Spherical Multiple-Cell (SMC) grid utility tools

Jian-Guo Li
Met Office, Exeter, UK
Email: Jian-Guo.Li@metoffice.gov.uk

28 June 2025

1. Introduction

This SMC grid tool package consists of SMC grid generating and test programs in Python and Fortran languages and Linux shell scripts. They are classified by language types and collected in self-explained sub-directories. Tool programs are designed to be as simple as possible so that no extra software is needed except for the essential Linux operation system, Python and Fortran compiler. The package is uploaded to Github for public access under the BDS license. Interested users can download the SMCGTools full package from the web site at:

https://github.com/ww3-opentools/SMCGTools

You need to copy the whole `SMCGTools/*` to your disk space and modify the paths inside some program files before you use it. A possible improvement is to set up a working path to `SMCGTools/` and pass it as the default top directory.

Step by step guides to use this tool package are given, using an example grid SMC61250 for demonstration. Users may modify their input files to change the spatial resolution for your specified requirements. Starting with preparation of your own bathymetry file, which is too large to be stored on Github site, you may go through all the steps for generating the grid cell array, trimming the cell array, creating the face arrays, and running the propagation test.

Users' feedback about the tool package is welcome and improvement of any of the tool programs or other contribution will be added to the tool package and properly acknowledged.

1. Prepare a bathymetry file

One global bathymetry data, GEBCO_2022, are available for download from:

https://www.gebco.net/data_and_products/gridded_bathymetry_data

The data set was published in June 2022 and is a global terrain model for ocean and land, providing elevation data, in meters, on a 15 arc-second interval grid. It consists of 43200 rows x 86400 columns, giving 3,732,480,000 data points. The GEBCO_2022.nc file is provided in NetCDF 4 format and is about 7.5 Gb in unpressed form. It should be quoted as

GEBCO Compilation Group (2022) GEBCO 2022 Grid.
(doi:10.5285/e0f0bb80-ab44-2739-e053-6c86abc0289c)

For SMC grid generating, the bathymetry data must be reduced to the required size-1 cell resolution. For the following example grid, the size-1 cell length will be set at Δλ = 0.087890625° and Δφ = 0.05859375°. The longitude resolution is chosen so that the total number of points along one longitude parallel circle is 4096 = 2**12, which eases the longitudinal merging at high latitudes for a global grid including the Arctic. The full latitude range from 90S to 90N will cover 3072 size-1 cells. This latitude cell length is roughly 6 km, so this reduced bathymetry is also referred as global 6 km bathymetry. It has equal number of size-1 cells on the north and south hemispheres so the Equator could be used as the reference point for cell latitude indices.

As the GEBCO_2022 bathymetry data are quite large and awkward to handle on a desktop machine, a first step reduction is done by just average a fixed number of points into one along both directions. For instance, a factor of 20 is chosen for the longitude direction and it reduces the 15 second resolution to 5 minutes with 4320 points in one parallel circle. For the latitude direction a factor of 12 is used and it reduces the 15 second resolution to 3 minutes or 3600 points along the latitude range. The temporary data are then interpolated to the required size-1 resolution in the longitude direction (from 4320 to 4096 points) and the latitude direction (from 3600 to 3072).

The first reduction step could be done with a Python program with the command line after modifying the input file `configreduce.txt` to point to your `BEBCO_2022.nc` file.

```
SMCGTools/PySMCs/ python reduce_interp.py configreduce.txt
```

Which created the temporary bathymetry in tmpfls/ and it is renamed by

```
SMCGTools/tmpfls/ mv GEBCO_reduced_20_12.nc ../Bathys/Bathy083_050.nc
```

The interpolation step could be done with the following command lines:

```
PySMCs/ python reduce_interp.py configinterp.txt
tmpfls/ mv Bathy_interpolated_*.nc ../Bathys/Bathy088_059deg.nc
```

The final bathymetry file `Bathys/Bathy088_059deg.nc` will be used for generating the SMC61250 grid. The advantage of this netCDF bathymetry file over the ASCII one is that it has merged both the elevation and the obstruction arrays into the same file, apart from the increase spatial resolution.

2. Generate and trimming SMC61250 grid

A sample 4-level (6-12-25-50 km) grid, SMC61250, is generated with the Python program, `SMCGloblCell.py`, which reads the `GridInfo61250.txt` input file and the bathymetry file, `Bathy088_059deg.nc`, created in section 1. The input file `GridInfo61250.txt` has the following lines of the SMC61250 grid information:

```
PySMCs/ cat GridInfo61250.txt
 SMC61250 4 level grid zlon zlat dlon dlat for size-1 cells.
  0.0  0.0  0.087890625  0.058593750
  ../tmpfls/   ../DatGMC/               Working and cell array dirs.
  1  Number of polar cells or polar parts. 0 implies no Arctic part.
  ../../S61250Tx/ww3.    ../OutDat/      SWH and PropTest file dirs.
```

The command line for generating the SMC61250 cell arrays is

```
PySMCs/ python SMCGloblCell.py GridInfo61250.txt
```

The water depth is derived by the difference from the bathymetry elevation to the mean sea level height. As a results, some inland lakes, like the Caspian Sea and the Great Lakes, do not have the correct water depth because their lake surface levels are different from the mean sea level. To rectify the water depths for these inland water bodies, a separate Python program is used to create their local cells with their specified water surface levels. Cell array files will be saved in a temporary sub-directory `SMCGTools/tmpfls/`, including an Arctic part cell array file as this is a full global grid.

The Python program, `Lakes625Cells.py`, created the 3-level (6-12-25 km) Caspian Sea and Great Lakes cells with the same bathymetry file but different water surface heights for different lakes. The smcellgen.py has a named parameter wlevel = 0.0 for dry cell specification. This parameter defines the water depth reference point and uses the sea level altitude as the default value. For inland lakes, the wlevel parameter could be assigned the actual water level elevation so that true water depth for the given lake could be derived. The depmin = 0.0 parameter is used to define the elevation of cell expanding domain, which could include dry cells if it is assigned a value above the water level. The depmin should be at least equal to wlevel so it is reset to be equal to wlevel if it is given a value lower than wlevel. The dshalw = 0.0 parameter is used to define the refined shallow water area. It should be lower than depmin for any refinement and it is reset to be equal to depmin if it is assigned a value higher than depmin.

Note all elevations or bathymetry input are assumed to be positive above sea level. Cell depth is calculated as depth = wlevel - bathymetry, so it is positive if bathymetry is below water level elevation. If the depth is negative, it means the cell floor is above the water level or it is a dry cell. Cells with depth = 0 are just at the water level and they are turned to be dry cells in the updated ww3_grid program. For tsunami model, these cells are sensitive to coastal inundation.

The individual lake cells are trimmed with awk scripts (stored in `Linuxs/zTrimming/`) to remove isolated cells or overlapping areas. First to remove the count/header line from each individual lake file and move it into `Linuxs/zTrimming/`, then use the Linux command line, taking the Ontario Lake cell array for example:

```
Linuxs/zTrimming/ awk -f awkSuperior Supr6125Cels.dat > Supr6125Cels.d
```

The trimmed Caspian Sea cells are saved in `Casp6125Cels.d` and the trimmed Great Lakes are merged into a single file `GtLk6125Cels.d`. The global SMC61250 cell array, `SMC61250Cels.dat`, is also edited to remove its count/header line and renamed `SMC61250Cels.d`. Then it is trimmed and merged with the individual lakes by the following Linux command lines:

```
Linuxs/zTrimming/ ls awkAmazon awkAustralia awkCaspiaNo awkGtLakeNo \
 awkBlackSea awkMeditern awkTurpBngl | ../trimtempcels ./SMC61250Cels.d
Linuxs/zTrimming/ cat GtLk6125Cels.d Casp6125Cels.d >> tempcels.dat
Linuxs/zTrimming/ ../countcell6lv tempcels.dat
Linuxs/zTrimming/ mv New_tempcels.dat ../../DatGMC/SMC61250Cels.dat
```

The first line trims the, `SMC61250Cels.d`, which is created by the Python program, `SMCGloblCell.py` and edited to remove the first count/header line. The second line append the trimmed Great Lakes and Caspian Sea cells into the trimmed global cell array. The third command line sorts the trimmed global cell array and add a new count/header line. The final Linux command line moves the new cell array to replace the old one in `DatGMC/`.

Note the Python programs must be executed within the `PySMCs` sub-directory because some called library files are store there. This may be changed by setting up a Python path so that you may run the Python program outside the `SMCGTools/PySMCs/` sub-directory.

The trimming scripts, awk*, are created manually by zooming local areas with Linux awk command and Python viewing programs, `latlon2ijs.py` and `smcviewdep.py`. How to create the trimming scripts will be demonstrated later in Part II when boundary cells are trimmed because they share the same trimming techniques.

One Python program `smcellgen.py` called by the `SMCGloblCell.py` program could be used separately to read a netCDF bathymetry data and created a SMC grid with specified resolution levels. This program may be adapted by advanced Python users to generate a different SMC grid, such as a regional grid.

The `Lakes625Cells.py` could be used as an example of regional grid generation. The merged cell array, `GtLk6125Cels.d`, could be used as a regional grid for the Great Lakes by sorting it with

```
Linuxs/zTrimming/ ../countcell4lv GtLk6125Cels.d
Linuxs/zTrimming/ mv New_GtLk6125Cels.d ../../DatGMC/GtLk6125Cels.dat
```

You may generate face arrays for the Great Lakes cell array and run the propagation test on it as a regional grid. Because the Great Lakes are isolated from other water bodies, there is no need for any boundary conditions.

One frequent request has appeared recent years for generating a regional wave model grid, matching exiting ocean and/or atmospheric model in coupled systems. The bathymetry from the existing model is usually over a regional rectangular domain. To create a SMC grid wave model grid matching the rectangular bathymetry could be done in two steps. First to create the ‘inner’ cells with the `smcellgen.py` function, which requires some extra bathymetry data beyond the grid cell area for multi-resolution level selection. This could be done with the exiting model bathymetry by limiting the cells inside a boundary zone. For instance, a 2-level SMC grid has been created for the Met Office AMM 1.5 km ocean model and the two lines of boundary rows and columns are used as boundary zone to generate the ‘inner’ cells for the SMC 1.5-3 km wave model grid. The second step is to generate the boundary cells in the boundary zone with the `smcellbdy.py` function and merge these boundary cells with the inner cells to form the full SMC 1.5-3 km grid. This is to meet the requirement of the coupled system so that the wave model covers all surface points of the ocean model. One Python program `PySMCs/AMM153kmCells.py` is used to generate the AMM 1.5-3 km wave model grid, with the 2 steps merged into it. Note that the boundary cells are limited to be size-2 cells as there is no room for multi-resolution level set up. The sea point criterion is also lowered for the boundary size-2 cells with at least a single sea point out of the 4 bathymetry points covered by the cell. This is to ensure all sea points are covered by the wave model grid for coupling purpose.

3. Visualise the global grid and zoom in selected regions.

The SMC61250 grid could be viewed by drawing the grid in a stereographic projection, which uses projection point 3 times of the sphere radius from the projection plane so that the Equatorial area distortion by the standard stereographic projection (2 radius from the projection plane) is mitigated. The projection is specified in the `PySMCs/steromap.py` and the grid plot can be generated by running the Python program, `SMCGloblGrid.py`, on a Linux command line:

```
SMCGTools/PySMCs/ python SMCGloblGrid.py GridInfo61250.txt
```

It will require a choice number for either a global view (0) or a regional view (1 for Euro-Arctic region, 2 for West-Pacific region and 3 for a hemisphere), or a conventional box format global view (4). The program will save the projection polygons for the selected view before drawing the grid plot.

The plot is saved in eps format in `SMCGTools/tmpfls/`. You may view the eps plot with Linux `gv` command or convert the eps file into a gif format for a quick view with Linux `xv`.

```
SMCGTools/tmpfls/ ls smc61250grdGlob.eps | ../Linuxs/conveps2gif
```

If you like a different regional view other than the provided ones, you may modify one of the regional view option inside the program `SMCGloblGrids.py` (or insert an extra option) by specifying a new set of parameters as below:

```
    if( pltype == 'Pacific'):
## West Pacific regional plot
        plon= 138.0
        plat=  18.0
        pangle=33.0
        clrbxy=[-13.6,  7.5,   7.5,  0.8]
        sztpxy=[ 15.0, 10.0, -10.0,  7.0]
        rngsxy=[-15.0, 15.0, -10.0, 10.0]
```

where `plon` and `plat` are the longitude and latitude of your new regional centre and `pangle` is the spanning angle from the centre to the edge of your selected region (at the default 10 unit radius). As the projection is like the orthogonal projection, the maximum spanning angle is 90 deg to cover one hemisphere. If your spanning angle is less than 90 degrees, the hemisphere is projected larger than the default circle of 10 unit radius. You may use the `rngsxy` range parameters to select you plotting area in this case. The above example selects a plotting area from -15 to 15 units in the x-axis and -10 to 10 units along the y-axis.

The `clrbxy` parameters specify the colour key box position and size in plotting units. It automatically chooses a horizonal or vertical colour bar by the x and y sizes of the colour box. The above colour box is a horizontal one, 7.5 unit long and 0.8 unit high and its SW corner is at the point `(-13.6, 7.5)` in the plot. You may position your colour key inside the plot over a blank land area or simply put it at the edge of the plot.

The `sztpxy` first two parameters specify the eps plot x/y-sizes in inches and the last two parameters specify where your text messages are printed in plotting units.

If you prefer the conventional regular grid box format presentation of the global grid, use the number 4: 'Regular' option. It will project the SMC grid into a rectangular box format with merged cells span their full range rather than mapping into evenly spaced regular mesh. This will save from converting any SMC grid field into regular meshed data for plotting in the box format later.

All grid eps plots are drawn in portrait format with the parameter `papror='portrait'`. This is for screen viewing convenience. You may use landscape orientation for ps printing files if you like.

The Gt Lakes regional grid can also be viewed with `PySMCs/GtLks6125Grids.py`, which reads the `GridInfoGtLks.txt` input file for grid information. Note the grid information file share the size-1 cell parameters as the SMC61250 grid but the resolution level is reduced by 1 as there is no 50 km cells in the Gt Lakes grid. There is also no polar cell in it.

The AMM153 grid can be viewed with the Python program, `PySMCs/AMM153kmGrids.py`, which also processes the boundary cells of this wave model grid with functions from `PySMCs/smcellmap.py` to generate WW3 model required boundary condition files. More details for the program will be skipped here, interested users please see the Python program code and WW3 model guide.

4. Generate face arrays.

SMC grid face arrays are generated with a FORTRAN program in `SMCGTools/F90SMC/`. The program `SMCGSideMP.f90` could be run in OpenMP mode with 2 threads or run as a single processor program without OpenMP. In this example, the program is compiled with OpenMP on a Linux machine:

```
SMCGTools/F90SMC/ ifort -r8 -qopenmp SMCGSideMP.f90
SMCGTools/F90SMC/ mv a.out SMCGSideMP
```

You need to work out the FORTRAN compiler on your system and create the executable with correct OpenMP option. The executable is renamed as `SMCGSideMP` for later use.

Before running the program to generate your face arrays, a few parameters are needed for input to the program. An example input file is given in `Linuxs/SideMPInput.txt`, which contains the following lines:

```
SMC61250
 260000   300000   4   ##  NCL,  NFC,  MRL
   4096     3072   0   ##  NLon, NLat, NPol
'../DatGMC/SMC61250Cels.dat'
 .FALSE.
```

The first line `'SMC61250'` is the grid name, and the second line specifies the cell number `NCL`, face number `NFC` and number of resolution levels `MRL`. The cell and face numbers should be larger than the actual cell and face numbers for memory allocation purpose. The cell number could be derived by counting the cell array file. The exact face number is not known yet so here an estimated face number about 10% more than the cell number is used. If the program failed because the face number was smaller than the exact face number, you could increase the `NFC` value and run the program again.

The 3rd line specifies a rectangular domain in size-1 cells. Here SMC61250 is a global grid, it covers 4096 size-1 cells in the longitude direction and 3072 size-1 cells in the latitude direction. The NLon number will be used to define the global periodic condition, which links up the first cell on one parallel line with the last one, if any. The 3rd number on the 3rd line is the number of polar cells in the grid. For this SMC61250 grid global part, there is no polar cell. The polar cell is in the Arctic part cell array and its face arrays will be generated later.

A Linux run script is provided in `SMCGTools/Linuxs/runSMCSideMP` to run the face generating program. You need to modify the paths to your directories before you run it. In Linux system, issuing the command line:

```
SMCGTools/Linuxs/ ./runSMCSideMP SideMPInput.txt
```

which will generate the face arrays for the global part in the SMC61250 grid. The face array files are saved in `SMCGTools/tmpfls/SMC61250[IJ]Sid.dat` and they need to be moved out of this temporary directory to `DatGMC/`.

```
SMCGTools/tmpfls/ mv SMC61250ISid.dat SMC61250JSid.dat ../DatGMC/
```

Face arrays for the Arctic part are created with a similar job script and input files (`Linuxs/runSMCSideMPArc` and `SideMPInputArc.txt`). Modify the paths inside the script before you run it. The Arctic part face arrays are saved as `SMC61250A[IJ]Sd.dat` in `tmpfls/` and they need to be moved into `DatGMC/` as well.

```
SMCGTools/Linuxs/ ./runSMCSideMP SideInputArc.txt
SMCGTools/tmpfls/ mv SMC61250AISid.dat ../DatGMC/SMC61250AISd.dat
SMCGTools/tmpfls/ mv SMC61250AJSid.dat ../DatGMC/SMC61250AJSd.dat
```

Face arrays for regional models could be generated as for the global part of the SMC61250 grid ones with corresponding input files for the regional grid. For instance, the face arrays for the AMM153 grid could be generated with the following Linux command line:

```
SMCGTools/Linuxs/ ./runSMCSideMP SideInputAMM153.txt
```

5. Run propagation model to test the cell and face arrays

There are two propagation test models available in `SMCGTools/F90SMC/PropOMP` and `PropHyb/`. The OpenMP version `PropOMP` model stores wave spectra in one memory block for all spatial points for shared memory machines. The `PropHyb` test is for distributed memory machines in hybrid mode (MPI-OpenMP). It distributes spatial sea points among MPI ranks for wave spectral storage as in WW3 and calculates spatial propagation in spectral component parallelization mode.

The `PropOMP` model can be run on multi-core desktop machines with a Fortran compiler. To speed up the model, a single frequency bin can be used with a few directional bins (24 or 36). Total wave height is output at selected time intervals as text files and visualized with a Python program. The following steps demonstrate how the model is run and results are checked:

i) Copy the whole subdirectory `SMCGTools/F90SMC/PropOMP/` to your desk space and modify the `Makefile` for your local Fortran compiler and OpenMP option.

ii) Compile the Fortran `PropOMP` model in Linux system with the make command.

iii) Change to sub-directory `SMCGTools/Linuxs/` and modify the `PropInput.txt` file for your model grid. The sample file is for the SMC61250 grid and parameters are explained by the comments following the # marks.

iv) Modify the Linux script `runSMCGProp` for your directory paths and run it.

```
SMCGTools/Linuxs/ ./runSMCGProp PropInput.txt
```

v) Move the output files to a storage directory (`OutDat/` for instance) and select files to be visualised into `cfiles.txt`, such as for selecting all files

```
   ls C* > cfiles.txt
```

Note that the first line in `cfiles.txt` will be skipped so you can put anything on the first line (some comments for instance). The above Linux command line puts the `CMesgs.txt` file on the first line.

vi) Modify the last line in `PySMCs/GridInfo61250.txt` if your propagation test results are not stored in `../OutDat/` and then run the `SMCGTools/PySMCs/SMCGloblProp.py` to draw the SWH plots for your selected files:

```
SMCGTools/PySMCs/ python SMCGloblProp.py GridInfo61250.txt
```

The plots will be saved in `../tmpfls/` in eps format. You may view them directly with Linux `gv` or convert them into another picture format, gif format for instance with `Linuxs/conveps2gif`.

The global propagation test demonstrates an initialised wave field propagating over the SMC61250 grid. Wave will travel as far as it reaches coastlines. If any premature energy loss appears, it may indicate something wrong with your cell or face arrays.

Obstruction ratios for the SMC61250 grid is generated with `SMC61250Obstr.py`, which uses the obstruction ratios saved in the bathymetry file `Bathy088_059deg.nc`. This obstruction ratio file is not used in the propagation test, but it will be required for the WW3 wave model propagation scheme.

6. Prepare input files for WW3 model to use your SMC grid.

Apart from the cell and face arrays for your SMC grid, WW3 also requires a few input files for its model configuration. A full list of WW3 input files for the SMC61250 grid is available on our Cray EX machine.

```
WW3Vn7/inp650/ ls -l
-rw-r--r-- 1 mo_users    21634 Oct 17  2018 SMC61250AISd.dat
-rw-r--r-- 1 mo_users    27826 Oct 17  2018 SMC61250AJSd.dat
-rw-r--r-- 1 mo_users    11920 May  4  2021 SMC61250BArc.dat
-rw-r--r-- 1 mo_users  6669105 May  4  2021 SMC61250Cels.dat
-rw-r--r-- 1 mo_users 14231888 May 17 14:11 SMC61250GISd.dat
-rw-r--r-- 1 mo_users 15123437 May 17 14:11 SMC61250GJSd.dat
-rw-r--r-- 1 mo_users  1190920 May 17 14:36 SMC61250Obst.dat
-rw-r--r-- 1 mo_users 62733319 Jul 27 16:28 mod_def.ww3
-rw-r--r-- 1 mo_users    28235 May 17 14:57 ww3_grid.inp
-rw-r--r-- 1 mo_users     3572 Oct 30  2018 ww3_ounf.inp.template
-rw-r--r-- 1 mo_users     2799 May 21 16:17 ww3_outf.inp.template
-rw-r--r-- 1 mo_users     7177 Sep 19  2016 ww3_outp.inp.template
-rw-r--r-- 1 mo_users    18425 May 21 16:17 ww3_shel.inp.template
-rw-r--r-- 1 mo_users     3401 Dec 20  2012 ww3_strt.inp
```

SMC grid also uses the sub-grid partial blocking ratios for unsolved small islands. The sub-grid obstruction ratios are generated with `SMCGTools/PySMCs/SMC61250Obstr.py`, which uses pre-calculated obstruction ratios on the same latitude-longitude grid at about 6 km as the bathymetry data (`SMCGTools/Bathys/Glob6kmObstr.dat`) and converts them onto the SMC grid cells. Note that only cells in the global part of the SMC61250 grid are assigned obstruction ratios. The Arctic part will assume all obstruction ratios to be zeros as it is in the central Arctic Ocean around the N Pole. This also makes it convenient to drop the Arctic part (for cold seasons) without modifying the obstruction ratio file.

The `SMC61250Obstr.py` also calls a Python function `global50km.py` to work out the regular 50 km grid parameters. As SMC grid shares some output settings with a regular grid at its base resolution, these parameters are required for WW3 input files. These parameters are saved in a text file `S61250Regul.txt`, in addition to the obstruction ratio file `SMC61250Obst.dat`.

A Python program `SMC61250RInfo.py` is also provided here to generate regular grid information for the SMC61250 grid. Users may modify the parameters inside to tuning your regular grid coverage area, instead of using the one fixed by `global50km.py`.

Instructions for preparing other WW3 input files and run the WW3 wave model are available in the WW3 user guide and will not repeat here.

7. Visualise SWH output from WW3 model on SMC grid.

The WW3 output file `out_grd.ww3` could be converted into sea point only SWH at the pre-set output interval in ASCII file if you choose the type 4 option in the `ww3_outf.inp` file for postprocessing with `ww3_outf`.

These text SWH output files are named as `ww3.yymmddhh.hs` and can be visualised with the Python program line

```
SMCGTools/PySMCs/ python SMCGloblSWHt.py GridInfo61250.txt
```

which read the `GridInfo61250.txt` input file for grid related information and the path for the WW3 model output files in `../../S61250Tx/`. You need to modify the input file if your WW3 output files are not stored there. Another input file the program reads is `strendat` in the same directory for selection of plotting times:

```
SMCGTools/PySMCs/ cat strendat
23090106  23093018  12h
```

The above `strendat` tells `SMCGloblSWHt.py` to draw one month from `(20)23090106` hr to `(20)23093018` hr at `12` hr interval. The SWH plots are saved as eps file in `SMCGTools/tmpfls/`. You may view them with `gv` or convert them into gif just as you view the grid plot. In fact, the `SMCGloblSWHts.py` uses the exact grid projection polygons of the grid plot to draw the SWH plots.

Part II: SMC sub-grids for multi-grid run

1. Splitting a global SMC61250 grid into 3 sub-grids

The `SMCGTools/PySMCs/Sub61250Splt3.py` program is used to split the SMC61250 global grid into 3 sub-grids. It uses three prescribed dividing lines to split the global grid and the dividing lines are defined by lists of latitude-longitude pairs, which are stored in the file `Bathys/Subgrdline3.dat`. You may modify these dividing lines for a different sub-grid configuration. The 3 sub-grids cell arrays are saved in `tmpfls/` along with their boundary cells. You need to modify the program for your paths before you run it. This path update applies to all subsequently mentioned programs and this reminder will not be repeated.

Use the `Linuxs/runSub3CountCell` script to sort the cell arrays into final form and move the `tmpfls/*61250*.dat` files into `DatSub/`.

Note that the Arctic part in the SMC61250 grid will be appended to the Atln61250 sub-grid and its cell and face arrays in `DatGMC/` can be used directly with the sub-grid cell and face arrays.

The sub-grids can be visualised with `SMCGTools/PySMCs/Sub61250Grid3.py` by selecting individual sub-grid or a full global view of all sub-grids together. The individual sub-grid option will also generate their boundary cell id list for use in the WW3 model run. A list of the boundary cell lon-lat positions are also saved for a global model to generate boundary conditions for each sub-grid if it need to run as an independent model.

A Python program `smcelsplit.py` is also provided here for a more general splitting job. Advanced users may adapt the program for your use. Details will not be explained here.

Once you see the sub-grid plots, you may find that some cells are isolated after the splitting and other cells may be better moved into its neighbouring sub-grid. These flaws are the results of the rough splitting lines, which must be single-valued functions of latitudes. If you could find a more flexible splitting method, these flaws may be removed. For the time being, some manual tuning with Linux awk editor is used to adjust the sub-grid cell array. This is the same method used to tuning boundary cells, which will be discussed later.

2. Generate face arrays and obstruction ratios for sub-grids

Use the `Linuxs/runSub3SideMP` to generate face arrays for the 3 sub-grids. Face array files are saved in `tmpfls/*61250[IJ]Sid.dat` and they need to be moved into `DatSub/` for later use.

Obstruction ratios for the 3 sub-grids are generated with `PySMCs/Sub61250bstr3.py` and saved in `tmpfls/*61250Obst.dat`. Move these files into `DatSub/` as well.

3. Other input files required for WW3 multi-grid run

Apart from the sub-grid cell and face arrays, WW3 multi-grid model also requires some extra input files. An example list of input files required for WW3 model multi-grid run on the Sub61250 sub-grids is listed here:

```
-rw-r--r-- 1    2460 Oct  7 10:52 Atln61250Blst.dat
-rw-r--r-- 1 1752832 Oct  7 15:01 Atln61250Cels.dat
-rw-r--r-- 1 3942279 Oct  7 15:15 Atln61250ISid.dat
-rw-r--r-- 1 4211380 Oct  7 15:15 Atln61250JSid.dat
-rw-r--r-- 1  313015 Oct  7 15:59 Atln61250Obst.dat
-rw-r--r-- 1    9520 Oct  7 10:55 Pacf61250Blst.dat
-rw-r--r-- 1 2965850 Oct  7 15:01 Pacf61250Cels.dat
-rw-r--r-- 1 6229376 Oct  7 15:14 Pacf61250ISid.dat
-rw-r--r-- 1 6608227 Oct  7 15:14 Pacf61250JSid.dat
-rw-r--r-- 1  529625 Oct  7 15:59 Pacf61250Obst.dat
-rw-r--r-- 1   21639 Oct  6 16:44 SMC61250AISd.dat
-rw-r--r—1     26855 Oct  6 16:44 SMC61250AJSd.dat
-rw-r--r—1     11921 Oct  6 10:40 SMC61250BArc.dat
-rw-r--r-- 1   12480 Oct  7 10:58 Soth61250Blst.dat
-rw-r--r-- 1 2018944 Oct  7 15:01 Soth61250Cels.dat
-rw-r--r-- 1 4226706 Oct  7 15:13 Soth61250ISid.dat
-rw-r--r-- 1 4478350 Oct  7 15:13 Soth61250JSid.dat
-rw-r--r-- 1  360535 Oct  7 15:59 Soth61250Obst.dat
-rw-r--r-- 1   28251 Oct 11 11:14 ww3_grid_Atln.inp
-rw-r--r-- 1   28071 Oct 11 11:24 ww3_grid_Pacf.inp
-rw-r--r-- 1   28071 Oct 11 11:28 ww3_grid_Soth.inp
-rw-r--r-- 1    5064 Sep 13 11:04 ww3_multi.inp.template
-rw-r--r-- 1    3610 May 11  2021 ww3_ounf.inp.template
-rw-r--r-- 1    2799 Nov  2  2018 ww3_outf.inp.template
-rw-r--r-- 1    7177 Jan 27  2021 ww3_outp.inp.tempAtln
-rw-r--r-- 1    7177 Oct  6  2020 ww3_outp.inp.tempPacf
-rw-r--r-- 1    7177 Feb 16  2021 ww3_outp.inp.tempSoth
-rw-r--r-- 1   17861 May 21 16:24 ww3_shel.inp.tempAtln
-rw-r--r-- 1   16550 May 21 16:25 ww3_shel.inp.tempPacf
-rw-r--r-- 1   16630 May 21 16:26 ww3_shel.inp.tempSoth
-rw-r--r-- 1    3401 Dec 20  2012 ww3_strt.inp
```

Instructions for generating those extra input files and running the multi-grid model are available in the WW3 user guide.

4. Postprocessing of multi-grid model output and visualisation of SWH field

Visualization of the multi-grid SWH field is like the single SMC grid case except for that there are choices to draw individual sub-grid field or merge them together like a single global output. The individual sub-grid output files `out_grd.[Atln/Pacf/Soth]` are converted into ASCII files with the postprocessing program `ww3_outf` type 4 conversion. The ASCII SWH output are transferred to our desktop machine for visualisation. They can be drawn with the Python program `SMCGTools/PySMCs/Sub61250SWHts.py,` which reads the same input file `PySMCs/strendat` for selection of model times as for single grid SWH plots.

5. Tunning sub-grid boundary cells

The sub-grid boundary cells generated by the splitting program `Sub61250Splt3.py` may not be perfect. For instance, some boundary cells near coastlines are isolated by land and they would not affect the sub-grid boundary conditions. These redundant boundary cells may be deleted from the boundary cell list, reducing the model communication load. On the other hand, some small islands within boundary zones may reduce the effective width of the boundary line and the boundary line is better to be widened around the small islands. These fine tunings of the sub-grids and their boundary cells are not automatized so far and must be done manually on cell-by-cell bases. To aid for the viewing of boundary cells, a few programs are developed for selecting some cells for a specified area. The following steps illustrate how these tools are used.

i) First specify your interested area by the latitude-longitude pairs for the SW and NE corners of the area and save them in a text file: `Slatlons.dat`. These lat-lon pairs may be retrieved from Google Map by clicking on these points. You may simply work them out from a world atlas.

ii) Run the python script, `PySMCs/latlon2ijs.py`, to convert you lat-lon pairs into corresponding cell index values:

```
SMCGTools/PySMCs/ python latlon2ijs.py Slatlons.dat 8
 Input argvs = ['Slatlons.dat', '8']
 Read grid info from  Gridinfo.dat
 Input file zlon zlat dlon dlat =
 [0.  0.  0.08789062 0.05859375]
 Read lat-lon pairs from  Slatlons.dat
[[ 33.6  -7.5]
 [ 37.4  -4. ]
 [-16.  115.5]
 [ -1.  138.6]]
 Total number of lat-lon pairs = 4
 SMC grid zlon, zlat, dlon, dlat = [0.         0.
0.08789062 0.05859375]
 i, j indexes will be rounded to nearest multiple of k = 8
  352.500   33.600   4008    576
  356.000   37.400   4048    640
  115.500  -16.000   1312   -272
  138.600   -1.000   1576    -16
 xlon ylat and converted i j's are saved in Slatlons.dat_ijs
Cells in area of the first two lines could be selected with
   awk -f awktemp Cell_file > tempcels.dat
```

This program gets the SMC grid `zlon, zlat, dlon, dlat` values from the default file: `Gridinfo.dat` and converts your `(lat, lon)` pairs into corresponding `(i, j)` indexes to the nears multiple of size-8 cells. The converted results are also saved in the file, `Slatlons.dat_ijs`. The sample area specified by the first two pairs are for the Gibraltar Strait area and an awk script file, `awktemp`, is produced for selection cells in the area.

```
SMCGTools/PySMCs/ cat awktemp
## awk script to select cells for area specified by
## SW corner:   352.500   33.600   4008    576
## NE corner:   356.000   37.400   4048    640
## Usage:  awk -f awktemp  Cell_file > tempcels.dat

 { if ((( $1 >= 4008 && $1 < 4048 ) &&
        ( $2 >= 576 && $2 < 640 ))) print $0 }
```

iii) Run the Linux awk command to select cells in your specified area:

```
SMCGTools/PySMCs/ awk -f awktemp ../DatSub/Atln61250Cels.dat \
> tempcels.dat
```

Here the sub-grid Atln61250 is chosen for illustration.

iv) The selected cells can be viewed with

```
SMCGTools/PySMCs/ python smcviewrap.py
 Input argvs = []
 Read cells from  tempcels.dat
 Total number of cells = 197
 Cel i, j range = 4006 4047 574 642
```

```
          Figure size = 4.1 6.8
          Read table from  rgbspectrum.dat
         136  colors with depth marks at  [1 10 100 1000 10000]
```

The png format diagram is used for viewing the cells within the selected area and may help make the decision whether any cell needs to be edited. Removing a boundary cell can be done by simply locating the cells in the boundary cell list file and deleting the cell line.

The awk command script file, `awktemp`, may be modified for selecting cells in a particular area specified by the `i` and `j` ranges as $1 and $2. You may delete these cells by simply reversing (adding logical not ! at the beginning of) the awk script condition, such as

```
          { if (!(( $1 >= 4008 && $1 < 4048 ) &&
                  ( $2 >= 576 && $2 < 640 ))) print $0 }
```

Another linux script, removecell, is also available for removing a list of cells from a given cell list. For instance, if you like to remove all the cells listed in `tempcels.dat` from the original cell list file, `../DatSub/Atln61250Cels.dat`, you may simply issue the command line:

```
   SMCGTools/PySMCs/ cat tempcels.dat | ../Linuxs/removecell \
                           ../DatSub/Atln61250Cels.dat
```

A new `templist` file will be created and it should be the same as `Atln61250Cels.dat` except that those cells listed `tempcels.dat` have been removed. To confirm the removal is done properly, you may compare the new list with the original one.

Further automatization of trimming cells could be achieved by saving those awktemp scripts with different names and do all the trimming in one go with the script:
`SMCGTools/Linuxs/trimtempcels`

For example, the following line combines all the listed awk* scripts to trim the given input file:

```
ls awkCaspian awkBaikal awkVictoria | trimtempcels \
   ../DatSub/Atln61250Cels.dat
```

This script may be used later to repeat the above trimming if the input cell array is updated.

6. Mixed resolution multi-grid option

The SMC multi-grid option treats each sub-grid as an independent 'regional' model so different resolution SMC sub-grids may be used in one multi-grid configuration. For instance, a refined European area at 3-6-12-25 km resolutions may be used with the 6-12-25-50 km sub-grids in the rest areas, if their boundary conditions are set up properly. As the present SMC multi-grid option uses a simplified 1-1 spectral swapping boundary condition, it would be ideal if boundary cells are identical cells in their donor sub-grids. These can be done by adding duplicated cells in donor sub-grids. If the donor sub-grid has different resolution levels from the receiving grid, the boundary cell location indexes and size numbers must be adjusted before adding to the donor grid cell array. The Python program `smcupalevel.py`, for instance could change the cell array up a resolution level by simply doubling the first 4 indexes of each cell in the given cell array file:

```
  python smcupalevel.py tempcels.dat
```

which will write the modified cell array in the file `tempcels.dat_New.`

Other tools for generating mixed resolution sub-grids are still in development stage and the guide will be updated when they are added to the package.

Part III: Shallow water equations model on SMC grids

1. Introduction

A shallow water equation model has been developed on SMC grids for possible simulations of storm surge and tsunami waves. More details of the SWEs model are available in another guide document, `SWEsonSMC_Guide.pdf`. The main purpose of this part is for description of Linux scripts to run the model and Python programs to visualize the model grid and results. These scripts and programs are also stored in the corresponding sub-directories by their language type as the SMC grid ones, such as the FORTRAN90 source code of the SWEs model is stored in `SMCGTools/F90SMC/SWESMCoMP/`, along side of the SMC grid face array generating programs and the propagation test models. Linux job scripts are stored in `Linuxs/` and Python programs in `PySMCs/`.

The SWEs model is firstly tested on some full global grids in widely used classical tests. Here only the 3-level SMC1° R3 grid will be used as an example to demonstrate how to set up input files and run the SWEs model. The SMC1° R3 grid is generated with the Python program `PySMCs/smcellSWE.py` and the default input file `GridInfoSMC1R3.txt`.

```
 SMC1R3 3 level SWE grid zlon zlat dlon dlat for siz-1 cells.
  0.0  0.0  0.0  0.3515625  0.25
  ../tmpfls/   ../DatSWE/
  2  Number of polar cells or polar parts. 0 implies no Arctic part.
```

which defines the size-1 cell in the grid and NLevel to be 3 in main(). The cell arrays are stored in `SMCGTools/DatSWE/` for SWEs model files.

Face arrays for the SMC1° R3 grid are generated with the command line:

```
SMCGTools/Linuxs/ ./runSMCSideMP SideInputSMC1dR3[Arc].txt
```

where the input files for the global part and Arctic part face arrays are

```
SMCGTools/Linuxs/ cat SideInputSMC1dR3.txt
SMC1R3
 60000   65000   3   ##  NCL,  NFC,  MRL
  1024     720   0   ##  NLon, NLat, NPol
'../DatSWE/SMC1R3Cels.dat'

SMCGTools/Linuxs/ cat SideInputSMC1dR3Arc.txt
SMC1R3A
  1000   1200   3   ##  NCL,  NFC,  MRL
  1024    720   2   ##  NLon, NLat, NPol
'../DatSWE/SMC1R3BArc.dat'
```

Note the NPol is used to indicate where polar cells are in the cell array. The 2 polar cells are all in the Arctic part cell array DatSWE/SMC1R3BArc.dat, where the Arctic part name is used though the full global grid also contains the Antarctic polar region.

The SMC1° R3 grid could be viewed with the aid of the `PySMCs/SMC1R3grids.py` program and there are 5 optional views for different SWEs model tests, including a full global view of the grid, and a single hemisphere view of the refined area and the northern polar region. These two view options are used to draw the SWEs model W2 test result. The W2 test started with a steady zonal flow as the second test suggested by Williamson et al (1992) and is expected the model could keep it steady as close to the initial state. This test will be used here to demonstrate how the SWEs model on the SMC 1° R3 grid is run and how the model results are visualised.

2. Compiling and running the SWEs model

The SWEs model can be compiled with the make command with default `Makefile` stored in `SMCGTools/F90SMC/SWESMCoMP/`. Users need to modify the `Makefile` for their own FORTRAN compiler name and options if they are different from those in the file. If the compilation is successful, it will generate an executable named as `SWEsnSMCeP` in the same directory.

The model has built in some test cases with their initial subroutines. These test cases can be selected with a case number in the input file, along with other model grid information. The default input file `SMCGTools/Linuxs/SWEsInput.txt` is for the W2 test:

```
SMCGTools/Linuxs/ cat SWEsInput.txt
SMC1R3
 '../DatSWE/'
 68000  69000   3  58235 # NCL,  NFC,  MRL,  Itrm
 1024    720    2    2   # NLon, NLat, NPol, Init
 9600    240   20   40   # NTS,  NWP,  NHrg, NLrg
 0.0   0.0  0.3515625  0.2500000   # ZLon, ZLat, DLon, DLat
 1.0  90.0  3.5E5      0.0000      # DFR0, DT,   AHK,  CBFr
 -180.0  2.864789  73.0   0.4      # PLon, PLat, PAvr, Beta
 .True. .False. .False. .False.    # Arctic, Source, WHPrts, Restrt
```

In which the first line is the model grid name and the second line is the grid cell and face array directory. The first two numbers of the third line specify the memory dimensions to store the cell and face arrays. They need to be larger than the actual cell number and the maximum cell arrays among u- and v-faces. The next number 3 is the number of resolution levels of the SMC1R3 grid, and the final number is a selected cell number for equation terms output. The fourth line contains the number of size-1 cells along longitude and latitude, number of polar cells in the grid and the selected test case number (2 is for the W2 test, for other tests, please read the source code `F90SMC/SWESMCoMP/SWEsnSMCPM.f90`).

The fifth line consists of the number of time steps of the model run (9600 steps), output interval in number of time steps (write out at every 240 steps), the polar and global average interval for the momentum equations. The sixth line are the cell array reference point and size-1 cell increments in longitude and latitude degrees. The 1st value on the seventh line is the initial diffusivity ratio for a ramping increase of the diffusivity to its maximum value AKH, which is given by the 3rd value on the same line. The DFR0 = 1.0 implies that the diffusivity started at the maximum value of AKH, or there is no ramping to reduce the initial diffusivity. The second value (90.0) on the seventh line is the time step in second and the final value CBFr is the coefficient of the bottom fraction. For the W2 test the bottom fraction is switched off by setting CBFr = 0.0.

The first two values on the eighth line in the `SWEsInput.txt` file specifie the W2 test rotated pole position in lon-lat degrees for the zonal flow. The rotation pole of the zonal flow is almost vertical to the geographical pole in the W2 test. The 3rd parameter PAvr = 73.0 is the latitude to define high latitude averaging zone and the final value Beta = 0.4 defines the polar biased diffusion for the water mass equation. The final line defines a few logical values. The SMC1R3 grid includes two polar regions so Arctic is set to be true. The other ones are for tsunami modelling and set to be false for the W2 test.

The SWEs model W2 test is started by the command line:

```
SMCGTools/Linuxs/ ./runSWEsOMP [SWEsInput.txt]
```
where the input file name is required if its name is not the default `SWEsInput.txt`.

3. Visualization of SWEs model test results

The W2 test results are saved into `SMCGTools/OutSWE/` and could be viewed with the Python program `SMCGTools/PySMCs/SMC1R3wdpth.py`, which require an input file named `OutSWE/cnfilesall.txt` to specify which model output files to be drawn. The comparison figure (Fig.3) used in the `SWEsonSMC_Guide.pdf` could be generated with `PySMCs/SMC1R3W2dif.py` program and the input file `OutSWE/cnfiles04d.txt`.