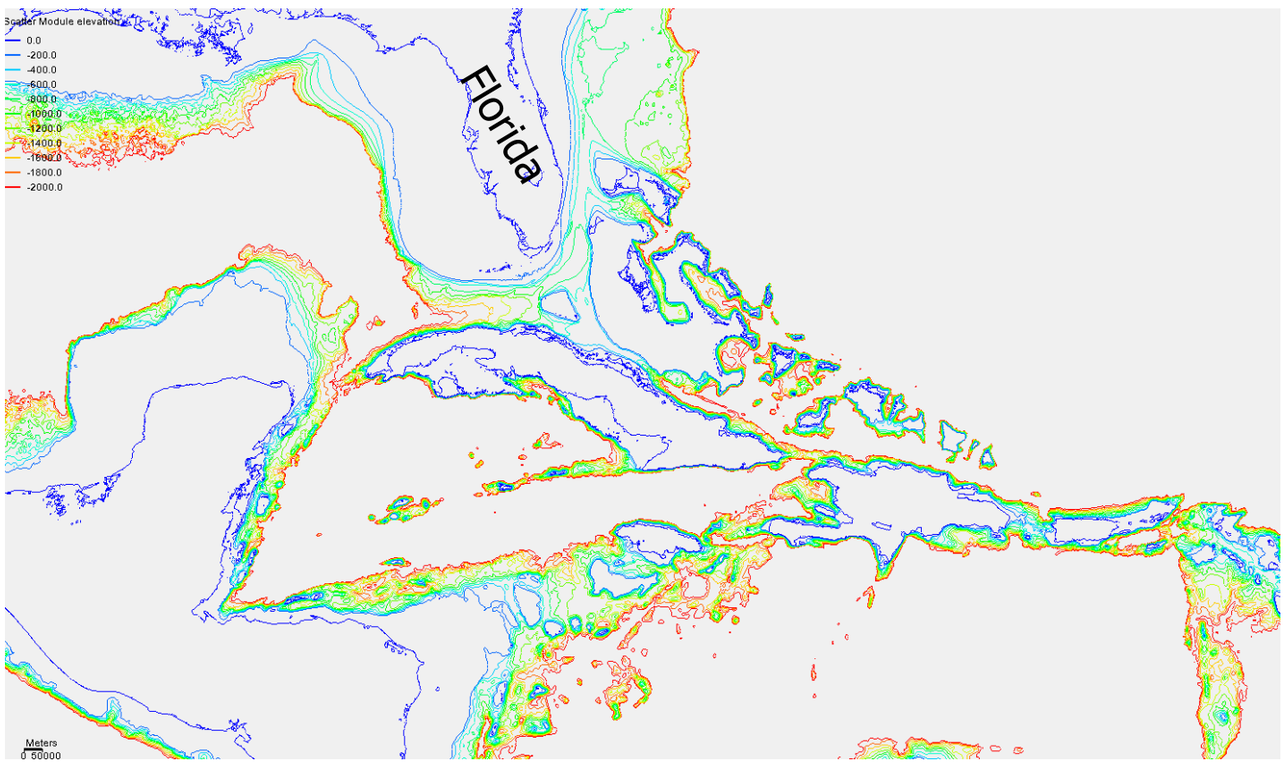


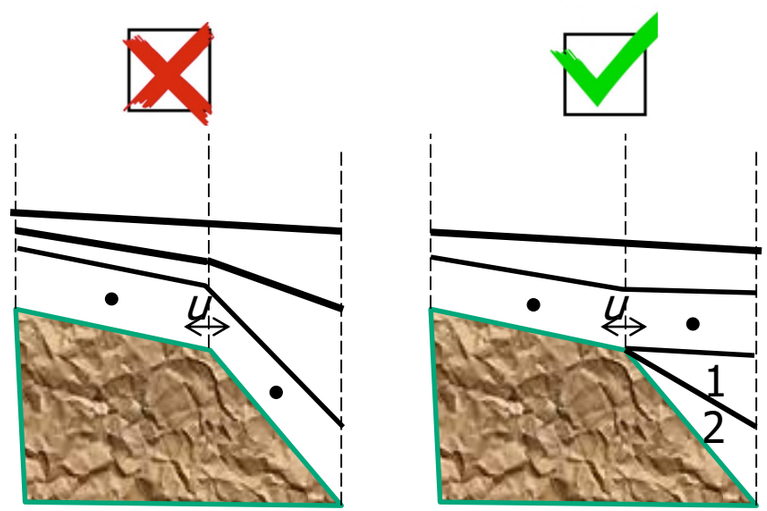
Regionally based LSC^2 grid



The case of under resolution in eddying regime

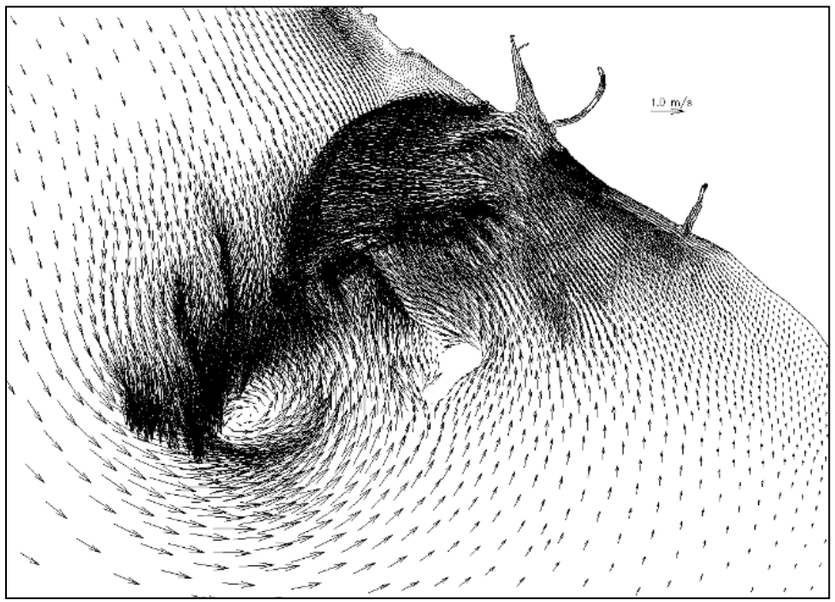
- Under resolution occurs frequently in areas of steep slopes, mostly to save cost in areas of no significant interest
- In general, under resolution in the horizontal dimension should be matched with an under resolution in the vertical, often taking advantage of many degenerate prism near the bottom and a non-smooth LSC^2 there. This reduces PGE
- The upcoming combined Laplacian and bi-harmonic viscosity option can be used to stabilize the momentum over the steep slopes

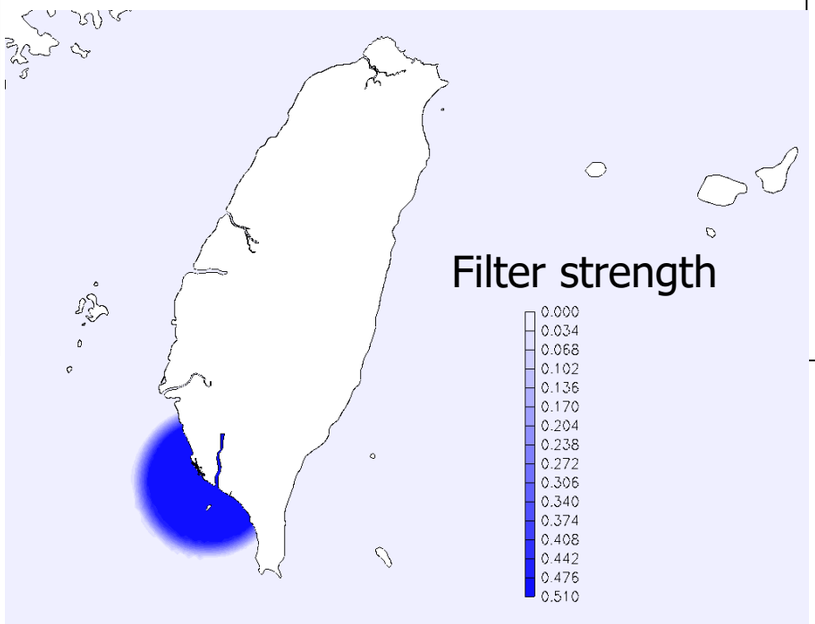




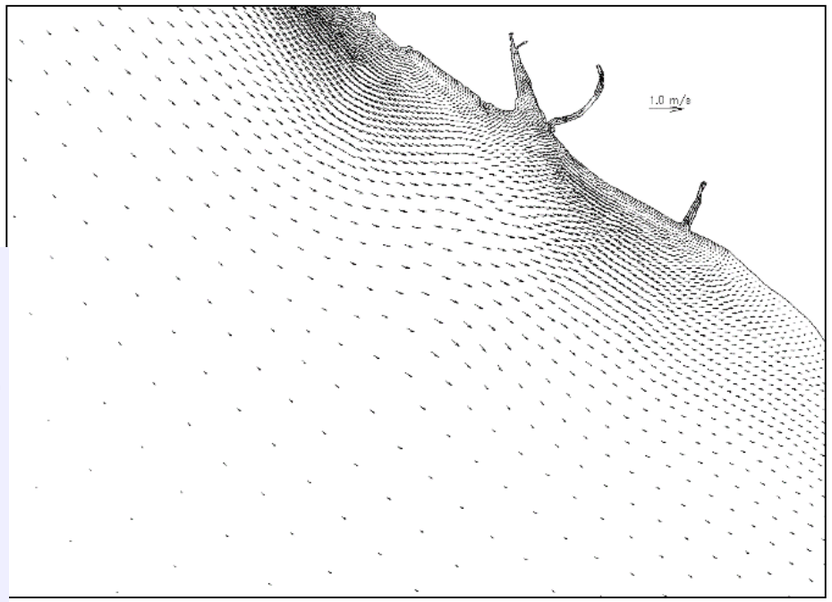
Transition between eddying and non-eddying regime

Sharp grid transition, with insufficient dissipation





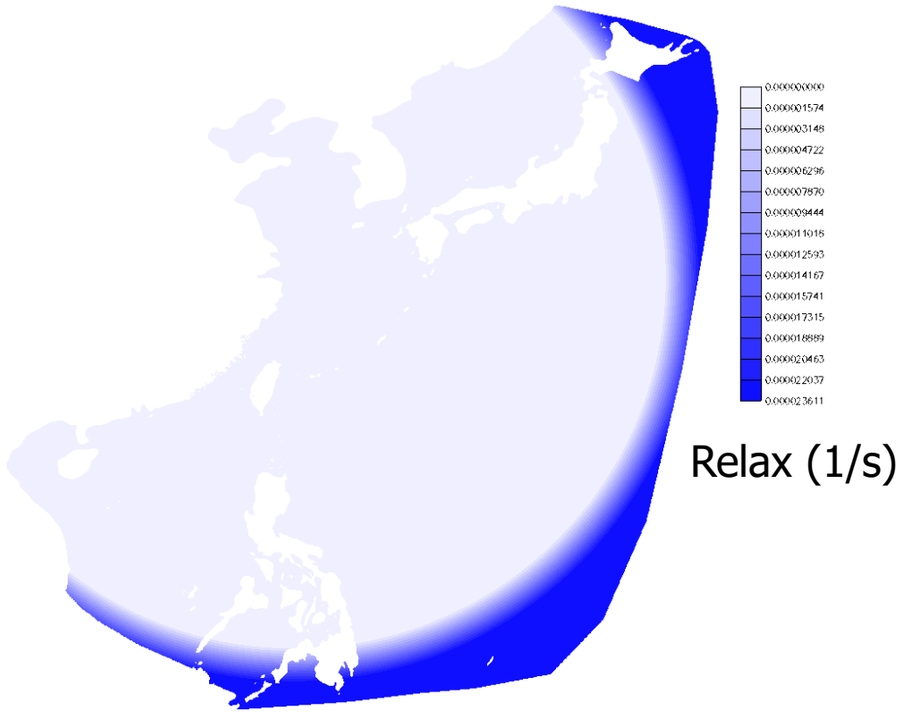
Smooth transition (no additional dissipation)



- Sharp transition in this critical regime, enabled by UGs, may trigger artificial accumulation of energy, without adequate dissipation mechanism
- Resolution gradient-aware viscosity may be the ultimate solution (e.g., combination of Laplacian & bi-harmonic viscosity)
- Two pragmatic solutions
  - i. Make sure the grid transition is smooth (in fact smooth grid is required in eddying regime as well)
  - ii. If non-smooth transition has to be used, add numerical dissipation in the form of filter or combined viscosities

Coupling with large-scale model (HYCOM)

- Need from HYCOM (scripts in svn directory)
  - b.c. (elev, u,v,T,S)
    - T,S: can be interpolated from (daily) HYCOM and used directly as [TEM,SAL]_3D.th.nc
    - SSH, u, v from HYCOM: as elev2D.th.nc and uv3D.th.nc (no tides yet; combine with OnewayNest to get tides)
  - Nudging for tracers (T,S)
    - [TEM,SAL]_nudge.gr3: relaxation factors (0 - $1/T_r$). May adjust T_r to get a smooth transition (usually ~1 day)
    - [TEM,SAL]_nu.nc: HYCOM values (can be junks outside the nudging zone)



Relax (1/s)

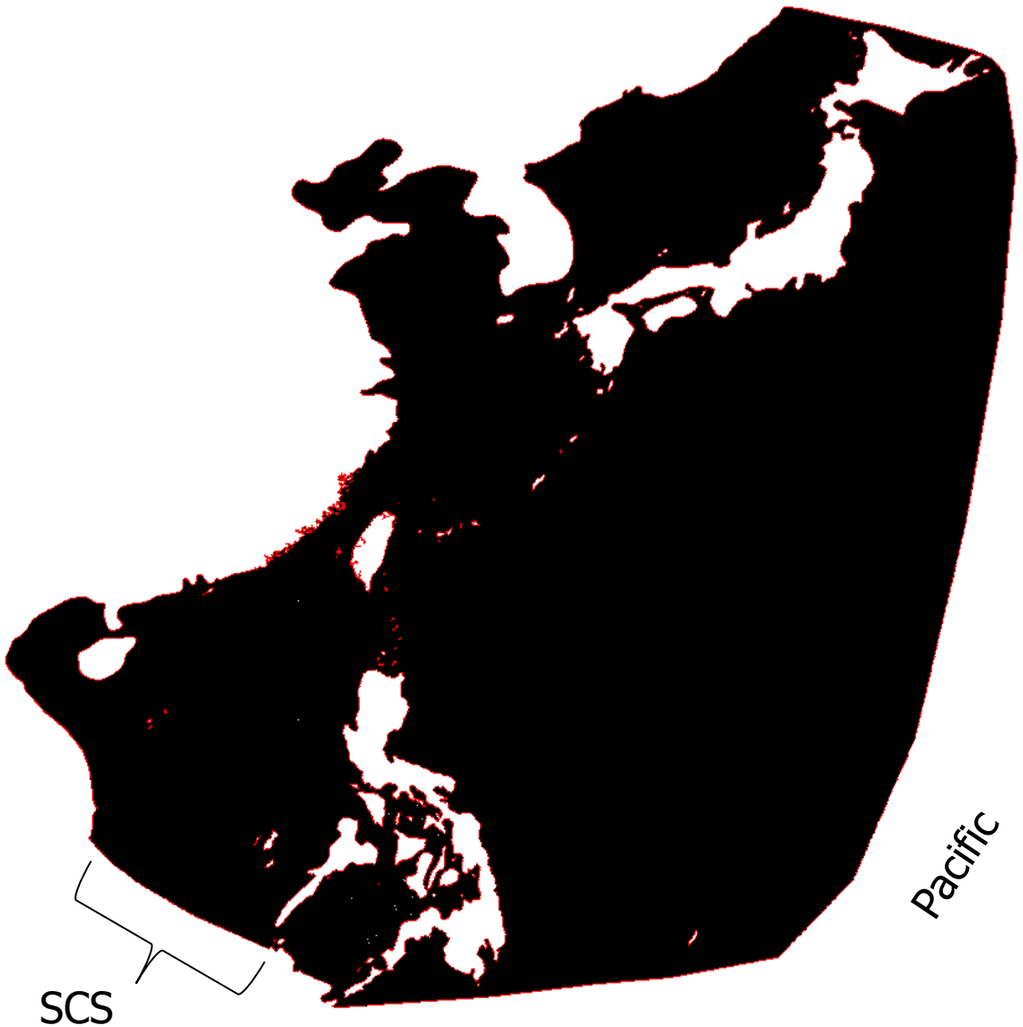
```
netcdf TEM_nu {
dimensions:
        node = 433559 ;
        nVert = 48 ;
        ntracers = 1 ;
        time = UNLIMITED ; // (121 currently)
variables:
        double time(time) ;
                time:long_name = "simulation time in days" ;
        float tracer_concentration(time, node, nVert, ntracers) ;

// global attributes:
                :_NCProperties = "version=1|netcdflibversion=4.4.1.1|hdf5libversion=1.8.18" ;
data:

 time = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
    20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
    38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
    56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
    74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
    92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107,
    108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120 ;

 tracer_concentration =
  -9999,
```

Add tides

- HYCOM does not have tides (actually it may be better to explicitly filter out any high-frequency signals in it)
- Tidal signals are added into HYCOM SSH and u,v at the boundary
  - Tidal potential should be added in the interior of the domain
- Tidal signals can be generated by using OneWayNestScripts/interpolate_variables7.f90, to get elev2D.th.nc and uv3D.th.nc
  - It's generally OK to use same background and foreground grids when the domain is large
  - Use a 2D b-tropic model for tides only (no wind)
  - The script will extract the time series at the ocean boundary
- Add tidal elevation
  - Can use iettype=5: tidal amplitudes/phases are specified in bctides.in, whereas non-tidal HYCOM is in elev2D.th.nc
- Add tidal velocity: use a simple script to combine two uv3D.th.nc (from HYCOM and from OneWayNestScripts)
  - The script basically interpolates in time and adds up the 2 components
- Since the velocity is never perfect at the boundary, the tracer concentration needs to be nudged in a zone near the ocean boundary
  - The incoming vel is more important, especially for indvel=0 which has lower dissipation and thus requires more stabilization

Case study: Kuroshio

- Step 1: grid generation for both 2D b-tropic and 3D b-clinic models
  - Start from a moderate resolution in eddying regime and refine as needed
  - Smooth transition from deep ocean to shelf break
  - Add nearshore features (channels, rivers, jetties); non-smooth grids are fine
- Step 2: do 2D b-tropic model
  - Get tidal amplitudes and phases from a global model (e.g. FES2012)
  - Make sure to output elev, hvel, at a tide-resolving frequency (e.g. hourly)

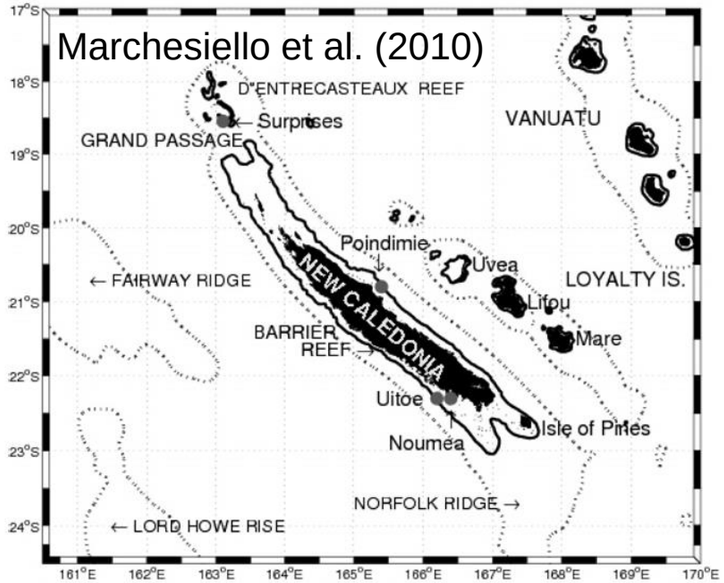


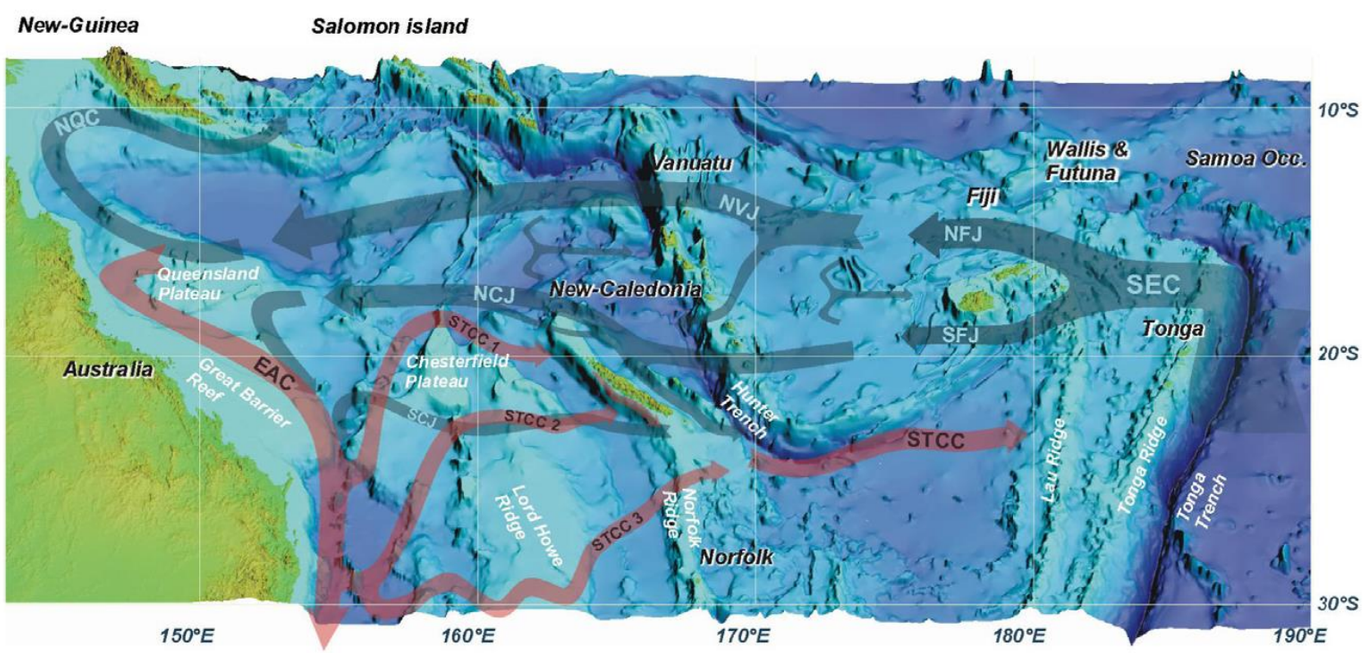
Kuroshio: baroclinic model

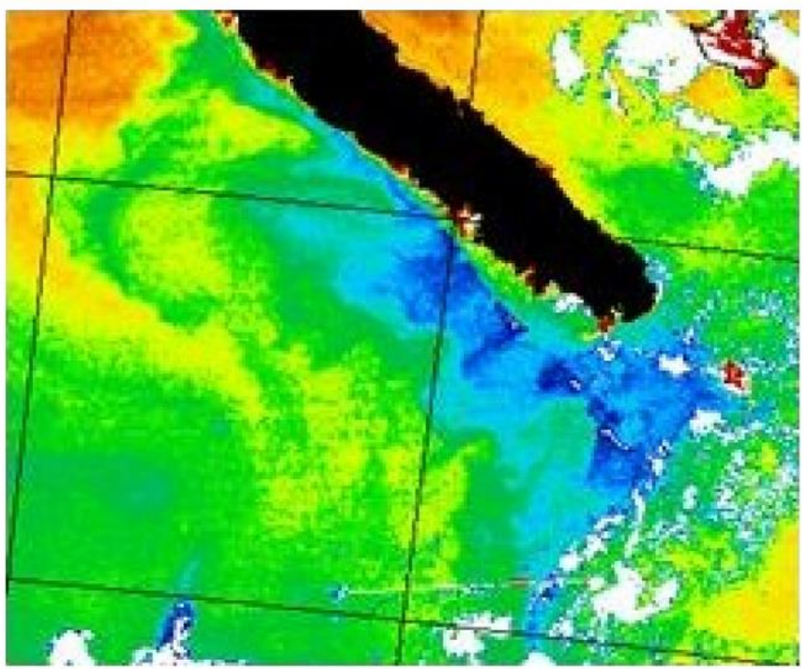
- Prepare a LSC^2 grid
  - Plot out many transects for review
- Download and prepare HYCOM files (e.g., [ST,UV,Z]_2013.nc)
- Download CFSR atmospheric forcing and prepare sflux/
- Prepare nudging grids ([TEM,SAL]_nudge.gr3): elliptic zone or fixed distance from ocean boundary
- Prepare nudging inputs (TEM,SAL]_nu.nc) from HYCOM
  - Tip: exclude part of the domain outside the nudging zone in include.gr3 to speed up
- Prepare hotstart.nc and *.th.nc from HYCOM
  - hotstart.nc and [TEM,SAL].th.nc are final
  - elev2D.th.nc and uv3D.th.nc will be combined with tidal components later
- Prepare tidal b.c. from 2D run: elev2D.th.nc and uv3D.th.nc
- Combine tidal and non-tidal elev, uv as final elev2D.th.nc and uv3D.th.nc
  - If you use iettype=3 (amplitudes and phases) in your run, you just need to change it to '5' in bctides.in, and use elev2D.th.nc from HYCOM only
- indvel=ishapiro=0, ihorcon=2
- If the grid is not smooth in the transition regime, use ishapiro=-1 (with shapiro.gr3) to add dissipation locally
- Enjoy!

Case study: New Caledonia reefs

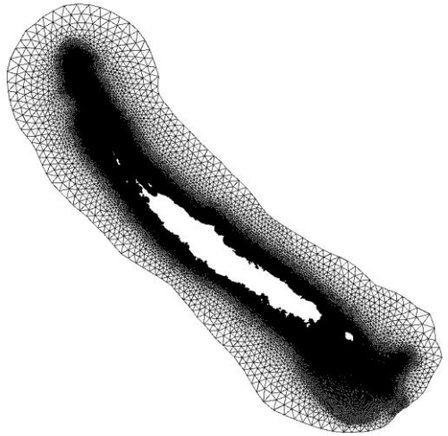
- South Pacific island surrounded by reefs
- Reefs are sentinel species for environmental changes
- There is a need to understand the flow around the reefs
  - Persistent upwelling wind on 1 side of the island, but only southern part of the island shows upwelling pattern

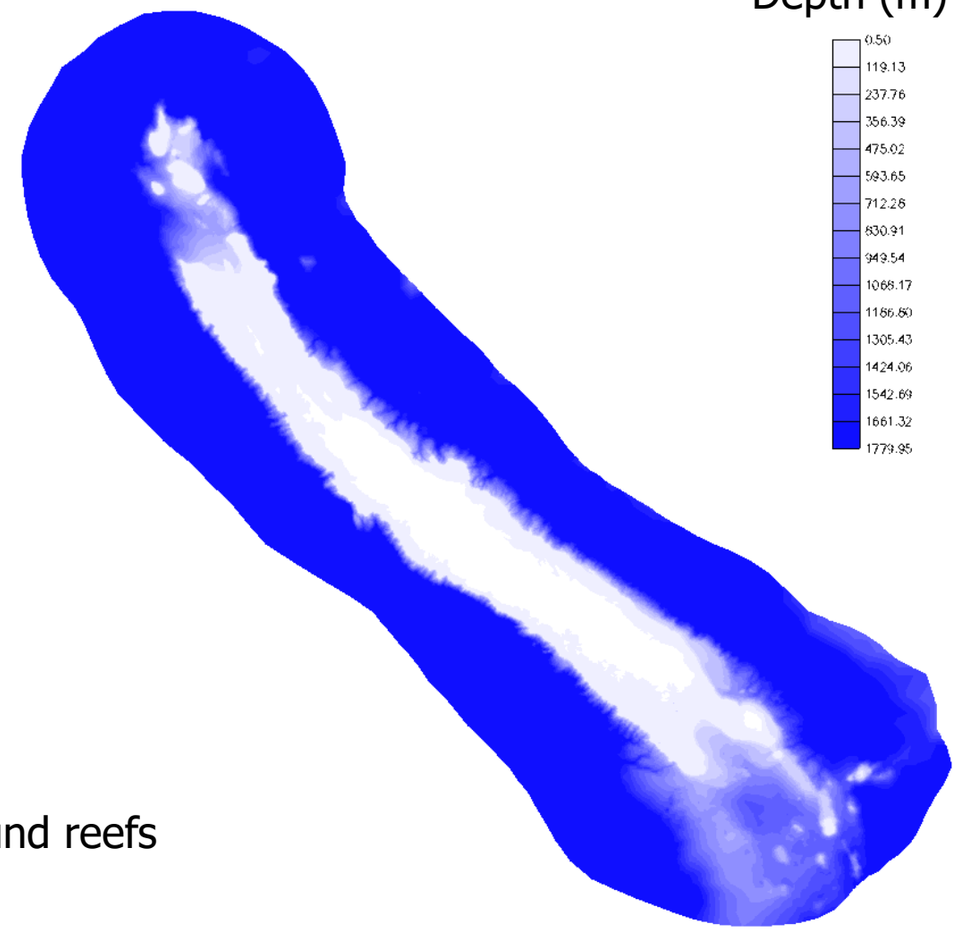


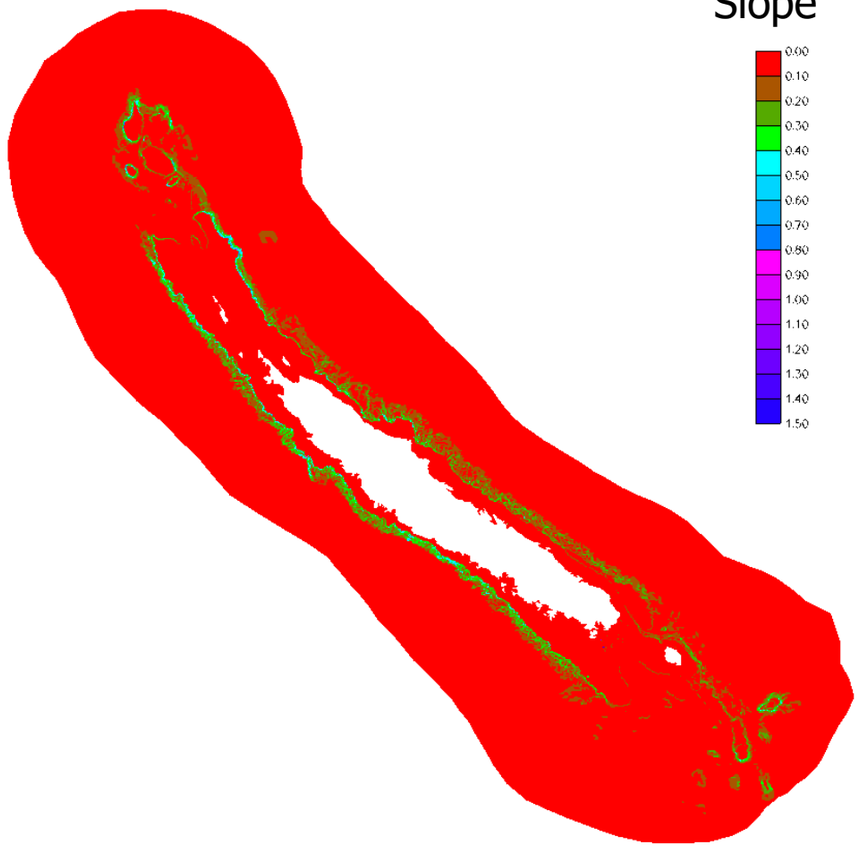




Bathymetry







Steep slopes around reefs

Model set-up

- Modest horizontal grid resolution around the reefs (~500m)
- LSC^2 grid
- PGE leads to sub-grid noise in momentum, which needs to be stabilized properly
- Non-eddying option (ishapiro=1, ihorcon=0) works better due to larger dissipation
- Possible to combine this option with eddying option elsewhere