Spherical Multiple-Cell (SMC) grid utility tools

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

26 April 2023

Part I: Single SMC grid including the Arctic

1. Generate a global SMC61250 grid.

Starting with the SMCGTools/PySMCs/SMC61250Cells.py Python program to generate a global SMC61250 grid at 6-12-25-50 resolutions.

You need to copy the whole `SMCGTools/*` to your disk space and modify the paths inside the Python program `SMC61250Cells.py` before you run it. A possible improvement is to set up a working path to `SMCGTools/` and pass it into the Python program.

The `SMC61250Cells.py` program will read a bathymetry file in ASCII format at 6 km global resolution from `SMCGTools/Bathys/Glob6kmBathy.dat` and call for a few Python functions stored in the sub-directory `SMCGTools/PySMCs/`. This may be changed by setting up a Python path so you may run the program outside the `SMCGTools/PySMCs/` sub-directory.

Cell array file will be saved in a temporary sub-directory `SMCGTools/tmpfls/`, including an Arctic part cell array file as this is a full global grid.

One Python program `smcellgen.py` called by the `SMC61250Cells.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. Details of its usage will not be discussed here.

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

Move the newly generated cell arrays from `tmpfls/` into `DatGMC/` and run the grid plotting Python program, such as on a Linux system:

```
SMCGTools/PySMCs/ python SMC61250Grids.py
```

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

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

```
SMCGTools/tmpfls/ ls smc61250grdGlob.ps | ../Linuxs/convps2gif90r
```

The above global view plot is in landscape format while regional plot may be in portrait format and could be converted into gif format with Linuxs/convps2gif.

If you like a different regional view other than the two provided ones, you may modify one of the regional view option inside the program SMC61250Grids.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]
        papror='landscape'
```

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 similar to the orthogonal projection, the maximum spanning angle is 90 degree to cover one hemisphere. If your spanning angle is less than 90 degree, 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 ps plot x/y-sizes in inches and the last two parameters specify where your text messages are printed in plotting units.
The last parameter papror='landscape' tells the program to draw the ps plot in landscape orientation. This is for printing convenience. You may use portrait orientation for all plots if you like.

3. 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 `SideMPInput.txt`, which contains the following lines:

```
'<GridName>'
 260000   300000   4   ##  NCL,  NFC,  MRL
   4096     3072   0   ##  NLon, NLat, NPol
'../DatGMC/<GridName>Cels.dat'
 .FALSE.
```

The `<GridName>` will be replaced with the grid name later when the file is used. The second line specify 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 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.

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
```

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/`.

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.

4. 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.

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 SMCGTools/PySMCs/SMC61250Props.py for your directory paths and draw the SWH plots for your selected files. The plots will be saved in SMCGTools/tmpfls/ in ps format. You may view them directly with Linux gv or convert them into gif with Linuxs/convps2gif(90r) for xv viewer.

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.

5. 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 X40 machine.

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

6. 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 `SMCGTools/PySMCs/SMC61250SWHts.py`, which read the input file `strendat` in the same directory for selection of plotting times:

```
SMCGTools/PySMCs/ cat strendat
16090106  16093018  12H
```

The above `strendat` tells `SMC61250SWHts.py` to draw one month from `(20)16090106` hr to `(20)16093018` hr at `12` hr interval.

The `SMC61250SWHts.py` needs to be modified for your storage paths and SWH plots are saved as ps file in `SMCGTools/tmpfls/`. You may view them with `gv` or convert them into gif for `xv` viewer just as your view the grid plot. In fact, the `SMC61250SWHts.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.

Appendix I. 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 firstly reduced to the required size-1 cell resolution. For the following example grid, the size-1 cell length will be set at $\Delta\lambda$ = 0.0439453125° and $\Delta\varphi$ = 1/30° or 0.0333333333°. The longitude resolution is chosen so that the total number of points along one longitude parallel circle is 8192 = 2**13, 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 5400 size-1 cells. This latitude cell length allows each full degree latitude falls on a size-1 cell face 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 10 is chosen for the longitude direction and it reduces the 15 second resolution to 2.5 minute with 8640 points in one parallel circle. For the latitude direction a factor of 8 is used and it reduces the 15 second resolution to 2 minute or 5400 points along the latitude range. The temporary data are then interpolated to the required size-1 resolution in the longitude direction (from 8640 to 8192 points). There is no need for interpolation in latitude direction as it is already at the required size-1 resolution. If the first 5120 (= 5*2**10) points from the NP are selected, it could reach about 80.66S, which is good enough for ocean surface coverage.

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_10_8.nc ../Bathys/Bathy042_033.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/Bathy044_033deg.nc
```

The final bathymetry file `Bathys/Bathy044_033deg.nc` will be used for generating the SMC371250 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.

Appendix II. Generate and trimming SMC371250 grid

A sample 4-level (3-7-13-26-52 km) grid, SMC371250, is generated with the Python program, `SMC371250Cells.py`, which uses the bathymetry file, `Bathy044_033deg.nc`, created in Appendix I. 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.

The Python program, `Lakes325Cells.py`, created the 4-level (3-7-13-26 km) Caspian Sea and Great Lakes cells with the same bathymetry file but different water surface heights for different lakes. The individual lake cells are trimmed with awk scripts (stored in `Linuxs/zTrim350/`) to remove isolated cells or overlapping areas. First to remove the count/header line from each individual lake file, the use the Linux command line, taking the Ontario Lake cell array for example:

```
Linuxs/zTrim350/ awk -f awkOnta350 Onta37125Cels.dat > Onta37125Cels.d
```

The trimmed Caspian Sea cells are saved in `Casp37125Cels.d` and the trimmed Great Lakes are merged into a single file `GtLk37125Cels.d`. The global SMC371250 cell array, `SMC371250Cels.dat`, is also edited to remove its count/header line and renamed `SMC371250Cels.d`. Then it is trimmed and merged with the individual lakes by the following Linux command lines:

```
Linuxs/zTrim350/ ls awkAmazn350 awkAustr350 awkCaspN350 awkGtLks350 \
                    awkMedtr350 awkTurBkl350 | ../trimtempcels      \
                    ./SMC371250Cels.d

Linuxs/zTrim350/ cat GtLk37125Cels.d Casp37125Cels.d >> tempcels.dat
Linuxs/zTrim350/ ../countcell6lv tempcels.dat
Linuxs/zTrim350/ mv New_tempcels.dat ../../DatGMC/SMC371250Cels.dat
```

The first line trims the, `SMC371250Cels.d`, which is created by the Python program, `SMC371250Cells.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/`.

The trimmed SMC371250 grid cells could be viewed with the Python program, `SMC371250Grids.py`, and its face arrays are generated with

```
SMCGTools/Linuxs/ ./runSMCSideMP SideInput371250.txt

SMCGTools/tmpfls/ mv SMC371250ISid.dat SMC371250JSid.dat ../DatGMC/

SMCGTools/Linuxs/ ./runSMCSideMP SideInput371250Arc.txt
SMCGTools/tmpfls/ mv SMC371250AISid.dat ../DatGMC/SMC371250AISd.dat
SMCGTools/tmpfls/ mv SMC371250AJSid.dat ../DatGMC/SMC371250AJSd.dat
```

Propagation test is done with an input file and the command line:

```
SMCGTools/Linuxs/ ./runSMCGProp PropInput371250.txt
```

The results are moved into `OutDat/` and viewed with `SMC371250Props.py`.