U. S. Department of Commerce
National Oceanic and Atmospheric Administration
National Weather Service
National Centers for Environmental Prediction
5200 Auth Road Room 207
Camp Springs, MD 20746

Technical Note

Automated grid generation for WAVEWATCH III[†].

Arun Chawla [‡] and Hendrik L. Tolman
SAIC-GSO at
Environmental Modeling Center
Marine Modeling and Analysis Branch

April 2007

[†] MMAB Contribution No. 254.
[‡] e-mail: Arun.Chawla@NOAA.GOV

This page is intentionally left blank.

Abstract

An automated grid generation package has been developed for WAVEWATCH III in MATLAB. The development of this package was motivated by the need for accurate objective obstruction grids representing unresolved islands. Previously, such grids have been created manually at great expense of man hours. The inputs to this package are a high resolution bathymetry and a separate high resolution coastline data set. The output from this package are the bathymetric grid, optional land−sea mask information (either as a separate file or incorporated in the bathymetric data) and the obstruction grids along the 2 main grid axes. Validation studies were carried out for 3 different regions − the Caribbean, Hawaii and the French Polynesian islands. In all the cases, grids were generated at 5 different resolutions (2',4',8',15' and 30' resolutions), and a constant swell was applied along the Northern and Eastern boundaries to create steady state solutions at the different resolutions. Though there were some differences between the lower and higher resolution simulations, most of these arose from the propagation of energy through narrow gaps between islands (which cannot be resolved by the lower resolution grids). Generally the systematic patterns of energy attenuation due to sub−grid features were well resolved by the obstruction grids in all test cases.

Acknowledgments. The authors thank Dr. Carlos Lozano and Dr. Robert Grumbine for their input in developing the algorithms and Dr. Avichal Mehra for comments on early drafts of this manuscript. This report is available as a pdf file from

http://polar.ncep.noaa.gov/waves

Contents

This page is intentionally left blank.

1 Introduction

An automated grid generation software package has been developed for use with the wind wave model WAVEWATCH III (Tolman, 2002b). The aim of this software is to develop accurate obstruction grids for sub−grid modeling (Tolman, 2003) as well as provide a powerful tool that will reduce the man hours necessary to generate grids for the model.

At its core the software uses two types of global data sets − one is a high resolution 2' global bathymetry (currently the choice is between using **ETOPO2**; NGDC 2006 and **DBDB2 v3.0**; NRL 2006) and the second is a high resolution shoreline database (**GSHHS** − **G**lobal **S**elf - consistent **H**ierarchical **H**igh - resolution **S**horeline[1]; Wessel and Smith 1996). The **GSHHS** database is stored as a collection of polygons. We decided to use it to determine our coastal boundaries instead of relying only on the high resolution bathymetry to provide this information, because the boundary data set conveniently resolves the small islands, jetties and other structures that are not easily resolved. Furthermore, the coastline database provides the starting point to develop obstructions for unresolved or poorly resolved features. In full resolution this database consists of 188606 boundaries of which 180,509 are coastal boundaries[2]. Out of these 180446 boundaries have an area less than 12400 km^2 (corresponding to a grid square of $1^\circ \times 1^\circ$ at the equator). Fig 1.1 shows a zoomed image of the cumulative distribution of these boundaries as a function of their areas (in km^2). 99 % of these boundaries have an area less than 6 km^2. In comparison the reference global 2' grid has a resolution of $\approx 14km^2$. Thus, most of these coastal features can only be represented as sub-grid obstructions and accounting for them manually is a prohibitive task. Having an automated grid generation software allows us to account for all of these coastal features fully and objectively. Furthermore, building multiple grids at different resolutions to take full advantage of the nesting capabilities of WAVEWATCH III becomes a fairly trivial task.

In this report there shall be repeated references to "grid cells". A grid cell is defined as the area of the domain associated with a grid point (or node). All the nodes have cells associated with them and the cell boundaries for any particular node are halfway between the node and its neighbors.

The different modules of our grid generation software are

1. A **grid generation module** → develops the first guess low resolution grid from the high resolution base bathymetry.

2. A **boundary module** → extracts all the boundaries from the global boundary database that lie inside the design grid, properly accounting for boundaries being split by the grid domain.

[1] available from the National Geophysical Data Center

[2] The rest are lakes, islands in lakes and ponds in islands in lakes

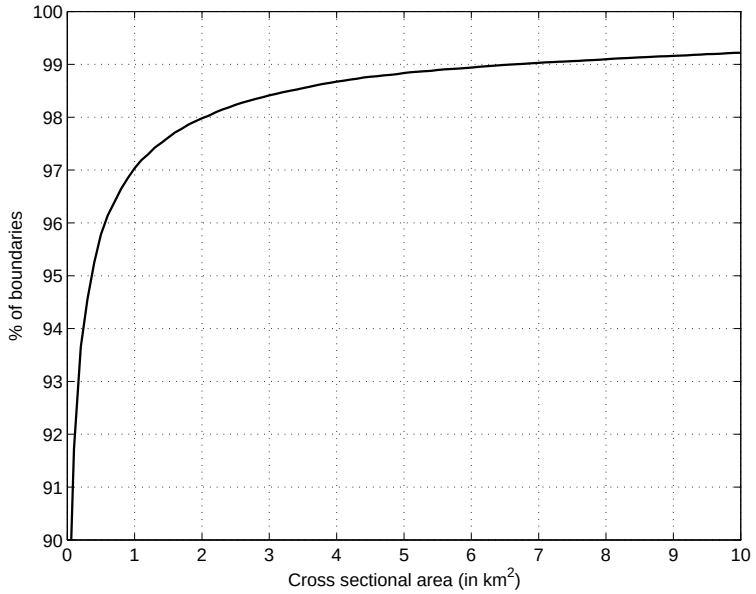


Fig. 1.1 : Cumulative distribution of coastal boundaries with area less than 12,400 km^2 (Number of boundaries = 180446)

3. A **land mask module** → generates the land−sea mask, separating the cells into 'wet' (or active computational) and 'dry' (or inactive computational) cells, with the help of the bathymetry and boundary data[3].

4. A **wet cell module** → groups all the wet cells that are connected together. Thus, separating the wet cells into different groups separated by the dry cells. This allows smaller water bodies to be masked out (turned inactive).

5. A **sub−grid module** → builds the obstruction grids for unresolved boundaries in the sub−grid domain.

6. A **mask modification module**[4] → modifies the final land− sea mask to separate the nodes into active, inactive and boundary nodes.

Fig 1.2 shows the flow chart of the package. The filled ellipses refer to data files (either reference database files that are needed as input to run the code or output files that can be used as input for WAVEWATCH III). The filled dashed circles refer to data variables, and the rectangles refer to the different modules. The package has been developed in MATLAB and consists of a series of function calls (one for each module) that can be called from a master script[5]. This provides the users with the flexibility to choose which parts of the package they wish to use.

[3]An optional module to split up large boundaries is available to speed up computations

[4]This module is only used for WAVEWATCH III version 3.10 or higher

[5]An example master script will be distributed with the package

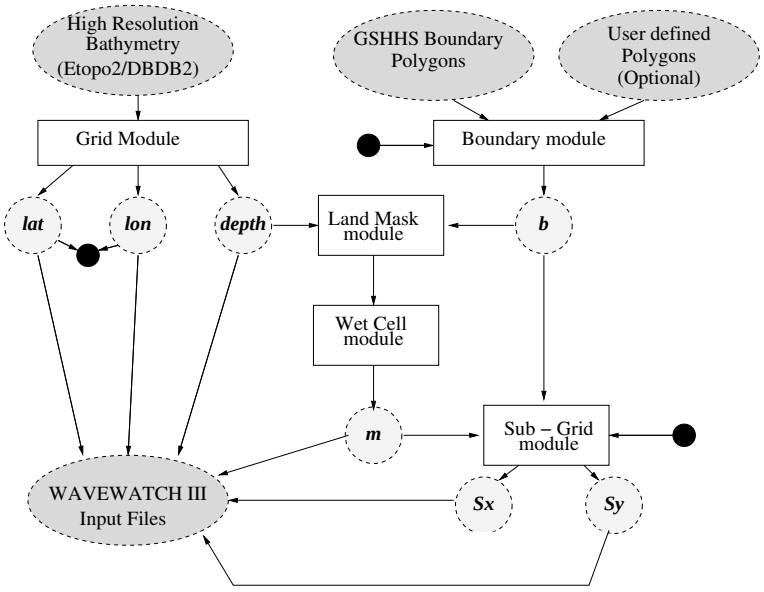


Fig. 1.2 : Flow Chart of the grid generation software (solid circle denotes the grid coordinate information that is obtained from the Grid module and passed to the Boundary and Sub−grid modules.)

The code uses several reference data sets. Two of these are high resolution global bathymetry data sets (in netcdf formats), the third is a collection of global high resolution shoreline polygons (in MATLAB binary formats), and the fourth optional data set is a collection of polygons (also in MATLAB binary format) used to "mask out" selected bodies of water. Using the reference data and grid information provided by the user, the **grid generation module** develops a design grid and stores information in the ***lon*** (x coordinate), ***lat*** (y coordinate) and ***depth*** (bottom topography in x and y coordinates; negative below the datum) variables. The ***lat*** and ***lon*** variables are used to determine the grid domain and this information is used by the **boundary module** to extract a subset of boundary polygons that comprise all the coastal features within the design grid domain (shown by variable ***b*** in Fig 1.2). The boundary polygons combined with the depth information are then used by the **land mask module** to develop an initial set of masks (stored in the 2D array ***m***) separating the wet computational cells from the dry cells. This initial mask information is then used by the **wet cell module** to group the wet cells into different water bodies and provide the user with the option of masking out cells corresponding to water bodies that do not play a role in the computation (e.g. lakes). Finally the mask and coastal boundary information are used by the **sub−grid module** to develop a set of obstruction grids in x and y directions (***Sx*** and ***Sy*** variables respectively in Fig 1.2) for coastal features that cannot be resolved by the grid. Bathymetric and obstruction grid input files are then created for WAVEWATCH III (in one of the

accepted formats) to be used for defining model domains.

The package also uses specific programs to read netcdf files in MATLAB (since the reference bathymetry is stored in netcdf format). These codes will be distributed with the package. However, if the user wishes to use an independent bathymetry data set, it does not preclude him/her from using this package as long as they can either write their own program to generate the initial grid variables (***lon***, ***lat*** and ***depth***), or create their reference grid in the same netcdf format as the provided reference bathymetry data. The codes assume a regularly spaced 2D grid, though spacing in the two directions can be independent of each other.

2 MATLAB modules

Detailed information on the algorithms of the different modules together with the arguments with which these functions are called in MATLAB are provided in this section.

2.1 Grid generation module

The grid generation module builds a design grid from a high resolution global bathymetric data set. Users can choose from either the **ETOPO2** (from NGDC) or the **DBDB2** grids (from NRL). Both grids have a 2' resolution in latitude and longitude and Chawla (2007) assess the quality of the two grids in select regions for WAVEWATCH III applications. Here these grids are collectively referred to as the reference grid.

Direct interpolation of bathymetry data from the higher resolution reference grid to the grid points of the lower resolution design grid is not advisable because of the possibility of aliasing. To avoid this, filtering needs to be done to remove features that cannot be resolved in the design grid. Here we use 2D averaging to remove fluctuations smaller than one grid cell.

The averaging routine (**generate_grid_av**) algorithm is outlined in Fig 2.1. It determines the number of wet cells in the high resolution reference grid that lie within a single cell of the design grid. If the ratio of the total wet area (from the high resolution grid cells) to the total area exceeds a user specified cut off limit (ranging between 0 and 1), then the cell is considered wet and its depth is denoted by the average depth of all the wet cells. Otherwise, the cell is considered dry. A cut off limit is used to prevent a design grid cell from becoming wet even if there is only a single wet cell within its domain in the reference grid. This method has the advantage of maintaining a reasonable shoreline even if the design grid is much coarser than the reference grid. For higher values of the cut off limit the shoreline would be further offshore and vice versa. If the user wishes to take advantage of the coastal database to represent the shoreline then the cut off limit should be set to a very low value so that the shoreline from the bathymetry data is further inshore and then use the **land mask module** described in section 2.3 to get a more accurate shoreline representation. The syntax for the averaging routine in MATLAB is

```
>> [lon,lat,depth] = generate_grid_av(ref_dir,ref_grid,grid_box,...
                                      dx, dy, icoords,cut_off);
```

Where we use the angle brackets to indicate that the function is called in the MATLAB environment (this convention to denote a MATLAB environment will be followed through the report) and the three dots indicate a continuation on the next line. The input variables are

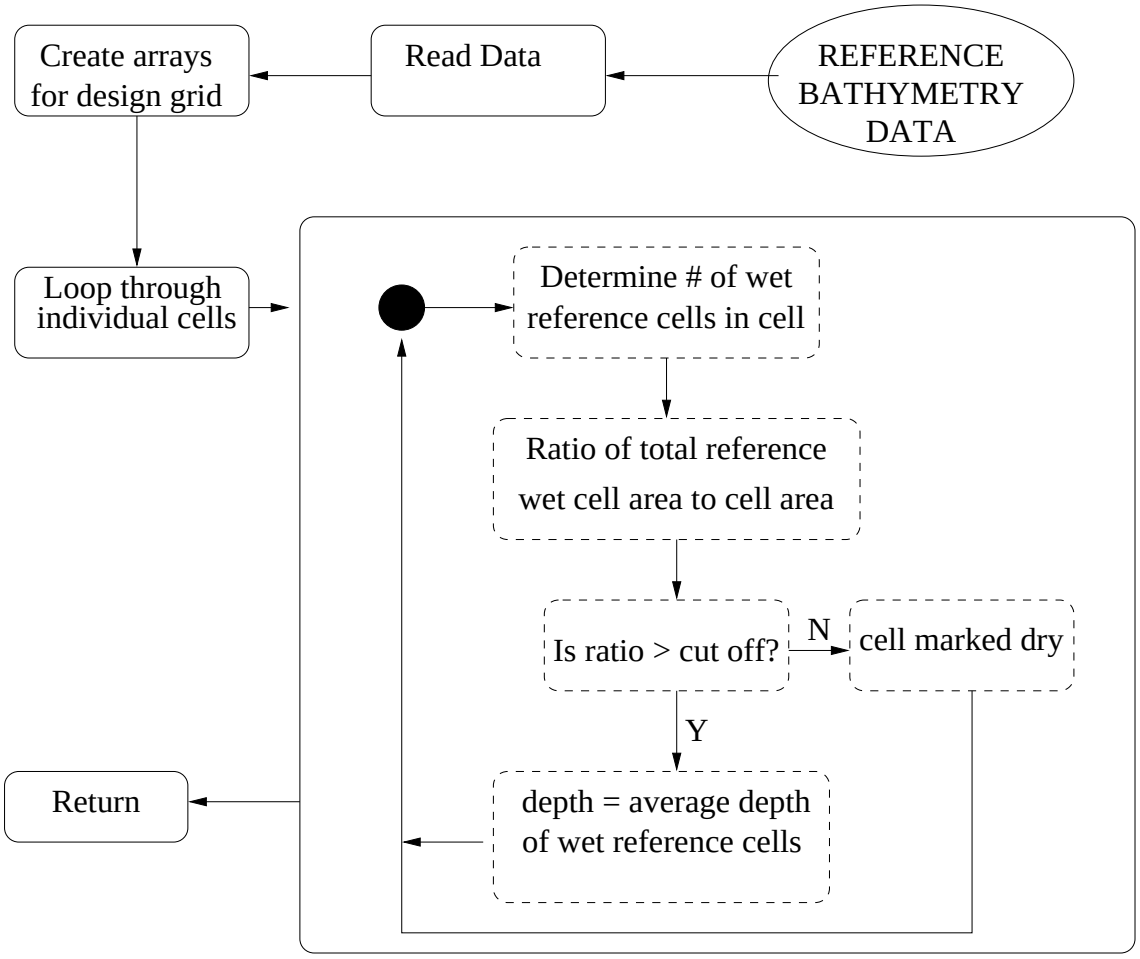


Fig. 2.1 : Flow chart of the grid module algorithm

- ***ref_dir*** → The reference directory location
- ***ref_grid*** → The choice of reference bathymetry (options are 'dbdb2' and 'etopo2')
- ***grid_box*** → A 4 element array providing the latitude (y)and longitude (x) coordinates of the lower left hand and upper right hand corners of the design grid respectively
- ***dx, dy*** → Grid resolution (in degrees) for x and y directions respectively
- ***icoords*** → Representation of longitude coordinates (options are 0 for longitudes ranging from $-180°$ to $180°$ and 1 for values ranging from $0°$ to $360°$)
- ***cut_off*** → Value ranging from 0 to 1 that provides the criteria for making a design grid cell 'wet' or 'dry'

and the output variables are

- ***lon*** → The 1D array along the x direction with length N_x
- ***lat*** → The 1D array along the y direction with length N_y
- ***depth*** → The 2D averaged bathymetric depth of size N_y by N_x[6].

Fig 2.2 shows cross-sectional profiles of the bathymetry in the Bahamas region as a function of the longitude at 5 different latitude sections. For comparison purposes, the bathymetry that would result from direct sampling of the reference bathymetry at grid points have also been plotted. At some locations this bathymetry will match the representative bathymetry, but at other locations it will lead to erroneous features (such as the deep trench along the $18°N$ transect.).

2.2 Boundary module

Coastal boundary information plays a crucial role in grid development, both for generating accurate land−sea masks and obstruction grids. We use a global database of shorelines (**GSHHS** − **G**lobal **S**elf - consistent **H**ierarchical **H**igh - resolution **S**horeline) that provide detailed land-water interface boundaries for the entire world. The shoreline data are stored as a collection of polygons and each polygon represents a closed boundary. The points that make up a polygon are ordered in a counter clockwise orientation. The boundaries represented in the database are coastlines, lakes, islands in lakes and ponds in islands in lakes.

[6]The way MATLAB defines 2D arrays is a transpose of the way FORTRAN defines 2D arrays

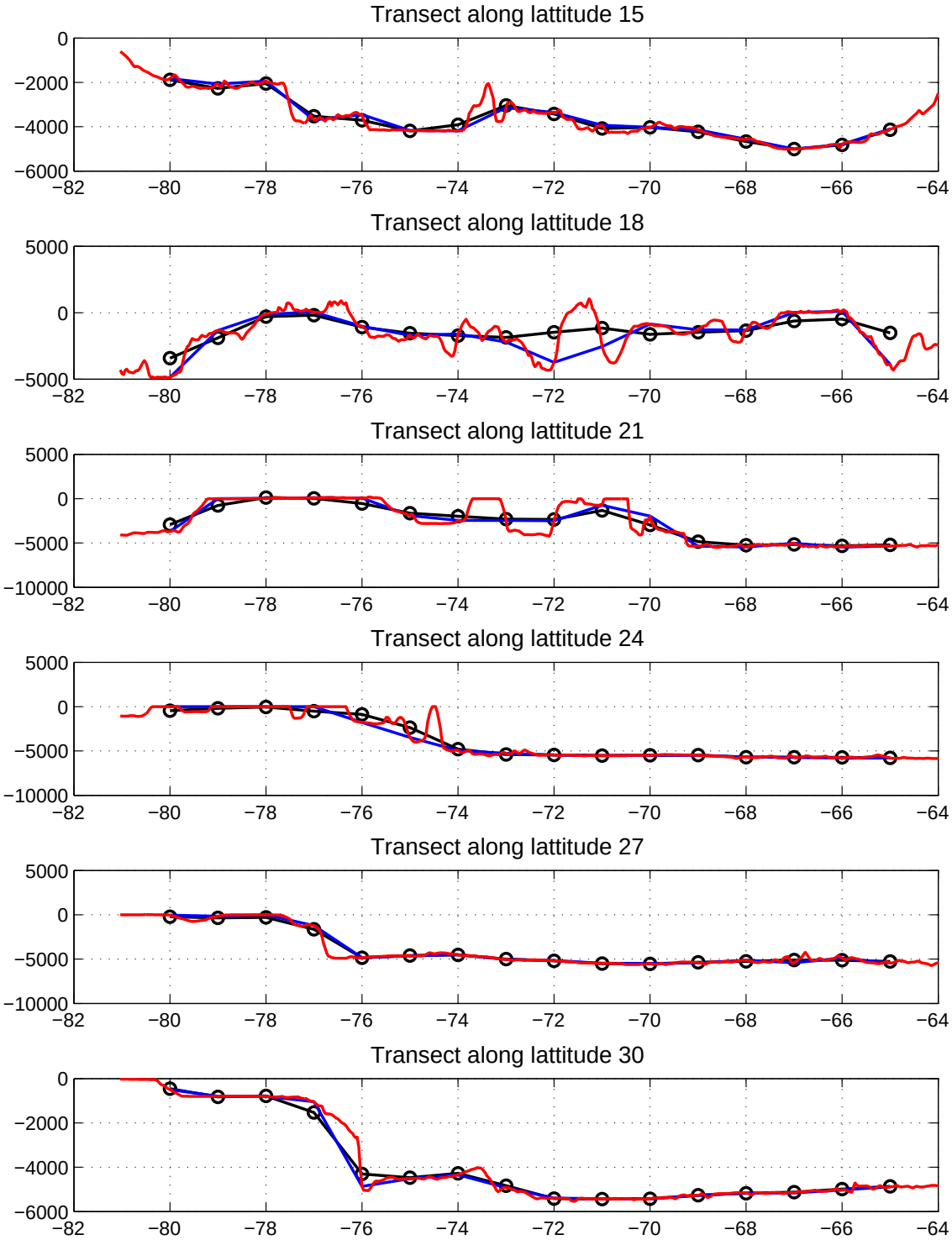


Fig. 2.2 : Bathymetry profile as a function of longitudes across several latitude sections in the Bahamas (negative values refer to the Western Hemisphere). Circles refer to the grid locations (resolution 1°). Black line: depth computed from the averaging module. Red line: depth in the reference grid (here **DBDB2***). Blue line: depth sampled from the reference bathymetry at the grid points only. (Depth below the Mean Sea Level in m)*

For our purposes we are primarily interested in the coastal boundaries though for certain applications (such as Great Lakes modeling) users may be interested in other types of boundaries as well. The global shoreline data sets are available at 5 different resolutions

- full resolution with 188606 separate polygons for the global set
- high resolution (0.2 km resolution) with 153539 separate polygons for the global set
- intermediate resolution (1km resolution) with 41523 separate polygons for the global set
- low resolution (5 km resolution) with 10769 separate polygons for the global set
- coarse resolution (25 km resolution) with 1866 separate polygons for the global set

For convenience the data sets at different resolutions have been stored as arrays of data structures in '.mat' file formats (a MATLAB binary format that can be easily loaded in MATLAB). There are separate '.mat' files for boundaries at different resolutions.

Using polygons to represent boundaries has distinct advantages. First, the boundary data are available at a much higher resolution and do not need to be generated from the reference grid. Second, it is easy to determine the extent of the boundary in the two horizontal directions from the polygon information. Third, since these polygons are closed we can use available functions to determine if points lie inside, outside or on the polygon (MATLAB has an intrinsic function to do this). Finally, we can create additional polygons either for any boundaries that are not represented in the global data sets (such as local dikes, breakwaters etc.) or to mask out water bodies that do not play a role in the simulations (eg masking out the Mediterranean Sea for a swell propagation study in the Atlantic Ocean).

The boundary module uses the global data set to obtain a sub set of boundaries that lie within the design grid domain. Accounting for boundaries that are completely inside the grid domain is trivial. However, boundaries that intersect the grid domain need to be split up and closed appropriately so that they can be used to determine land-sea masks as well as obstruction grids. A typical grid domain will intersect several boundaries as illustrated in Fig 2.3. In the figure the boundary labeled 1 lies inside the grid domain and is included as such, but all the other boundaries need to be manipulated for efficient use.

A simple algorithm which just ignores boundary points that lie outside the grid domain and adds additional points for where the boundaries intersect the

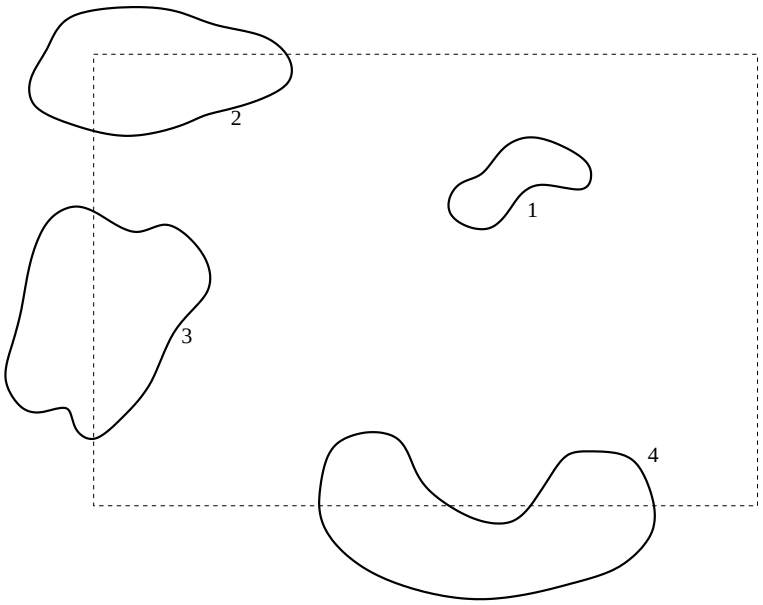


Fig. 2.3 : A typical sketch of coastal boundaries intersecting a grid domain (defined by the dashed rectangular box)

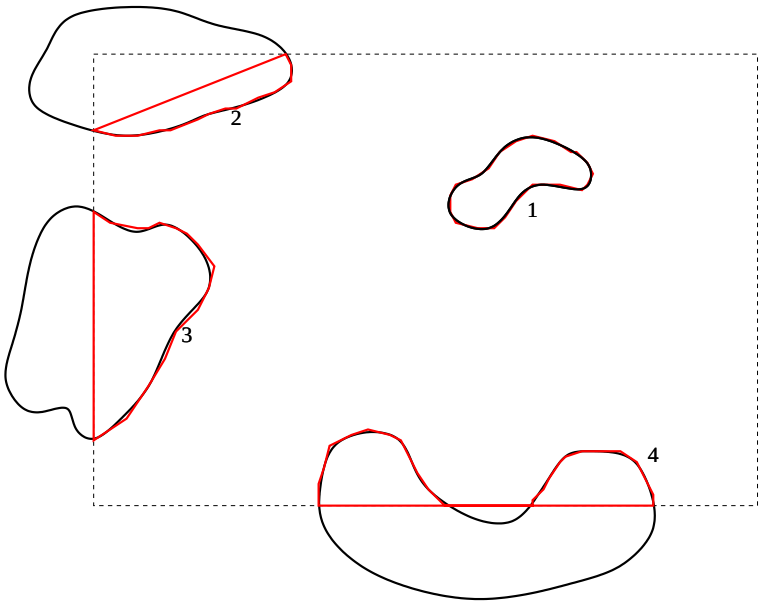


Fig. 2.4 : Coastal boundaries (in black) and the modified boundaries (in red) that would result if the points lying outside the grid domain in Fig 2.3 are ignored

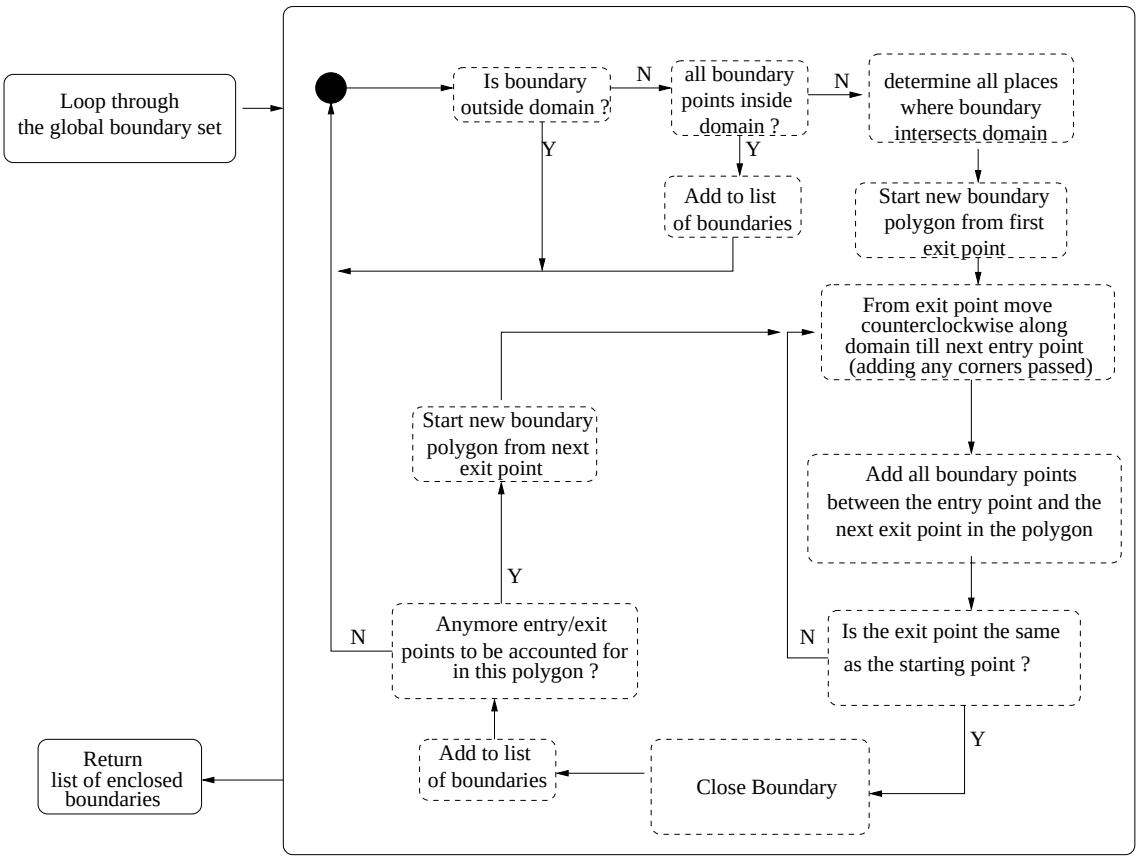


Fig. 2.5 : Flow chart of the coastal boundary module

grid domain will only work for boundary 3 (see Fig 2.4). For boundary 2 we need to account for the grid corner that is enclosed within the grid domain, and boundary 4 needs to be split up into two unconnected separate boundaries. An algorithm to account for this and other cases is outlined below (see Fig 2.5 for the accompanying flow chart). We start by initializing a list that will include all the boundary polygons that lie within the grid domain and loop through the global set of boundary polygons. Each boundary polygon is then handled in the following steps

Step 1: Determine if a boundary polygon lies inside or outside the grid domain. Polygons that are completely inside (added to the list) or completely outside (discarded) are easily handled. For the rest, compute the points of intersection where the polygon exits and enters the grid domain[7].

Step 2: Start a new polygon set from a point of exit and move along the grid domain in a counter - clockwise direction (adding all corners that you come across to the polygon) until the next entry point is encountered. Ignore all the points in the original boundary polygon that lie between these two points as they lie outside the domain[8].

Step 3: Add all the points (in the original polygon) that lie between the entry point to the next exit point to the polygon set (since these lie inside the domain).

Step 4: Check the exit point. If the exit point is the same that we started from then the polygon has been closed and we add it to our list of boundary polygons. At this point there are two possibilities − one, that there are no more entry / exit points in the original polygon that have not been accounted for in which case we move on to the next boundary, or alternatively, there are still entry / exit points in the original polygon that are not accounted for, in which case we start a new polygon set from the next unaccounted exit point and repeat from step 2 for all the unaccounted points (effectively splitting a boundary into smaller polygon sets).

Step 5: If on the other hand the exit point in step 4 is not the same as the starting point then that means that the polygon cannot be closed. In which case we add the exit point to the polygon set and continue as in step 2.

Step 6: Once all the entry / exit points in the original polygon are accounted for we move on to the next boundary polygon.

[7]For each point of exit there will be a corresponding point of entry

[8]If the points of intersection between the grid domain and the polygon are not part of the polygon set of points then these points are added to the new polygon set

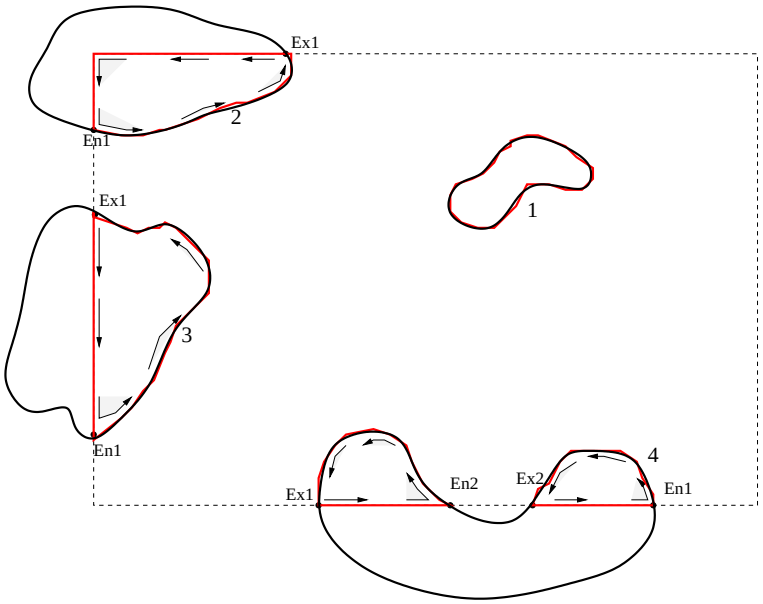


Fig. 2.6 : Coastal boundaries obtained by appropriate boundary algorithm

We now revisit the coastal boundaries shown in Fig 2.3 and apply the above algorithm. Fig 2.6 shows the corresponding boundary sets that are generated with the arrows indicating the path followed in developing each boundary. The exit and entry points (from the grid domain) for each boundary are labeled by the prefixes Ex and En respectively. In boundaries 2 and 3 there are only one set of exit and entry points while for boundary 4 there are two such sets. Starting with the first exit point for boundary 2 (Ex1) we move in a counter clockwise direction along the grid domain, and adding the corner point crossed (as outlined in step 2) until we reach the entry point (En1). Adding all the internal points between En1 and the next exit point, which in this case happens to be Ex1, we find that the boundary has been closed as per step 4, and since all the exit / entry points have been accounted for, we move on to the next boundary. The algorithm works the same way for boundary 3, except that this time the boundary closes without crossing any corners. Boundary 4 is a little different as now there are two sets of exit / entry points. Starting from the first exit point for this boundary (Ex1) we move in the counter clockwise direction till the entry point En2 is reached. Again following step 4 we find that the boundary is closed, but at this point we are still left with one set of exit / entry points unaccounted for and thus a new boundary segment is created starting from exit point Ex2 and following steps 2 to 4. Thus, boundary 4 is split up into two smaller boundaries.

Before the function call to this module can be made the global boundary data have to be loaded into the MATLAB environment. Depending upon the resolution that the user wishes to work with this will be one of 5 different '.mat' files,

and can be easily loaded in MATLAB. The syntax for the boundary extraction module is

```
>> b = compute_boundary(coord,bound);
```

Where the input variables are

- ***coord*** → A 4 element array providing the latitude and longitude coordinates of the lower left hand and upper right hand corners of the design grid[9].
- ***bound*** → A data structure array containing the global boundary data which is created when the coastal boundary mat file is loaded. (The global boundary data sets are always stored in an array called ***bound***). Each element of the data structure consists of eight different variables listed below
  - ***bound.n*** → The number of boundary points in the polygon
  - ***bound.level*** → The type of boundary data (1 = coastal, 2 = fresh water lake, 3 = island in fresh water lake, 4 = pond in island in fresh water lake)
  - ***bound.west*** → The western extent of the boundary polygon
  - ***bound.east*** → The eastern extent of the boundary polygon
  - ***bound.north*** → The northern extent of the boundary polygon
  - ***bound.south*** → The southern extent of the boundary polygon
  - ***bound.x*** → An array of length ***n*** containing the x (***lon***) values of the boundary points
  - ***bound.y*** → An array of length ***n*** containing the y (***lat***) values of the boundary points

and the output variable is another data structure array ***b*** that consists of all the boundary polygons that lie within the grid domain. The data structure format for ***b*** is slightly different than the one for ***bound***. The variable ***level*** is no longer used. Since the boundary algorithm filters out all boundaries other than the coastal boundary all boundaries in ***b*** are level 1[10]. Instead we have two other variables that are used in the sub-grid algorithm later for building obstruction grids

[9]This array is constructed from the starting and ending values of ***lon*** and ***lat*** in section 2.1 and may be different from the values of array ***grid_box***

[10]If the user wishes to use the software to create grids for a lake application such as swell propagation in the Great Lakes then a slight modification will have to be made to this routine where only polygons that represent islands in lakes are considered. This is fairly straight forward to do

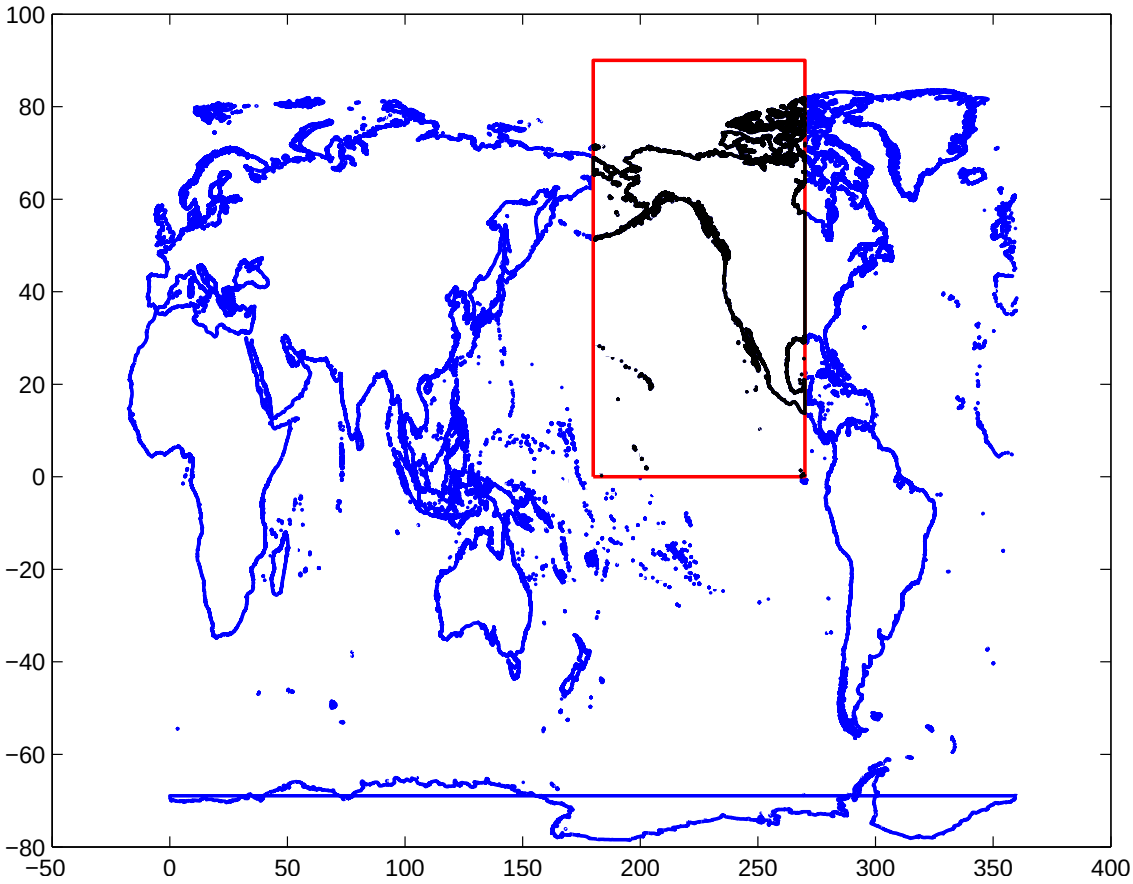


Fig. 2.7 : Full resolution coastal boundaries for the global domain in blue with a subset of coastal boundaries (obtained using the boundary algorithm) in black. The red box identifies the grid domain used to extract the boundaries.

- ***b.height*** → Height of the boundary in the vertical direction (North - South)
- ***b.width*** → Width of the boundary in the horizontal direction (East - West)

Fig 2.7 shows a plot of the full resolution coastal boundaries together with the boundary data that lies within the domain of interest. The domain was cut through the North American continent to test the ability of the module to appropriately split up the boundaries. From the figure we can see that the boundary module works well.

Two potential problem areas are the polygons that make up the Eurasian and Antarctic coast lines. These polygons wrap around because they extend around the globe (to avoid wrap around for the Eurasian coast line while plotting the longitudes in Fig 2.7 were extended from -50 to 400). This is an issue only for creating global grids because then the polygons that wrap around do not close

uniquely which is necessary for the remaining modules in sections 2.3 to 2.5. To avoid this, global grids are created by developing grids for the two halves of the hemispheres separately and then splicing them together [11].

2.3 Land mask module

The land mask module generates the land-sea mask. A first cut to the land-sea mask is determined from a cut-off depth in the grid. All depths above the cut-off depth are marked land (or dry) and the rest are marked sea (or wet) cells. This however, may lead to cells that are marked wet but lie inside boundary polygons either due to interpolation from reference grid or because the reference grid itself does not resolve the land features represented by the polygons. This is particularly true for some of the smaller islands in the Bahamas as well as the island chains of French Polynesia. Furthermore, using polygons provides a mechanism for masking out wet cells not needed in the computation (see discussion on user-defined polygons in section 2.2).

The land mask module steps through each of the polygon boundaries, determines which of the wet cells are enclosed within the boundary and turns them into dry cells. The algorithm does not try to do the reverse (make dry cells outside boundaries into wet cells) as the wet cells also need a representative depth for computation. The algorithm proceeds as follows (see Fig 2.8 for the flow chart).

Step 1: Determine the cells that bound a particular polygon by using the boundary limits in the x and y directions

Step 2: Then determine all the cells that are marked wet, within this range and loop through them (this avoids having to loop through large sets of cells that are well inside a polygon).

Step 3: For each wet cell distribute a set of equally spaced points (≈ 25) over the entire cell and determine which of these points lie inside the polygon

Step 4: Compute the ratio of the number of points inside the polygon to the total number of points. This provides an approximate estimate of what portion of the cell lies inside the boundary. Add this to any existing value for the cell (so that we can account for multiple boundaries in a particular cell)

Step 5: For each cell if the total ratio exceeds a user specified cut-off value (between 0 and 1) then the wet cell is switched to dry. This approach is used so that a cell is switched to dry only if a significant portion of its area is inside a boundary.

The syntax for the call to the function is

[11] An example master script describing this process will be distributed with the package

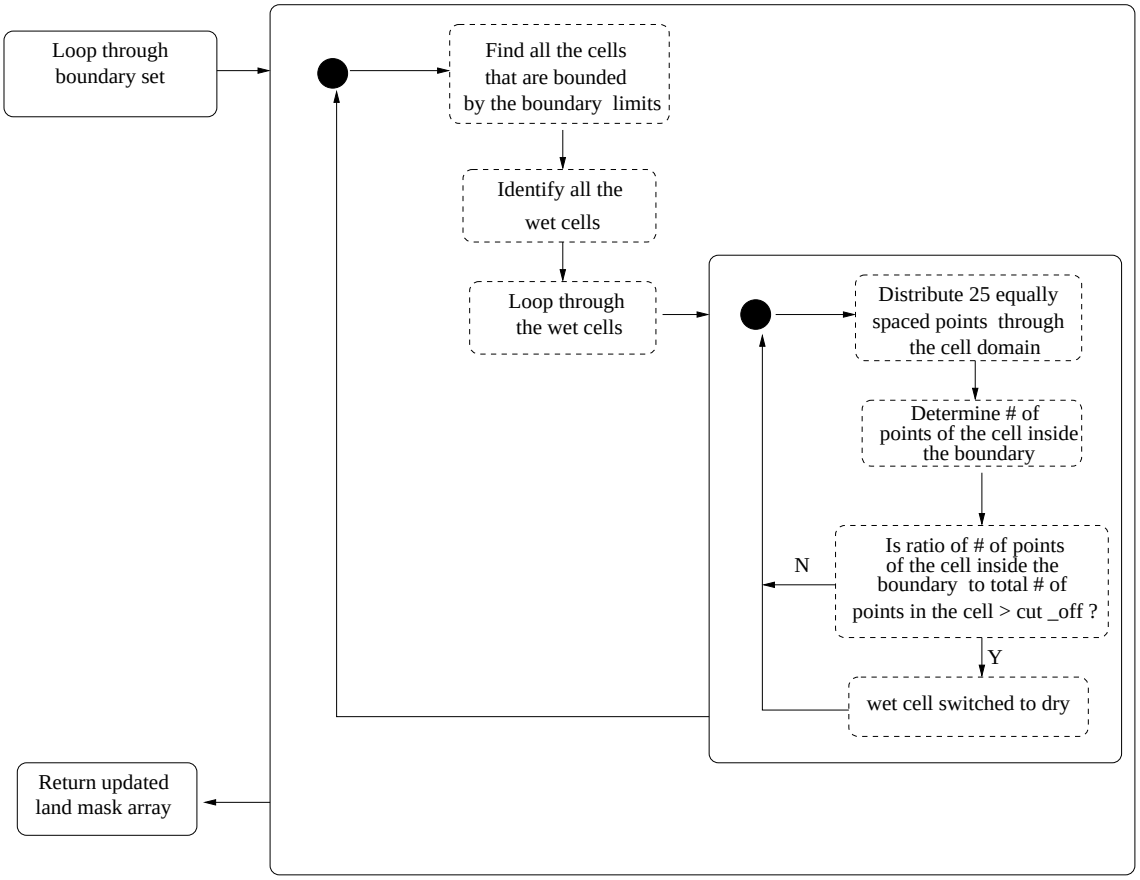


Fig. 2.8 : Flow chart of the land mask module

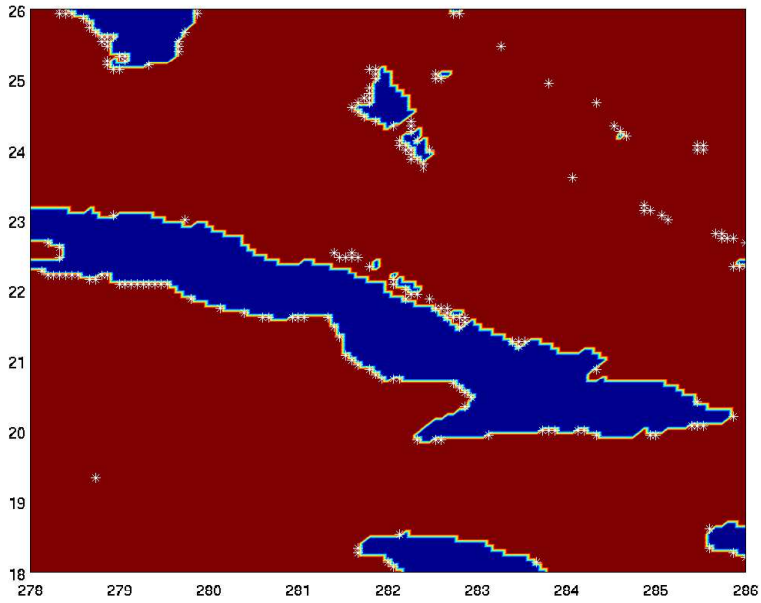


*Fig. 2.9 : Land-sea mask for the Caribbean domain. Red region specifies the wet cells, blue region refers to dry cells determined from a bathymetric cut off depth of 0 m, and white stars are cells that were originally wet but were turned to dry because they were inside boundary polygons (**cut_off** set to 0.5)*

```
>> m2 = clean_mask(lon,lat,icoords,m,b,cut_off);
```

Where the input variables are

- ***lon,lat*** → 1D arrays representing the x and y coordinates
- ***icoords*** → Flag for orientation of the longitudes (see section 2.1).
- ***m*** → A 2D input land mask array of the same dimensions as ***depth*** with a value of 0 for land and 1 for water. This array is generated from a depth cut-off criterion as a first approximation to the land mask.
- ***b*** → The boundary polygon data structure array from the **compute_bound** function in section 2.2.
- ***cut_off*** → A value ranging between 0 and 1 which determines how much portion of a cell has to be enclosed within the boundary to be declared dry

and the output variable is ***m2***, a cleaned up version of the land mask array that accounts for cells that were originally marked wet in ***m*** but are switched to dry because they are enclosed inside polygons.

Fig 2.9 shows an example of the use of this algorithm for a region in the Caribbean. The grid was generated from the 2' **ETOPO2** reference grid at a resolution of 4'. The white stars in the figure denote the additional wet cells that are switched to dry because they lie inside boundary polygons. This routine accounts for cells that are close to the coastline which were marked as wet cells because of grid resolutions but should be dry because they lie within a land boundary, as well as coastal features that may not be well resolved in the reference grids but are well resolved as coastal boundary polygons.

Having the list of boundary polygons as an argument to the function provides considerable flexibility while developing the land-sea mask. The set of boundary polygons ***b*** can also contain a list of user-defined polygons which will automatically switch all the wet cells that the user wishes to mask out to dry cells. This provides a convenient way to mask out cells where energy will not propagate (e.g. Lagoons, reefs, harbors etc.). Alternatively, for some cases it may not be desirable to switch cells enclosed within boundaries to dry cells (e.g. inundation studies) in which case those polygons can be dropped from the list that is passed to the function.

2.3.1 Split boundary module

When the number of points defining a boundary polygon becomes significantly large the subroutine to determine points that lie inside/outside the polygon becomes computationally very intensive, making the land mask module very slow[12]. To avoid this problem, a splitting module is used that splits up the larger boundary polygons into smaller polygons.

The command line arguments for this module are

```
>> b_sp = split_boundary(b,tol);
```

where the input variables are

- ***b*** → The boundary polygon data structure array from the **compute_bound** function in section 2.2.
- ***tol*** → A tolerance limit. If the width and/or height of a boundary polygon exceeds this limit then the polygon is split up into a number of segments.

and the output variable ***b_sp*** is the split boundary data structure array that is then provided as input to the land mask module.

Though using this module is not necessary for obtaining the land mask, it is advisable to do so, specially when using the full resolution polygon data set. Rule of thumb is to set the tolerance limit to be at least 4 or 5 times the grid resolution with the minimum value being at least 2°. This prevents too much

[12]This is an issue with most high resolution boundary polygons

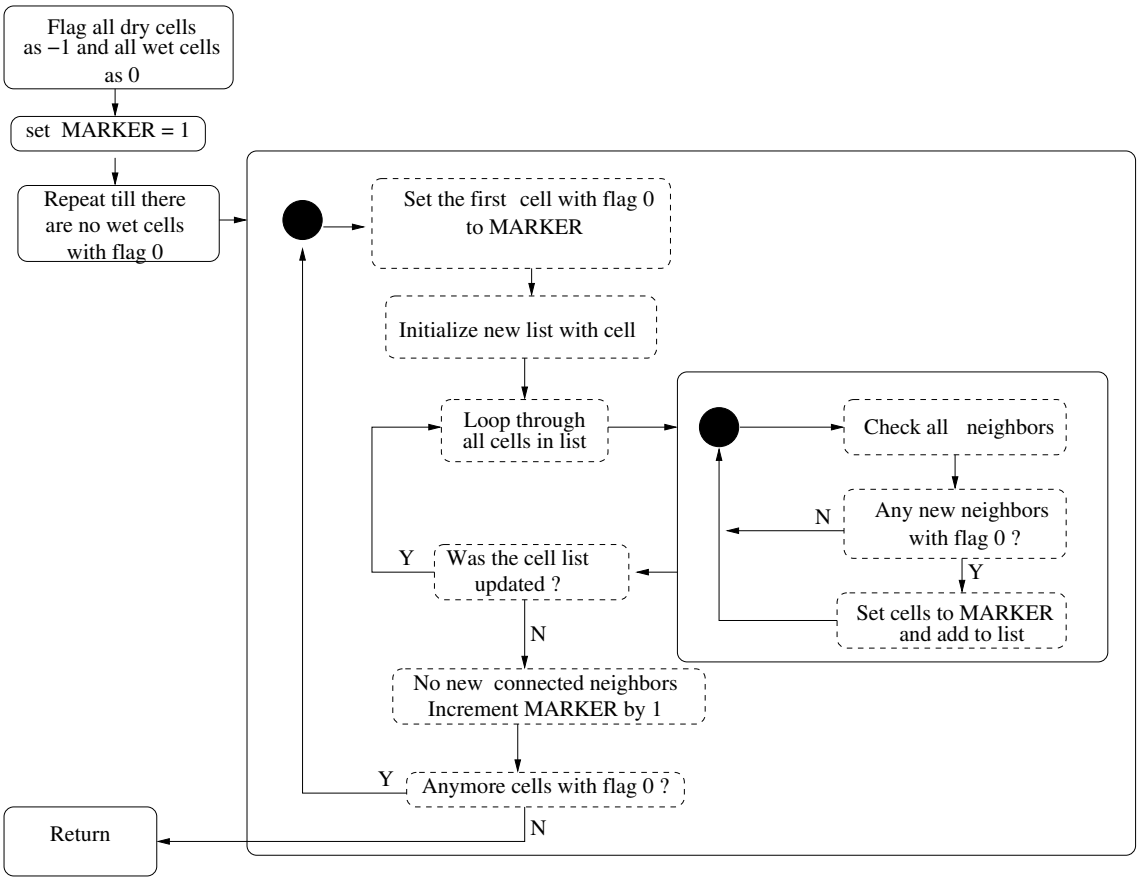


Fig. 2.10 : Flow chart of the wet cell module

splitting of the polygons and at the same time provides a significant speed up in the land-sea mask computation[13].

2.4 Wet cell module

A grid in many cases includes several unconnected water bodies, some of these, particularly those consisting of a few cells, should be removed from the grid. The wet cell module is designed to group the wet cells into independent water bodies. Once the water bodies are identified the user can then decide which to include in the grid. The algorithm is outlined below as well as in the flow chart of Fig 2.10

Step 1: Start from the first wet cell (obtained from the land mask 2D array ***m2*** in section 2.3) and mark it with a unique flag value

[13]The split boundary polygons are only used for determining the land - sea mask. The rest of the modules use the boundary polygons from the boundary module

Step 2: Check its neighbors (neighbors are defined as cells to the left, right, up and down, but not diagonal) to see if they are wet, mark with the same flag and add to a list.

Step 3: Go through each cell in the list and repeat step 2.

Step 4: Keep doing this until no new wet cell neighbor is found.

Step 5: Go to the next unmarked wet cell and set it to a new flag value. Repeat steps 2 to 4 till no new wet cell neighbor is found.

Step 6: Keep repeating until no unmarked wet cell remains.

The command line arguments for the module are

```
>> [m3,mask_map] = remove_lake(m2,tol,icf);
```

Where the input variables are

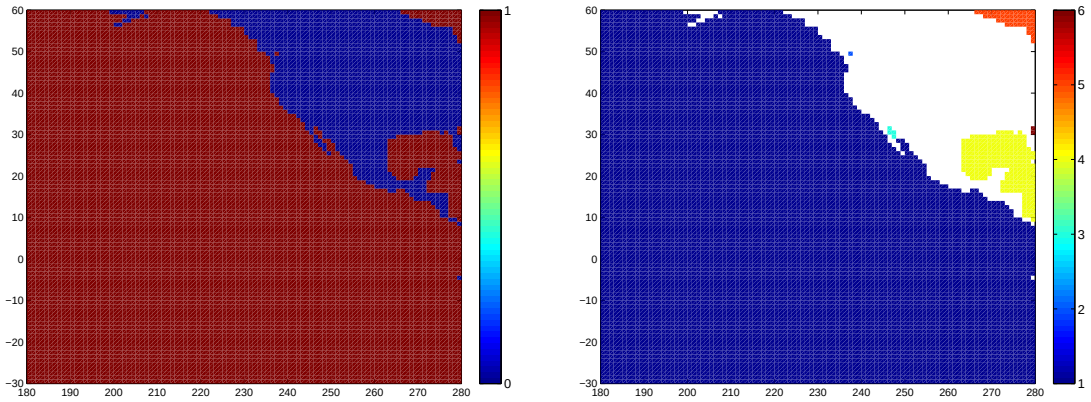
- ***m2*** → The 2D land-sea mask array from the land mask module.
- ***tol*** → An integer value that determines how the mask array ***m2*** is modified. If the ***tol*** is a positive number, then all water bodies which have less number of cells than this value are set to dry. If the ***tol*** is set to a negative number then only the water body with the largest number of cells (the main water body) is kept and all the others are switched to dry cells.
- ***icf*** → A flag to determine if the grid is global (wraps around). Flag of 1 indicates a global grid and 0 indicates a regional grid (i.e. the first and last longitude cells are not connected).

and the output variables are

- ***m3*** → The modified land-sea mask based on the values of ***tol***
- ***mask_map*** → A 2D array the same size as ***m2*** or ***m3*** which contains the flag id values for the wet cells corresponding to different water bodies. In it the land values are set to -1, all the wet cells corresponding to the first water body to 1, wet cells corresponding to the second water body as 2 and so on[14].

Fig 2.11 shows an example of the use of this module. In this example a 1° resolution grid for the Pacific Ocean has been created which also includes parts of the Gulf of Mexico and Hudson Bay. The left hand side is a plot of the original land-sea mask that does not separate out the different water bodies. The right

[14]The wet cells in ***mask_map*** correspond to the wet cells in ***m2***



(a) *land sea mask* (b) *Water bodies with flag ids*

Fig. 2.11 : Land sea mask and wet cell flags for a 1° resolution Pacific ocean grid

hand side is a plot of the variable ***mask_map*** obtained from the land-sea mask. In this case, we get 6 independent water bodies − the Pacific Ocean (ID = 1), the Strait of Juan de Fuca (ID = 2), part of the Gulf of California (ID = 3), both of which are seen here as internal lakes because of poor grid resolution, as well as the Gulf of Mexico (ID = 4), Hudson Bay (ID = 5) and a part of the Atlantic Ocean (ID = 6). Technically these are all water bodies, but depending upon the application only a few of these are relevant. The land-sea mask is modified by the value of ***tol*** which the user sets depending upon the operation in mind. For example, if the user wishes to use the grid for swell propagation in the Pacific Ocean only, then all the smaller water bodies are not needed and the user can choose a negative ***tol*** value to mask them all out. However, if the grid is nested into a larger global grid and the user is interested in swell dynamics for both the Pacific Ocean and the Gulf of Mexico, then the user would choose a positive ***tol*** value to mask out only the smaller water bodies. Keep in mind that setting ***tol*** to 0 will still create the appropriate ***mask_map*** but lead to no operation on the land-sea mask and ***m3*** will be the same as ***m2***.

More often than not the user wants specific water bodies masked out and is not sure what the right value for ***tol*** should be to achieve this. Since the wet cell module recursively searches all wet cells for matching neighbors and can be a computationally intensive algorithm, it is not ideal to keep iterating through this routine until the correct value is specified. Alternatively, ***mask_map*** can be used to modify the land-sea mask interactively. Taking the previous example, if the user is interested in keeping the Pacific Ocean and the Gulf of Mexico he could proceed by masking out all the smaller water bodies with the command

```
>> [m3,mask_map] = remove_lake(m2,-1,0);
```

which would dry out all the water bodies except the largest one (in this case the Pacific Ocean). Then from a plot of ***mask_map*** (see right side plot in Fig 2.11) the user could identify the wet cells that make up the Gulf of Mexico (flag id 4) and switch these cells in ***m3*** from dry back to wet in with the following set of commands

```
>> loc = find(mask_map == 4);
>> m3(loc) = 1;
```

where all the elements (cells) of the 2D array ***mask_map*** that are equal to 4 are identified and these cells which had been turned from wet to dry in ***m3*** are reverted back to wet.

2.5 Sub-grid module

The Sub-grid module makes up the final part of our grid generation package for WAVEWATCH III. Tolman (2003) introduced obstruction grids to block swells propagating through regions with unresolved islands. Going back to the example for the Caribbean region (Fig 2.9) and adding the coastal boundaries to this plot (Fig 2.12) shows that even for a high resolution grid (4') a large number of coastal features are either poorly or not represented by the land-sea mask. Hence, the need for obstruction grids to properly account for swell propagation through this region. Considering the extent of coastal features that would have to be accounted for this would be an arduous manual task which would benefit from automation.

As specified in Tolman (2003) obstruction grids are necessary for both grid axes. These obstruction grids are generated at the same points as the bathymetric grid and represent the amount of energy being blocked in the cells corresponding to the grid points. A value of 0 represents no obstruction and a value of 1 complete obstruction. Fig 2.13 shows a sketch of how a sub-grid obstruction is computed. Along the x direction, the obstruction is determined by how much of the cell height is blocked by the island (shown by the vertical double arrow line) and along the y direction the obstruction is determined by how much of the cell width is blocked by the island (shown by the horizontal double arrow line). Obstruction grid values for dry cells are kept at 0 and non-zero values are only generated for wet cells with unresolved boundaries in them. To prevent spurious attenuation of swell traveling into the coast, obstruction values for cells next to dry cells are also set to zero. Thus, if a particular cell is flagged as a dry cell then in the x direction obstruction grid (hereafter represented by the variable ***Sx***) cells to the left and right of the dry cell will be set to 0 and in the y direction obstruction grid (hereafter represented by the variable ***Sy***) cells above and below the dry cell are set to 0.

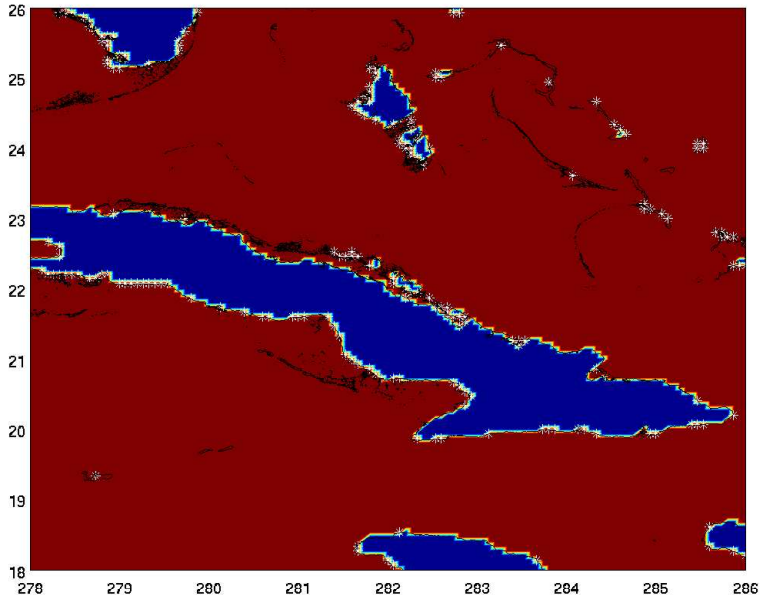


Fig. 2.12 : Land mask for the Caribbean in Fig 2.9 with the coastal boundary polygons in black added on

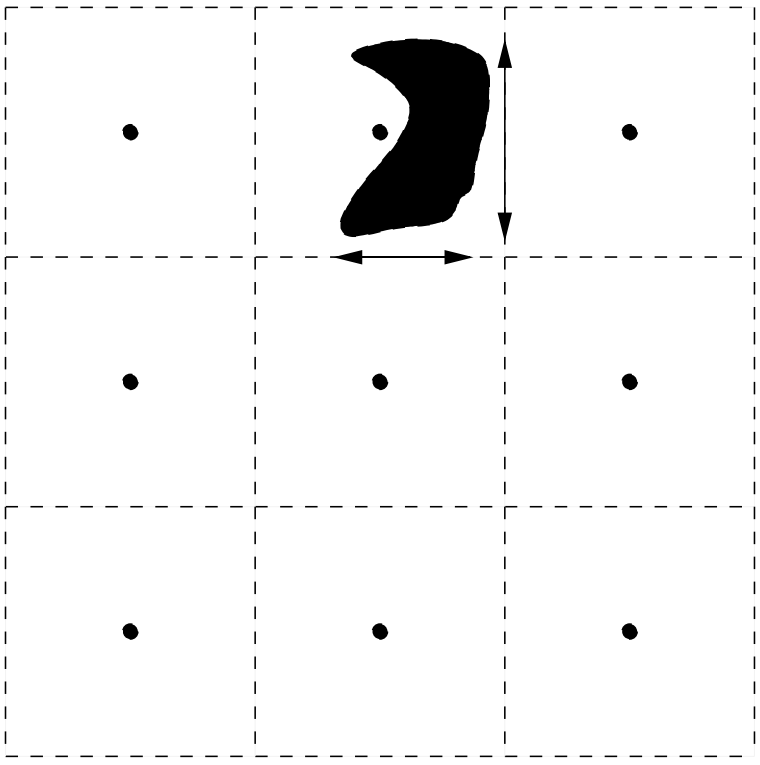


Fig. 2.13 : Representation of an island as a sub-grid obstruction. Solid circle represents the grid points where the obstruction sets are calculated, dashed lines represent the corresponding cells

There are several complications in constructing obstruction grids (Tolman, 2003, Fig 1).(i) What should be done with boundaries that overlap two or more cells (but are still not resolved by the grid).(ii) How do we account for multiple boundaries in a cell.(iii) Do we take the effect of neighbors into account while building the obstruction grid sets and how. These points will be considered separately below.

(a) Boundaries overlapping cells

When boundaries overlap cells then there are potential problems in determining obstruction data sets. For $\boldsymbol{Sx}$ the problems occur when the overlaps are over cells in the same row and for $\boldsymbol{Sy}$ the problems occur when the overlaps are over cells in the same column (see Fig 2.14). In the figure there are two boundaries that cover multiple cells. Boundary 1 overlaps cells **B**, **C** and **F**, while boundary 2 overlaps cells **E** and **H**. From the orientation of boundary 1 we can see that wave energy propagating in the x direction from cell **A** to cell **C** should be completely blocked. However, if $\boldsymbol{Sx}$ values for cell **B** and cell **C** are to be determined based only on the proportion of the boundary found inside each cell then only partial blocking in each cell will take place. Alternatively, since the same boundary segment occurs in both the cells, we could move the segment from one cell to the other and use that to compute the obstruction grids. This will lead to complete blocking in one grid and no blocking in the other. Since our aim is to model the correct amount of sub-grid energy at the resolution of the coarse grid, moving the segment from one cell to the other is acceptable. For convenience, we move the segments to the cell with the larger boundary segment. The argument does not extend to $\boldsymbol{Sx}$ computations in cell **F** as the obstruction there occurs at a different row, and is not affected by the obstructions in cells **B** and **C** (at least at the local level). The same argument also holds for $\boldsymbol{Sy}$ computations in cells **E** and **H** with respect to boundary 2[15].

(b) Boundaries in the shadow zone

Consider a single cell with multiple boundaries (Fig 2.15). If we were to compute $\boldsymbol{Sx}$ and $\boldsymbol{Sy}$ values by adding up all the heights and widths of the boundaries, they would exceed the height and width of the cell respectively leading to full obstruction. However, a closer inspection of the boundaries show that for calculating $\boldsymbol{Sx}$, boundaries 3 and 4 lie in the shadow zone of boundary 1 and should have no role in determining the obstruction segments, while only a part of boundary 2 should play a role in determining the obstruction segments. Similarly for $\boldsymbol{Sy}$ boundary 2 should play no role and only parts of boundaries 1, 3 and 4 should be

[15]Moving obstructions to a single cell also avoids "double counting" as identified in Fig 1 of Tolman (2003)

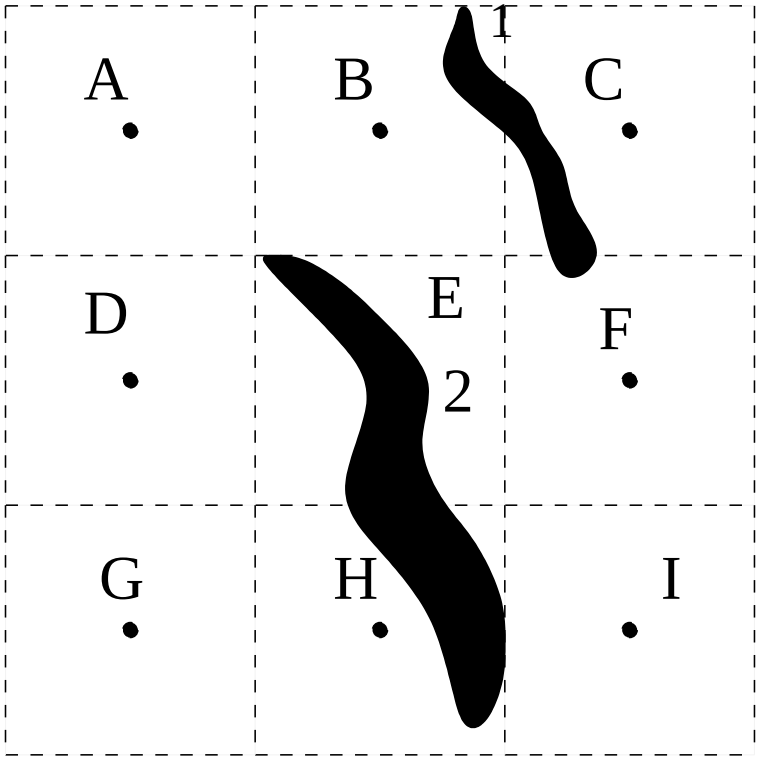


Fig. 2.14 : Sub-grid obstructions for boundaries overlapping multiple cells

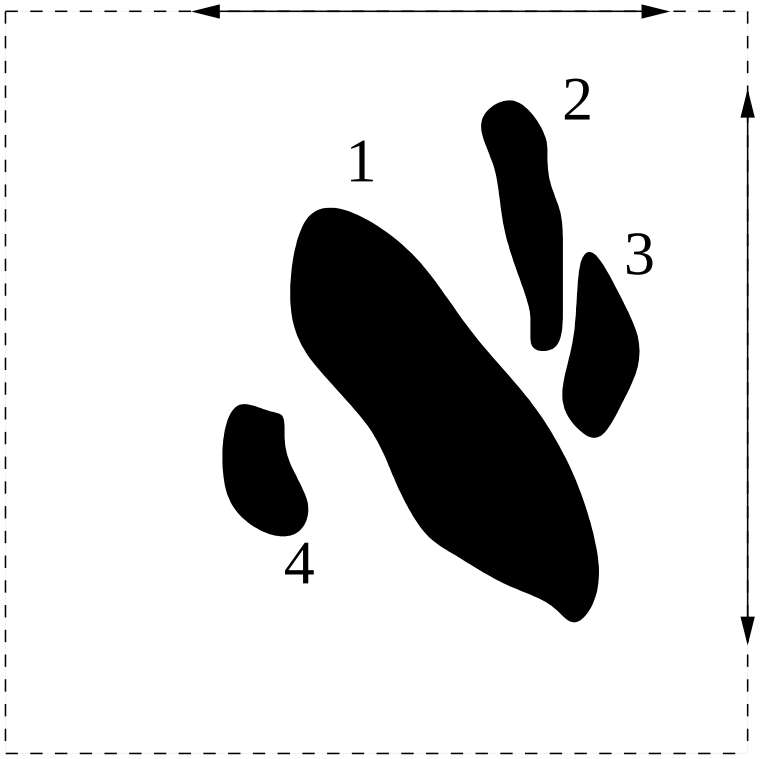


Fig. 2.15 : Multiple boundaries and the resulting net obstruction segments along the 2 directions in a single cell

used in determining the actual obstruction segment(s). Thus, in any particular direction a net obstruction segment needs to be generated which removes segments that are completely overlapped and joins segments that are only partially overlapped, resulting in net segments shown by the double arrow lines in Fig 2.15. This provides a more accurate representation of the obstruction process.

(c) Accounting for neighboring cells

A final factor to consider in building the obstruction grids are the orientations of the boundaries in neighboring cells. Since obstruction grids only compute the attenuation of energy due to sub-grid blocking, and not where in the cell this blocking occurs, the orientation of boundaries in neighboring cells will have to be accounted for in the building of the obstruction grids if this process needs to be reproduced at some level. We can illustrate this with an example for computing $\boldsymbol{Sx}$ grid along a row segment as shown in Fig 2.16 (Similar arguments will hold for $\boldsymbol{Sy}$ grid along a column segment). If the obstruction grid was computed for each cell without taking into account the boundaries in the neighboring cells then $\boldsymbol{Sx}$ values for cells **A** and **B** in the figure would be approximately 0.5. However the boundaries are oriented in such a manner that blocking in cell **A** occurs in

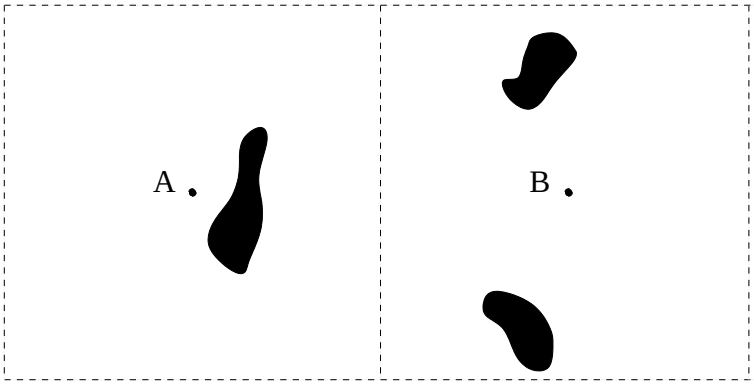


Fig. 2.16 : Orientation of boundaries in neighboring cells

the center while blocking in cell **B** occurs along the sides. Thus, as the waves propagate through the cells most of the wave energy should get blocked, and the obstruction values should be higher. Alternatively, if the islands in cell **B** lie in the shadow of the island in cell **A** then no obstruction should occur in cell **B**. We can take these effects into account by including the boundary segments of the neighboring cells when building the net segments.

If obstruction grids are to be built by taking obstructions in neighboring cells into account then a few questions arise. How many neighboring cells should be taken into account ? Should preference be shown to one direction or the other ? Since we are trying to best simulate local processes, we shall limit our checking to only the immediate neighbors. This also has the added advantage of keeping our algorithm relatively simple. Considering obstructions in adjacent cells is particularly important to assure full blocking of distributed islands. Considering more than directly adjacent cells is inconsistent with the resulting grids.

Fig 2.17 illustrates the computation of $\boldsymbol{Sx}$ along a row of cells. First consider using both neighbors[16]. In this case contributions for cell **A** to $\boldsymbol{Sx}$ come from cell **B**, for cell **B** from cells **B** and **C**, for cell **C** also from cells **B** and **C** (cell **D** boundary lies in the shadow of cell **B**), for cell **D** the contribution comes from cell **C** and for cell **E** contribution comes from cell **D**. We can identify over-obstruction in several cases. First, cells **E** and **A** contain no obstructions and thus should have $\boldsymbol{Sx} = 0$. However, because of information from neighboring cells, non-zero values are being registered for these cells. This is easily fixed by ensuring that obstruction values are only assigned if the cell in question is contributing to the obstruction of wave energy. Then the $\boldsymbol{Sx}$ values in cells **A** and **E** (because of no boundaries) as well as cell **D** (because the boundary lies in a shadow zone of neighboring cells) would be set to 0. This however does not completely remove the over obstruction. For energy propagating from left to right, the effects of cells

[16]Note that the obstructions considered here are computed with the previously discussed algorithm but by considering the expanded area covered by multiple cells

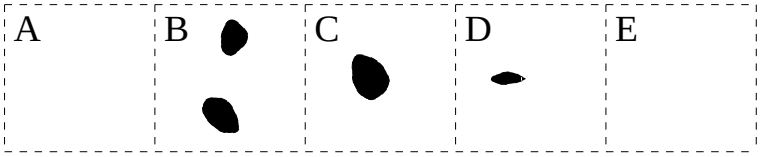


Fig. 2.17 : Using neighboring cell boundary information for computing ***Sx***

B and **C** would be felt both in cell **B** as well as cell **C**.

On the other hand we can intentionally introduce directional bias in our ***Sx*** computations by taking only neighbors from one side into account. So we build two sets of ***Sx*** grids, one which is built taking account of neighbors on the left and the other taking into account neighbors on the right. In the case of the example looking to the left only, the contribution to cell **B** will come from **B** itself, for cell **C** from **B** and **C** and so on. For waves propagating from left to right this creates an obstruction data set with no over obstruction. If on the other hand we take neighbors on the right hand side to compute the ***Sx*** grid then, contribution to cell **C** would now come from **C** itself (since cell **D** is in the shadow of cell **C**), and for cell **B** from cells **B** and **C** and so on. Again for waves propagating from right to left this set of obstruction values would not lead to over obstruction. Thus, the advantage of having directionally biased obstruction values would be that over obstruction would not be an issue. However, the disadvantage would be that now you would have two sets of ***Sx*** grids and the choice of grid would depend upon the direction of wave propagation[17]. We shall use numerical experiments in section 3 to explore the impact of building obstruction grids with neighboring cells on the energy attenuation of propagating swells.

Based on the considerations outlined above, the algorithm for building sub-grid obstructions proceeds as shown in the following steps as well as in the flow diagrams in Figs 2.18 and 2.19 (the flow chart was too big to fit in one figure and was thus split into two).

Step 1: Loop through all the boundaries and for each boundary determine all the cells that the boundary passes through. For each wet cell (this automatically discards all cells that are enclosed by boundaries) determine the North, South, East and West limits of the boundary in that cell. At the end of this loop the information available for each cell are the number of boundaries that are fully / partially enclosed in the cell, and for each boundary their upper and lower limits in the 2 directions.

Step 2: Now loop through all the cells and for each cell that is wet and has boundaries, compare with neighboring cells (for x direction we compare with the cell to the right and for the y direction we compare with the cell above)

[17]Note that WW-III already uses directionally dependent obstructions internally (Tolman, 2003). Adding directionally dependent grids would be relatively straightforward

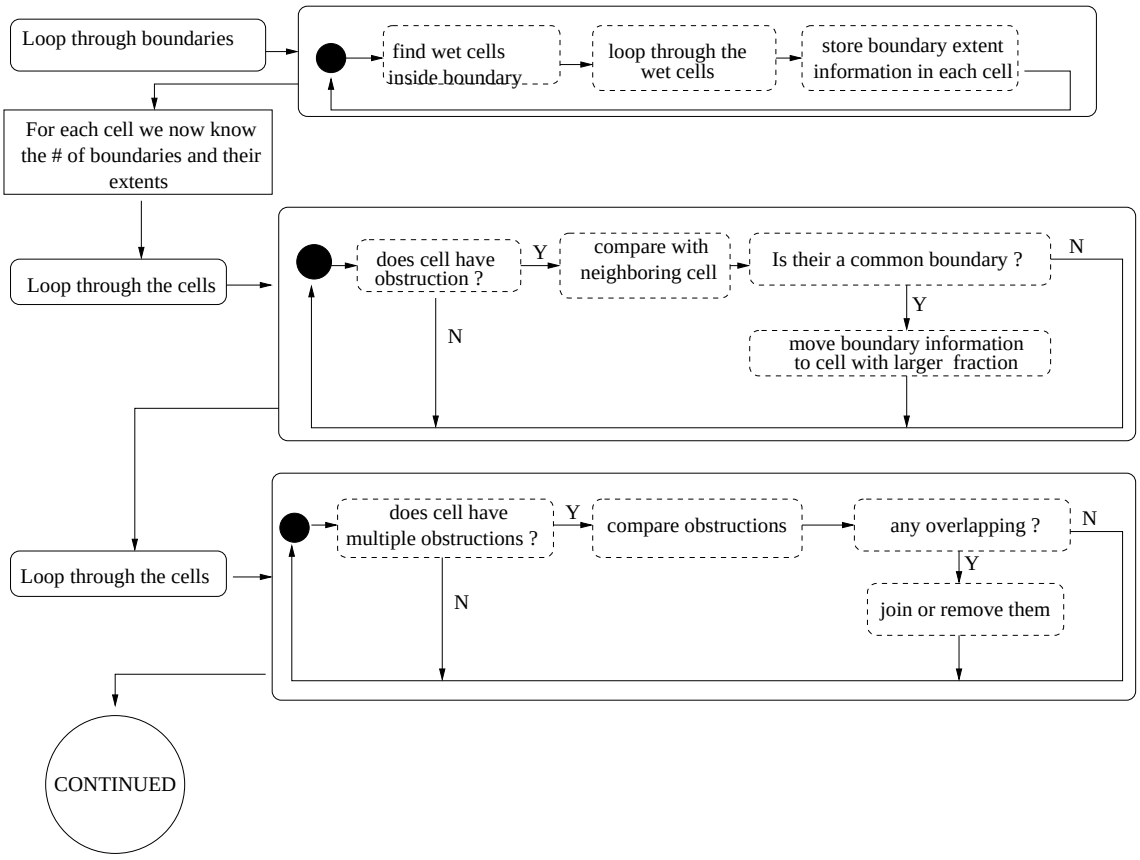


Fig. 2.18 : PART 1 of the flow chart of the sub−grid module algorithm (outlines the steps 1 to 3 of the algorithm)

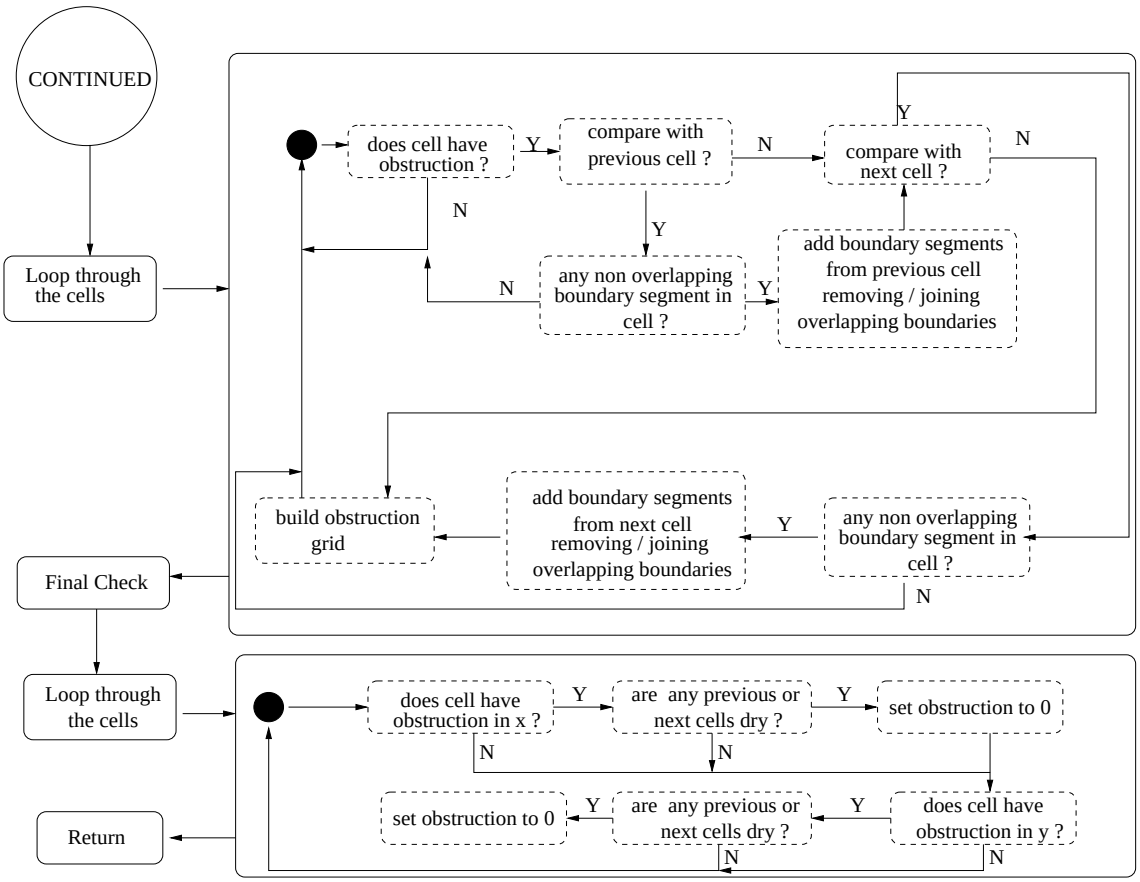


Fig. 2.19 : PART 2 of the flow chart of the sub−grid module algorithm (this is a continuation of the flow chart from Fig 2.18 and outlines the steps 4 and 5 of the algorithm)

to check if they share common boundaries. If yes, then move the boundary segment from the cell that has a smaller portion to the cell that has a larger portion so that the obstruction effect from a particular feature occurs over one cell. This prevents double counting of the same coastal feature in neighboring cells[18].

Step 3: Now that overlapping boundaries have been moved to one grid or the other, once again loop through the cells and determine the net segments in both x and y directions for each cell, removing boundary segments that are in the shadow zone of another boundary segment and joining segments that overlap.

Step 4: Now loop through the cells to build the obstruction grids. As outlined above there are three ways that this can be done − only account for boundary segments in the cell, account for boundary segments in both neighboring cells (left and right for x direction and above and below for y direction), or account for boundary segments in only one neighboring cell. All three options are provided in the algorithm. If neighboring cell information is to be accounted for then the segments of the neighboring cells are compared with the segments in the cell just like in step 3 and a set of non overlapping boundary segments determined. If during the determination of the non overlapping segments, all the boundary segments in the cell under consideration are removed (because they lie in the shadows of the boundary segments in the neighboring cells) then obstruction for that cell is set to zero (since the boundary segments of the cell are not contributing to the obstruction of the energy). If on the other hand there are unique boundary segments in the cell then the obstruction values ($\boldsymbol{Sx}$ and $\boldsymbol{Sy}$) for the cell are computed using the following formulas

$$Sx = \frac{\sum h_{seg}}{h_{cell}}$$

and

$$Sy = \frac{\sum w_{seg}}{w_{cell}}$$

where, h_{seg} are the heights of the individual non-overlapping boundary segments, h_{cell} is the height of the cell, w_{seg} are the widths of the individual non overlapping boundary segments and w_{cell} is the width of the cell.

[18]We are only concerned with cells in the same row for x direction and cells in the same column for y direction. If a coastal feature is distributed over more than two cells and still shows up as a sub-grid obstruction then we let the obstruction effect be split over the different cells

Step 5: The final step in the obstruction grid computation is to check the neighbors of the cells with non-zero obstruction values. If either of these neighbors are dry cells the obstruction along that direction is set to zero.

The syntax for the function call is as follows

```
>> [Sx,Sy] = create_obstr(lon,lat,b,m,icoords,off_left,off_right);
```

where the input variables are

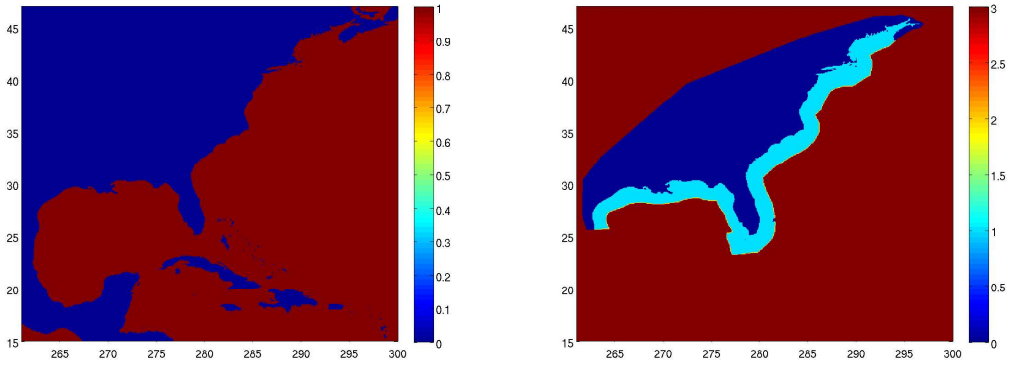
- ***lon***,***lat*** → The grid coordinates obtained from the grid generation module
- ***icoords*** → Flag for orientation of the longitudes (see section 2.1)
- ***b*** → The array of polygon data structures from the boundary module
- ***m*** → The final set of land-sea masks from the wet cell module
- ***off_left, off_right*** → Values to determine if neighbor information is to be taken into account. If ***off_left*** is set to 1 then cell to the left (below) is taken into account for computing ***Sx*** (***Sy***) values. If ***off_right*** is set to 1 then cell to the right (above) is taken into account for computing ***Sx*** (***Sy***) values. If both are set to 0 then the obstruction grid is determined by not taking neighboring cell information into account, and if both are set to 1 then neighboring information on both sides are taken into account

The output variables are ***Sx*** and ***Sy*** the 2D obstruction grid arrays in the x and y directions respectively. This information together with the grid information is provided as input for WAVEWATCH III.

2.6 Mask modification module

This module is needed only when working with WAVEWATCH III version 3.10 or higher. From version 3 onwards WAVEWATCH III has become a multi - grid model which uses nested grids with with two way nesting (Tolman, 2006). The land - sea mask and obstruction grids for the multi-grid model are built in the same manner as the earlier versions. However, some modifications to the land-sea mask are needed to take full advantage of added capabilities[19]. For computational efficiency, coupling between nested grids is not limited to the edges of the grids. Fig 2.20 illustrates the differences between the land-sea masks for the older and new versions of WAVEWATCH III. This grid was generated for high resolution simulations along the United States Gulf and East coasts, and is nested inside a larger regional 10' grid (which is further nested inside a global 30' grid). The left

[19]Only grids that get boundary data from another "upper level" or "parent" grid need to be modified



(a) *Land-Sea Mask for version 2.22*

(b) *Land-Sea Mask for version 3.10 and higher*

Fig. 2.20 : Land-Sea Masks for a 4' US East Coast grid. This grid is nested inside a larger 10' grid.

hand panel shows the land - sea masks for version 2.22 of WAVEWATCH III. This mask was generated using the algorithms described in this report. However, the high resolution grid is needed only up to a certain distance offshore (≈ 60 nautical miles), making a significant number of computational points unnecessary. In version 3.10 of WAVEWATCH III, the computational points that are not needed in the calculation can be switched off and the right hand panel in Fig 2.20 shows the land-sea mask for this model. Instead of 2 the mask now has 4 values $-$ 0 for land, 1 for water, 2 for boundary nodes[20] and 3 for nodes that are ignored in the computation.

The mask modification module is used to modify the mask for WAVEWATCH III version 3.10. The syntax for the call to the function is

```
>> [m2] = modify_mask(m,lon,lat,px,py,mb,lonb,latb);
```

where the input variables are

- ***m*** $\rightarrow$ the final land - sea mask that is generated for WAVEWATCH III v2.22 from the wet cell module described in section 2.4
- ***lon,lat*** $\rightarrow$ longitudes and latitudes corresponding to the grid.
- ***px,py*** $\rightarrow$ x and y coordinates that make up the polygon defining the region of interest for the grid[21].

[20] Boundary conditions from the parent grid (in this case the regional 10' grid) are applied at these nodes

[21] The polygon must be closed, i.e. the first and final points should be the same

- ***mb*** → land - sea mask for the parent grid which provide the active boundary points..
- ***lonb,latb*** → corresponding longitudes and latitudes.

and the output variable ***m2*** is the modified mask with 4 possible values as shown in the right hand panel of Fig 2.20.

The algorithm proceeds as follows

Step 1: Using the polygon and grid information the nodes are separated into those that lie outside the polygon, on the polygon and inside the polygon.

Step 2: All the points that lie outside the polygon are flagged with the value 3 while all the wet points that lie on the polygon are flagged with the value 2[22].

Step 3: For each row the algorithm proceeds down the columns till the flag changes from 3 to 1 or vice versa. It then checks the neighborhood of points to determine which cell has the largest segment of the polygon passing through it. That cell is where the boundary condition is to be applied and it is flagged with a value of 2.

Step 4: The above step is repeated for all the columns.

Step 5: Finally the edges of the grid are checked to see if they lie inside the polygon. Those edges that are inside the polygon and are wet have their flag value switched to 2.

Step 6: Once all the boundary points have been assigned, the algorithm checks to make sure that a boundary condition can be prescribed at each of these points from the parent grid. To do so it compares the boundary point with corresponding neighbor points in the parent grid. If enough wet points[23] from that grid are found so that a reliable boundary condition can be obtained then the flag value remains unchanged. Otherwise the flag value is changed to 3.

[22]Dry points do not provide any boundary data to the grid and are hence ignored

[23]For coinciding points this corresponds to 1, for 1 coinciding axis it corresponds to 2, otherwise it corresponds to 4

This page is intentionally left blank.

3 Numerical test cases

To determine how well the obstruction grids attenuate wave energy, we conducted a series of swell propagation studies in 3 different regions − the Caribbean, Hawaii and French Polynesia. Grids for the Caribbean and Hawaii were generated using **ETOPO2** data and for French Polynesia using **DBDB2** data.

Our design experiment consisted of a constant swell propagating from the North West at an angle of 45° with a significant wave height of 4 m. The swell is monochromatic with a peak frequency of 0.1 Hz. The frequency spectrum is discretized by 3 components with the middle frequency at 0.1 Hz. The directional spectrum is of the cos type with a power of 8, yielding a directional spread of ±30° about the swell propagation direction. Discretization of the directional spectrum has a significant impact on the development of the "garden sprinkler effect" (Booij and Holthuijsen, 1987) and is discussed in some detail in section 3.1. The spatial spreads of the swell were made large enough that nearly constant swell boundary conditions could be applied. These are prescribed along the Northern and Eastern boundaries. In all the cases the runs were continued until steady state conditions were reached.

For each test case simulations were carried out at 5 different resolutions (2',4',8',15' and 30'), both with and without obstruction grids. The corresponding global time steps for the different grid resolutions were 300, 600, 1200, 1800 and 1800 sec respectively. Refraction effects were switched off in the model runs to eliminate cumulative differences associated with bathymetry representation at the different grid resolutions. Simulations at the highest (2') resolution were generally used as ground truth. Since this is a test of energy propagation only significant wave heights were compared. When comparing results at two different resolutions, the results were averaged over all the cells in the higher resolution grid that lie inside the cells of the lower resolution grid, so that differences can be assessed directly.

A point raised while building the obstruction grid algorithm in section 2.5 was the possibility of over obstruction when accounting for coastal features of neighboring cells. Alternative suggested methods were to only account for obstructions in individual cells or to consider only the neighbor in the path of the swell. For the test cases considered here where a constant swell is propagating in from the North - West, the correct choice would be to use the neighbor to the right (up) in x (y) direction. The three different ways to build the obstruction grids − not accounting for obstructions in neighboring cells (hereby referred to as Test OB-0), accounting for obstructions in both neighboring cells (hereby referred to as Test OB-2) and accounting for obstruction in only one neighboring cell (hereby referred to as Test OB-1) − will be compared with the impact of not taking any obstructions into account (hereby referred to as Test NO-OB).

3.1 Caribbean

The Caribbean region with the small islands of the Bahamas and Lesser Antilles provide a good test case for our grid generation package, in particular the obstruction algorithm. Fig 3.1 shows how well the coast lines are represented in the different grids in relation to the **GSHHS** database. From the figure we can see that even the 2' resolution grid cannot reproduce all the islands in the Bahamas (particularly in the North) and some of the Lesser Antilles islands in the west ($10°N - 16°N$ and $65°W - 60°W$). As the resolution decreases, this representation gets worse, for the 8' resolution most of the obstruction effects by the islands of the Bahamas and the Lesser Antilles on the swell propagation can only be modeled using obstruction grids.

Garden Sprinkler Effect

Due to the discretized representation of the ocean spectrum in spectral wave models, spatial spreading of the spectrum occurs along discrete bands leading to what is known as the "garden sprinkler effect"(GSE). If the discretization is too coarse, continuous dispersion disintegrates into discrete wave fields. This becomes particularly visible behind islands and obstacles. Since our test cases consist of a monochromatic wave component in frequency, GSE effects arise primarily from the discretization of the directional spectrum. To avoid GSE effects the resolution should be such that the spreading (after the swell has crossed through the domain) should not be larger than the mesh size (Booij and Holthuijsen, 1987). This leads to a limit on the directional resolution

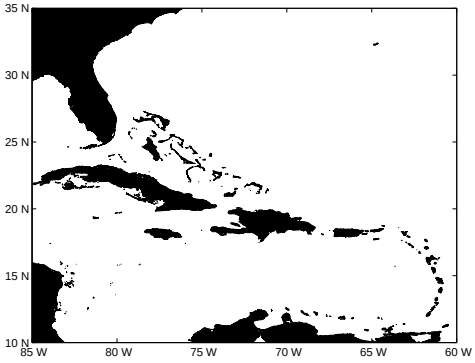
$$\Delta\theta < 1/N,$$

where N is the distance across the ocean in the number of meshes. For a given spectral resolution the greatest GSE effects will be observed in the grid with the finest resolution. For the test cases here, the 2' grid is the finest grid with $N = 751$. This leads to a directional resolution of $\Delta\theta \approx 0.075°$. Clearly GSE effects cannot be avoided here and they can have a significant impact on the wave field (Fig 3.2 (a)).

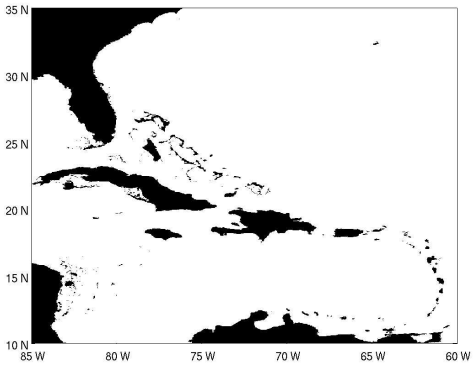
There are two alternative GSE alleviation methods available in WAVEWATCH III[24]. Booij and Holthuijsen (1987) considered the wave action equation[25] for spectral bands (as opposed to spectral components) and developed an alternative spatial propagation scheme that introduced diffusive correction to account for continuous dispersion of the spectrum. Though the technique is very adept at alleviating GSE, numerical stability requirements of this scheme put a limit on the size of Δt (Tolman, 2002a). This is particularly true for the very high

[24] A third method is currently under development

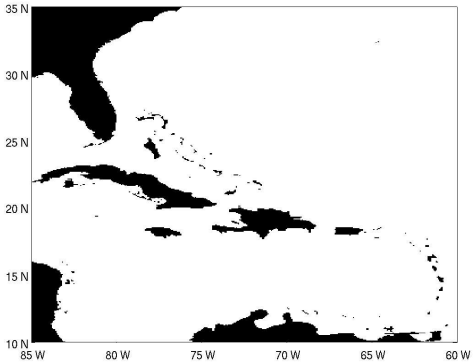
[25] Governing equation for 3^{rd} generation wind wave models



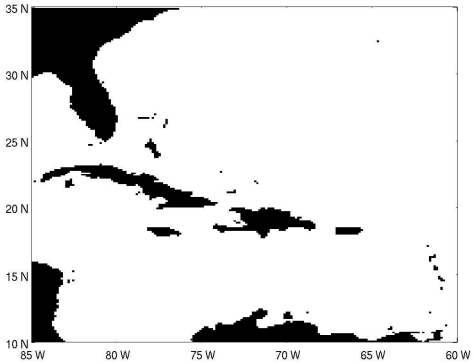
(a) *GSHHS polygons*



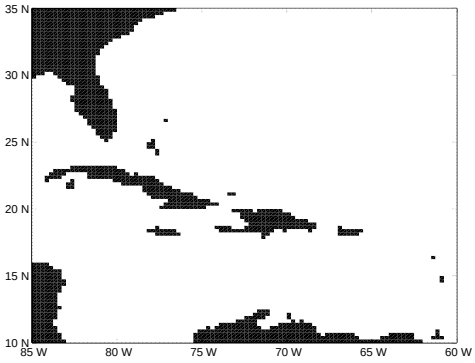
(b) *2' grid*



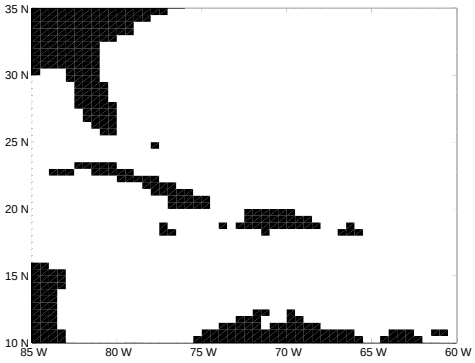
(c) *4' grid*



(d) *8' grid*



(e) *15' grid*



(f) *30' grid*

Fig. 3.1 : Land masks for the Caribbean test case. Fixed boundary conditions of a swell ($H_s = 4m$) propagating from the North West applied along the 30° N and 55° W boundaries.

resolution runs where computational times can increase by greater than 75 %. This approach is not viable with the very high resolution grids that we are working with. An alternative approach suggested by Tolman (2002a) applies a local average, with the averaging box directly proportional to the product of a tunable parameter γ and the time step Δt. Default value for γ in WAVEWATCH III is 1.5 and Tolman (2002a) has shown that this approach yields virtually identical results as Booij and Holthuijsen (1987). The averaging approach also does not add any additional stability constraints on the time step. However, the averaging area is related to the time step Δt and therefore, the extent of GSE alleviation would be related to grid resolution, with the highest grid resolution having the least alleviation (Fig 3.2 (b)). Increasing the tunable parameter γ so that all the different grid resolutions have the same averaging area does not work because in the 2' grid this leads to $\gamma = 12$, which for the given spectral resolution exceeds the grid resolution[26] (Fig 3.2 (c)).

Even if spectral resolution $\Delta\theta$ was changed so that the averaging box is not limited by mesh size, keeping the averaging area unchanged across different grid resolutions does not imply that the same level of GSE alleviation is applied in the different grids because the averaging is done more often in the higher resolution grid (smaller time step). Since our aim is to determine the impact of obstruction grids in blocking swells, we want to avoid introducing additional differences from different levels of GSE alleviation in different grids. Keeping that in mind GSE alleviation in the runs was turned off and the grid resolution $\Delta\theta$ was increased to 2°. Even though this is not enough to completely remove GSE, it does limit its influence (Fig 3.2 (d)).

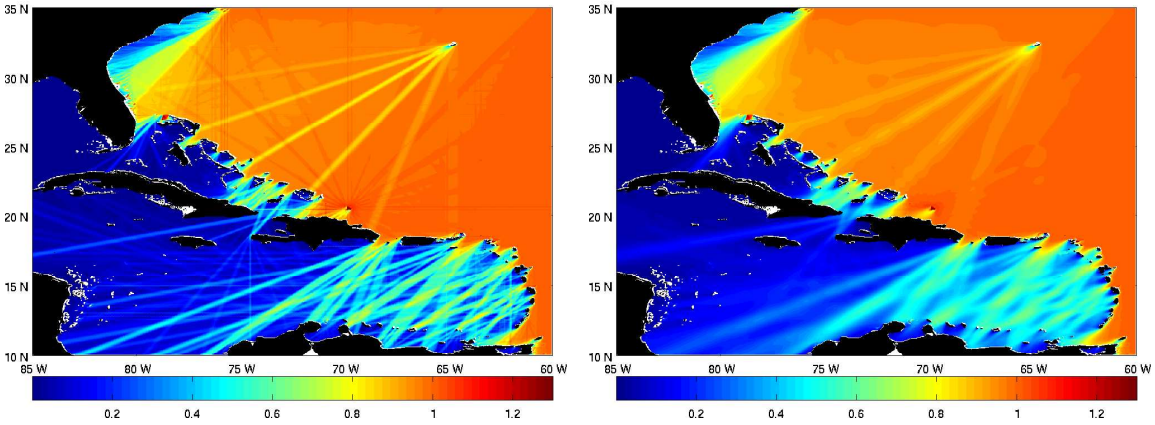
Obstruction results

To assess how well the obstruction algorithm works, we need to have a ground truth solution. A high resolution run where the obstruction grid does not play a significant role would be the ideal ground truth. Fig 3.3 compares the wave height distribution with and without the obstruction for the 2' grid. As can be seen, even at this high resolution the obstruction grid does play a role in suppressing the wave field, particularly behind the Lesser Antilles. However the differences are less than 10 % and we will use the 2' grid resolution run as our ground truth [27]. Note the clear GSE patterns around $10°N - 15°N$ and $85°W - 80°W$ from swell energy propagating through the Lesser Antilles.

The increasing importance of the obstruction grids at lower resolutions can be seen by comparing swell propagation without obstruction (Fig 3.4) and with ob-

[26] WAVEWATCH III currently limits the size of the averaging area to the mesh size (Tolman, 2002a)

[27] When comparing the different scenarios (NO-OB, OB-0, OB-1 or OB-2) the highest resolution grid corresponding to that scenario will be taken as the ground truth



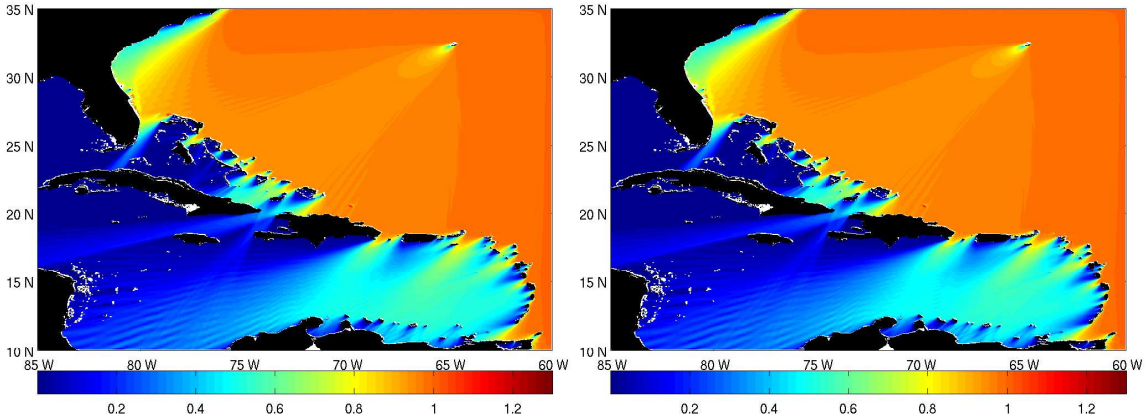
(a) *Without any GSE alleviation and* $\Delta\theta = 15°$

(b) *GSE alleviation with* $\gamma = 1.5, \Delta\theta = 15°$

(c) *GSE alleviation with* $\gamma = 12, \Delta\theta = 15°$

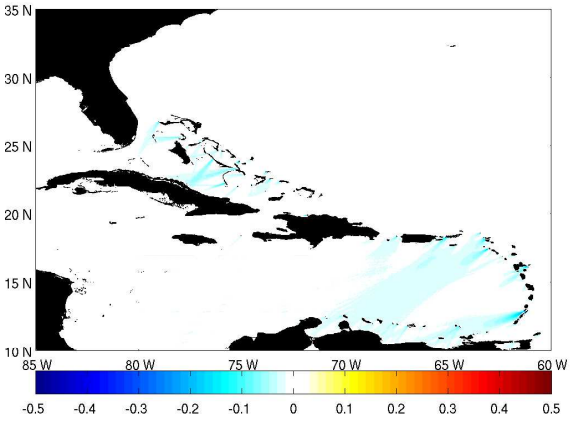
(d) *Without any GSE alleviation and* $\Delta\theta = 2°$

Fig. 3.2 : GSE alleviation in the 2' grid using alternative approaches. Significant wave heights have been normalized by the design spectrum height of $H_s = 4m$



(a) *Test NO-OB*

(b) *Test OB-2*



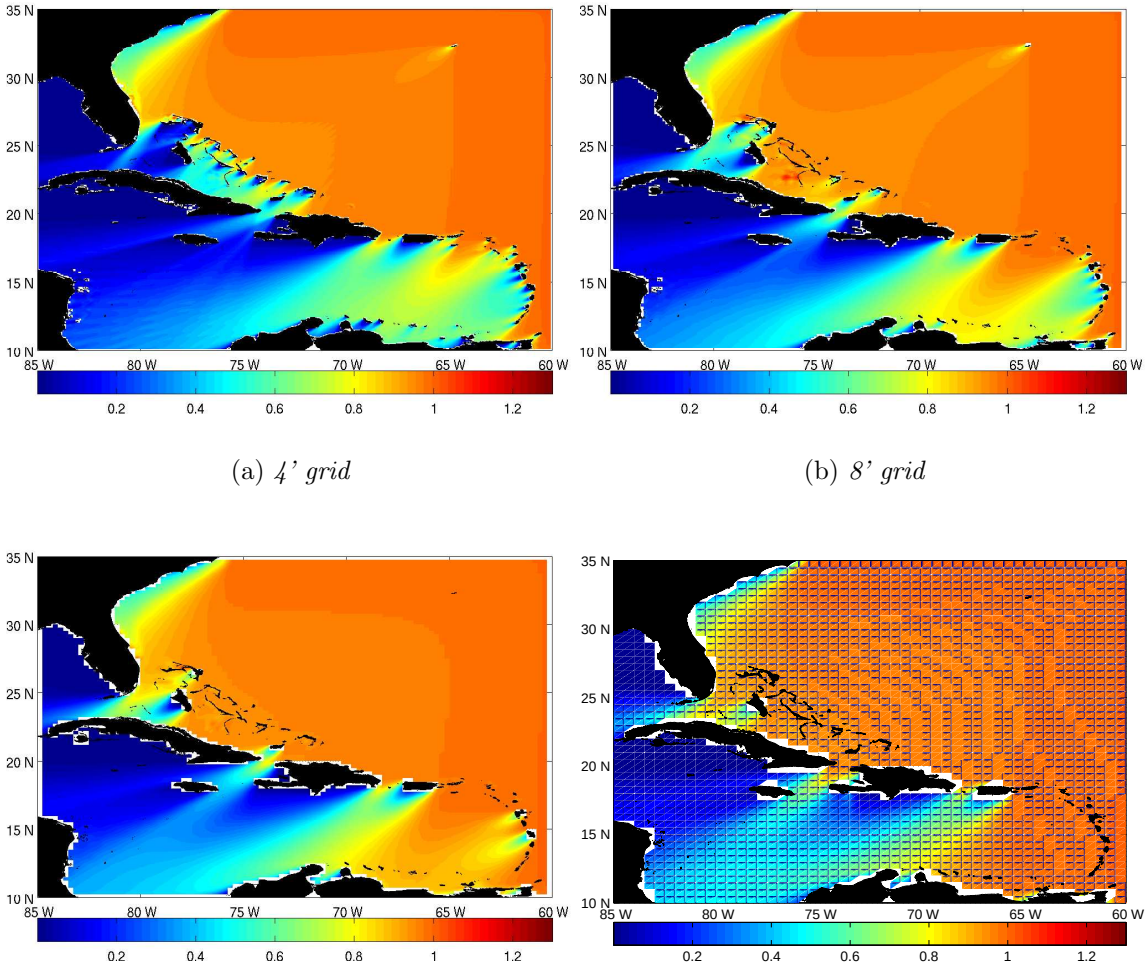
(c) *Difference between the two (with obstruction - no obstruction)*

Fig. 3.3 : Normalized Significant wave heights for 2' grid

struction (Fig 3.5) to the swell propagation in the high resolution run in Fig 3.3. Without obstruction grids we see that even at the 8' resolution a significant portion of the energy propagates through the Bahamas and Lesser Antilles islands, because these islands are all but invisible to the lower resolution grids. Including the obstruction grids re-introduces the blocking effects from the island chains, particularly for the lower resolution grids where the islands are not resolved at all.

The differences between the 2' resolution solution and the lower resolution solutions with and without obstruction grids are shown in Figs 3.6 and 3.7 respectively. In the absence of obstruction grids wave height is consistently over predicted by more than 50 % of the incident wave height behind the island chains, because of the inability of the grids to resolve the islands. Including sub-grid obstruction effects largely corrects this, and differences between the lower resolution and high resolution runs reduces to less than 10 % over most parts of the domain and are mostly localized near the obstructions. After including obstruction, the main causes for differences between low resolution and high resolution runs are due to − (a) representation of headlands and gaps in different grids are different, (b) shoaling effects in shallow waters will be different due to differences in bathymetric representation (refraction processes have been switched off and do not play a role here) and (c) limitations of the obstruction algorithm. The differences between runs (with blocking) are significantly smaller than the removed "first order" effects of lack of blocking of swell propagation, and are therefore acceptable.

The two remaining methods of building obstruction grids (Tests OB-0 and OB-1) are compared in Figs 3.8 and 3.9 respectively. Overall we find that maximum blocking of swell energy occurs for Test OB-2 and the least for Test OB-0. This is as we would expect, thus increasing our confidence in the algorithm. The differences in swell propagation behind the Lesser Antilles islands for the 30' grid are worth noting. Behind the Bahamas where most of the island chains are represented by obstructions (Fig 3.1) Test OB-0 under suppresses the wave energy leading to larger wave heights at Cuba. The same is true for the wave height distribution behind some of the island chains of the Lesser Antilles. On the other hand Test OB-O does a better job in the region behind the Dominican Republic ($15°N$ and $70°W$). This is because the coarser representation of the channels is compensated by the under suppression of wave energy in Test OB-0. We shall look at another example of this in greater detail for the Hawaii test case in the next section. In general however, the results show that though there are some differences between the three approaches, these are fairly minimal and within 10 - 15 % (larger differences are fairly localized around the islands).



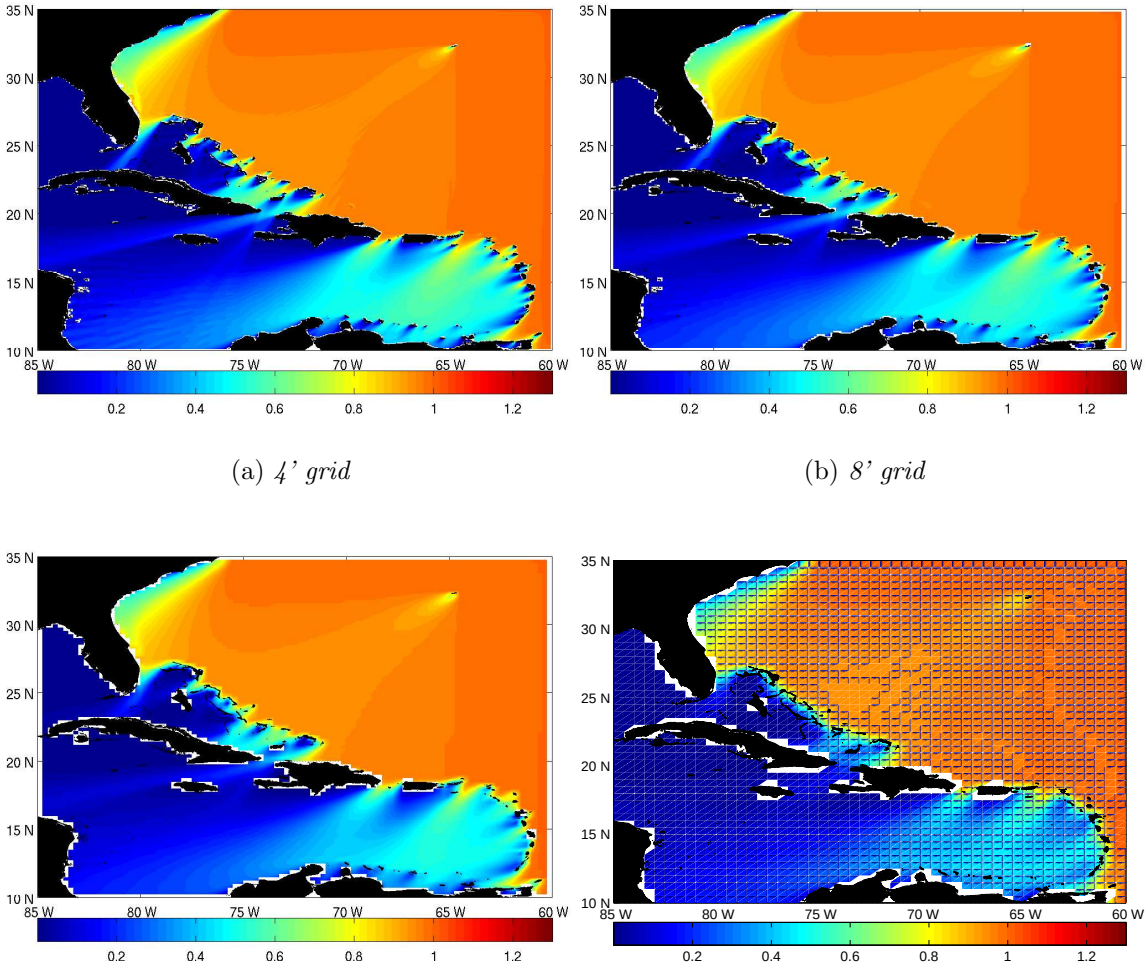
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.4 : Normalized Significant wave heights for Test NO-OB



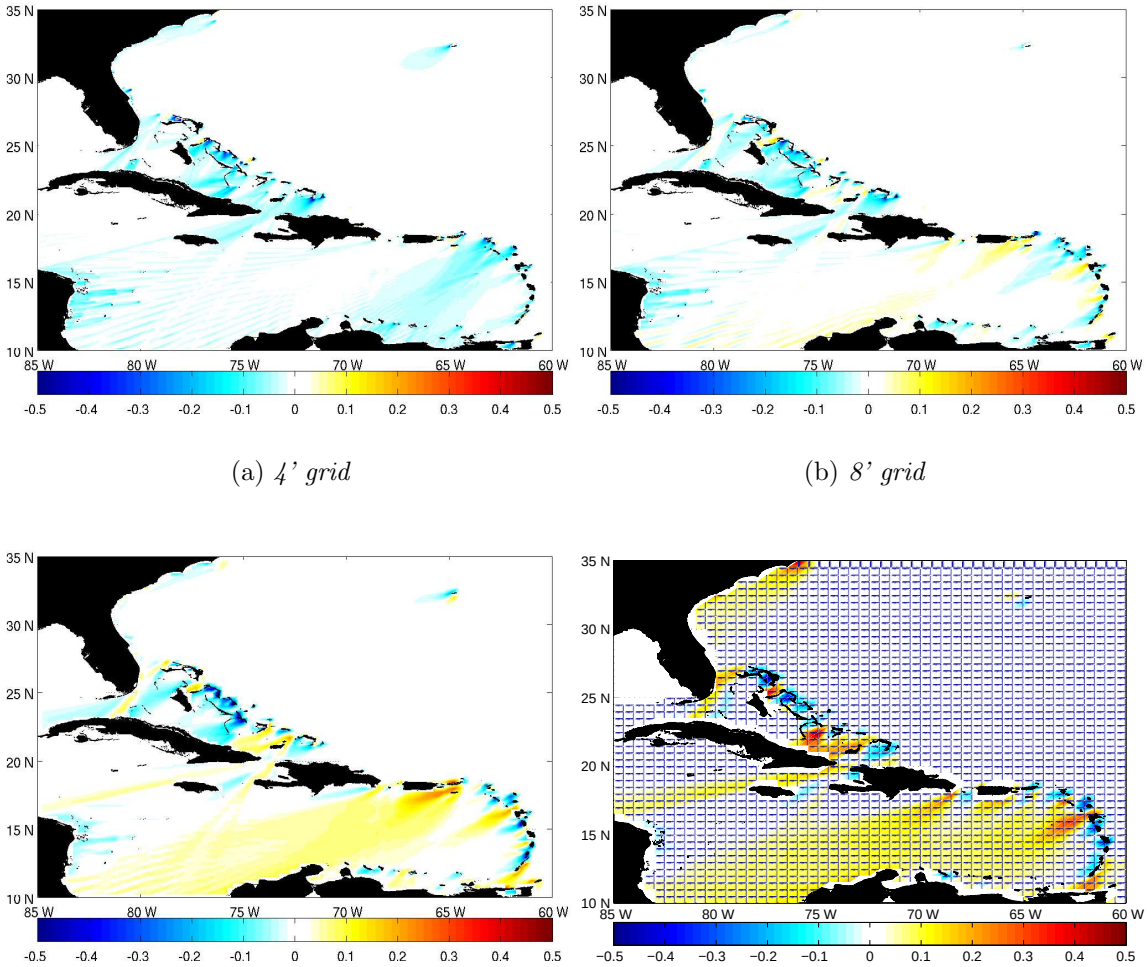
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.5 : Normalized Significant wave heights for Test OB-2



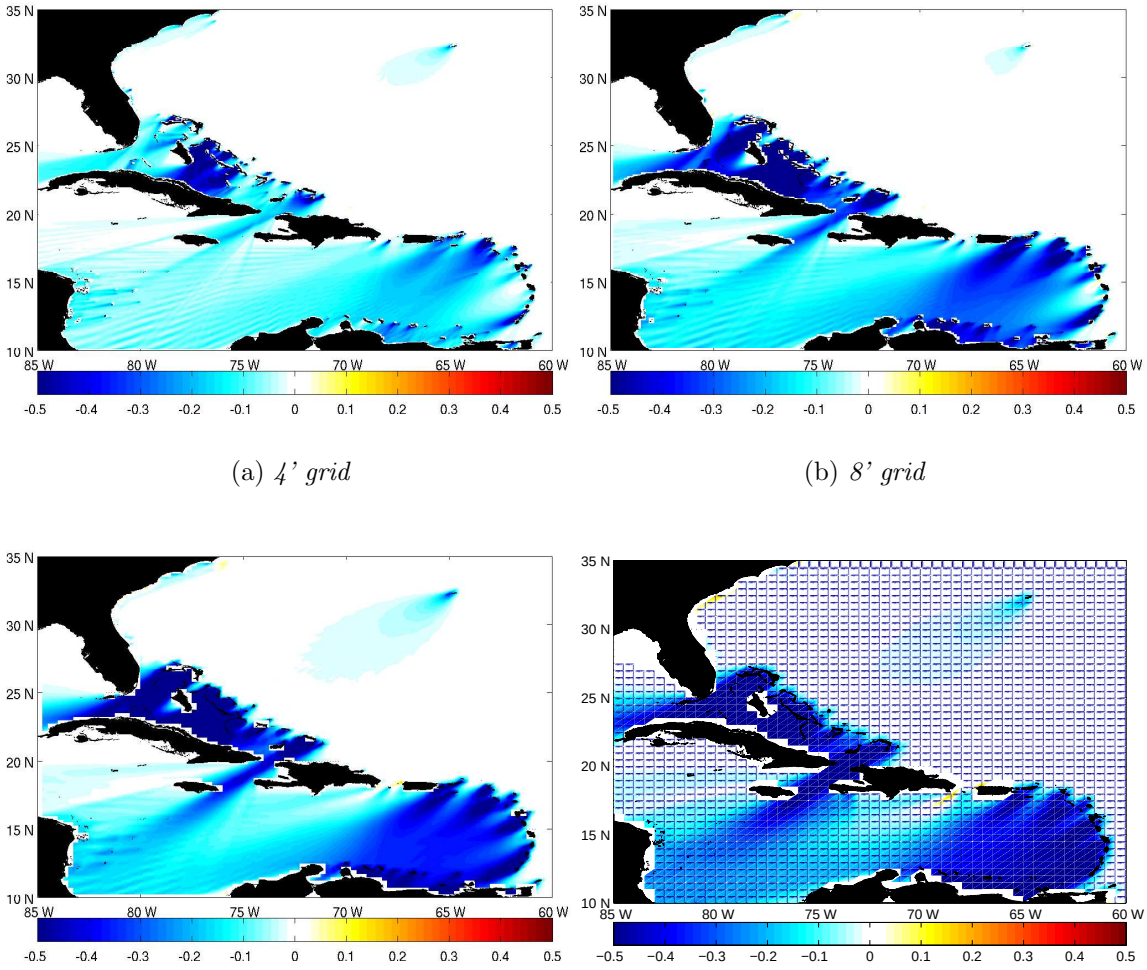
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

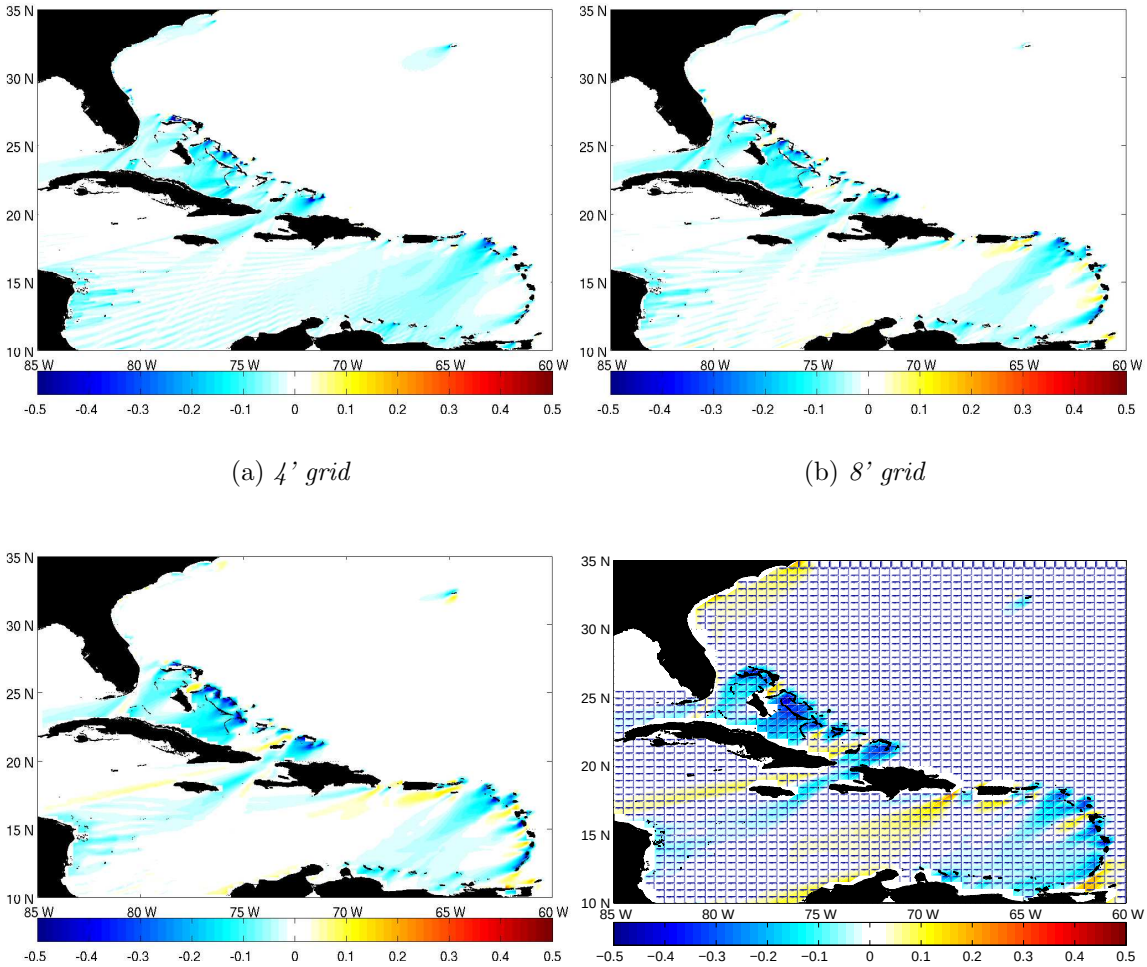
Fig. 3.6 : Differences between 2' and lower resolution grids (Test OB-2)



(a) *4' grid* (b) *8' grid*

(c) *15' grid* (d) *30' grid*

Fig. 3.7 : Differences between 2' and lower resolution grids (Test NO-OB)



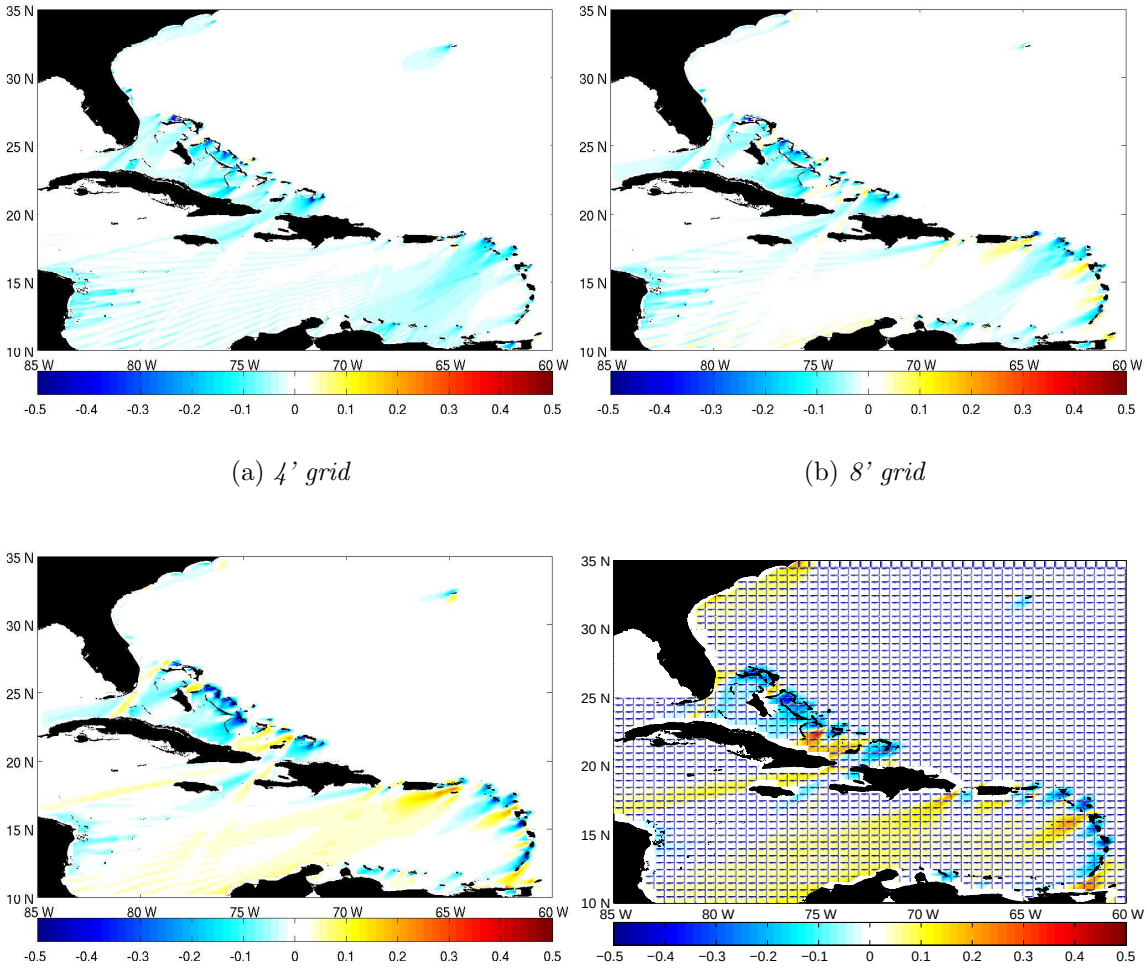
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.8 : Differences between 2' and lower resolution grids (Test OB-0)



(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.9 : Differences between 2' and lower resolution grids (Test OB-1)

3.2 Hawaii

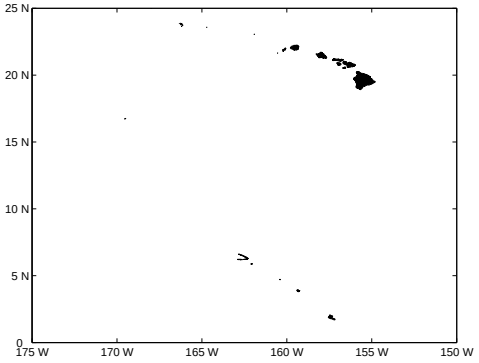
The second test case is the region around the Hawaiian islands in the Pacific Ocean. The water around these islands is deep, so obstruction of swell propagation due to the islands is the main physical process. The islands are well represented at the higher resolution grids, while at the lower resolutions they can only be accounted for as obstructions (see Fig 3.10). At the 2' resolution all the main islands are well resolved. As a result there are no significant differences between the swell propagation runs with and without obstruction (see Fig 3.11[28]). At the 4' resolution the main islands are fairly well represented (though the smaller chain of islands around $157°W$ and $21°N$ − Islands of Molokai, Lanai and Kahoolawe − are starting to lose their structure). At the 15' resolution only the Island of Hawaii (southernmost island also referred to as Big Island) is reasonably resolved and at the 30' resolution even that is not properly resolved.

In the absence of any obstruction (Fig 3.12) we can see that the blocking effects around the islands of Maui and Molokai, Oahu (further north) and Kauai (northern most island) are significantly reduced in the 15' grid and absent in the 30' grid. Including the obstruction grids reproduces the blocking effects (Fig 3.13).

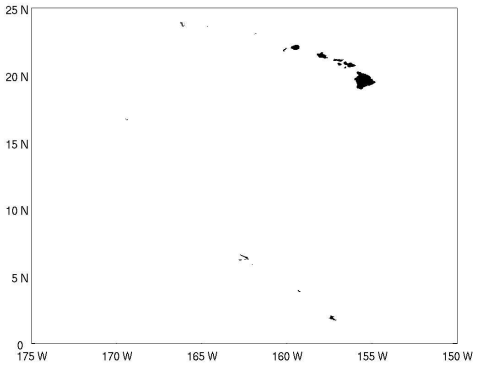
Just like in the Caribbean region test case the obstruction grids reproduce the energy suppression behind the islands. Figs 3.14 to 3.17 compare the lower resolution grids with the ground truth, using different ways of accounting for obstructions. Once again we note that the three different ways of accounting for obstructions in neighboring cells leads to some differences in wave height patterns behind the islands. However, these differences are far less than what occur when obstructions are not taken into account at all.

Unlike the Caribbean test case we find that the best results are obtained here when obstructions are only accounted for in the individual cells (Fig 3.15). This is primarily because the main Hawaiian islands are not made up of a large number of smaller islands where the obstruction in the neighboring cells become important. Once again the biggest differences are in the coarsest resolution grids which are unable to represent channels. Fig 3.18 shows swell propagation through the Alenuihaha Channel (gap between the two southernmost islands of Hawaii − Big Island and Maui) for the 2' and 30' grids. In this case, the poor representation of the channel in the coarser grid is compensated for by the lesser obstruction of swell energy in Test OB-0 leading to a better representation of wave heights behind the islands.

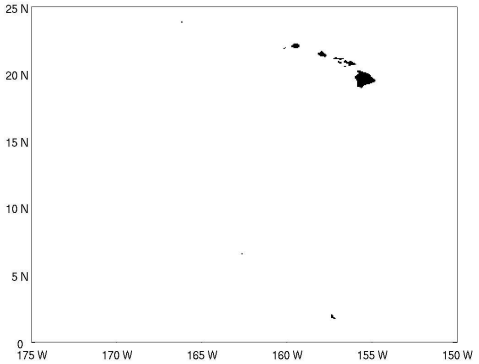
[28]Classical patterns of GSE can be seen in the SW region of the grid for this test case because of the larger region of swell propagation behind the islands. The GSE patterns however are too small to make any significant impact on the results



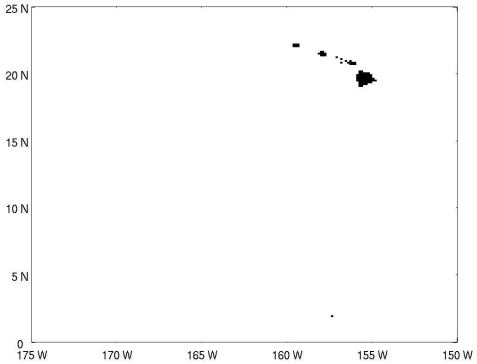
(a) *GSHHS polygons*



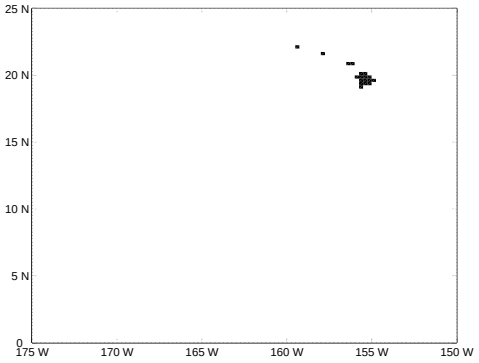
(b) *2' grid*



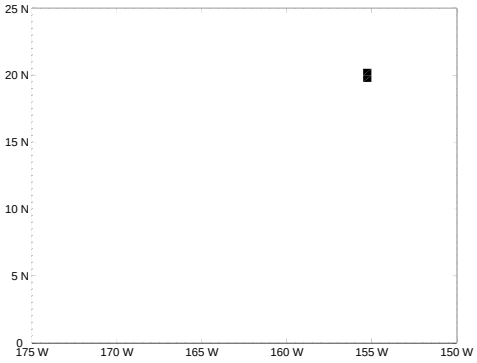
(c) *4' grid*



(d) *8' grid*

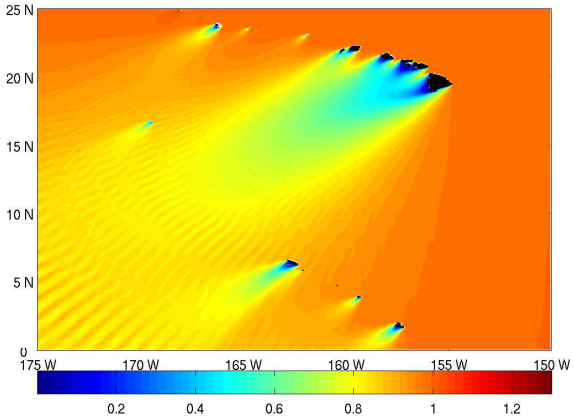


(e) *15' grid*

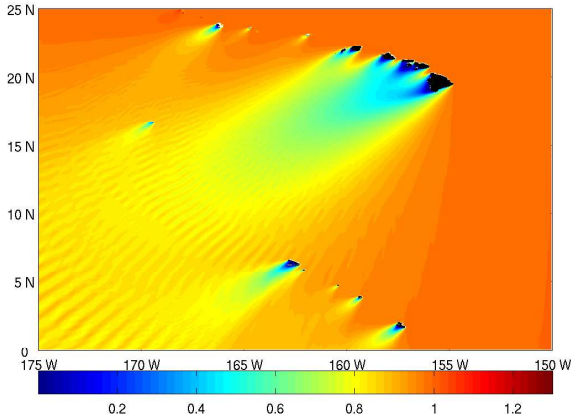


(f) *30' grid*

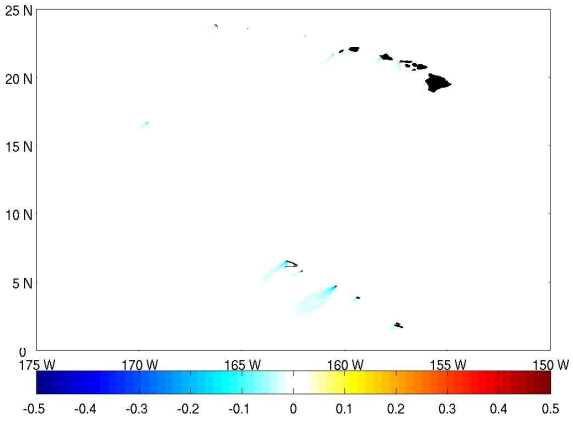
Fig. 3.10 : Land masks for the Hawaii test case



(a) *Test NO-OB*

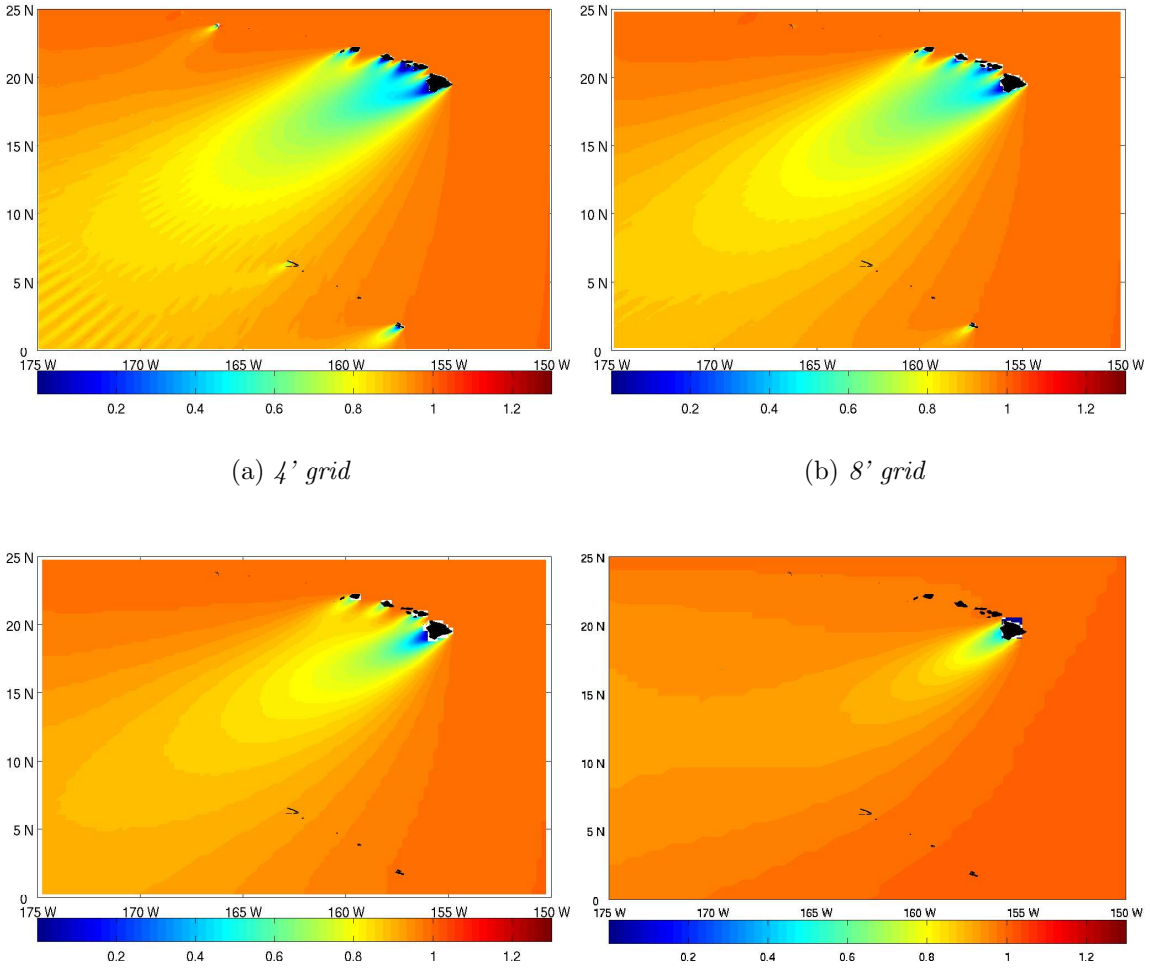


(b) *Test OB-2*



(c) *Difference between the two (with obstruction - no obstruction)*

Fig. 3.11 : Normalized Significant wave heights for 2' grid



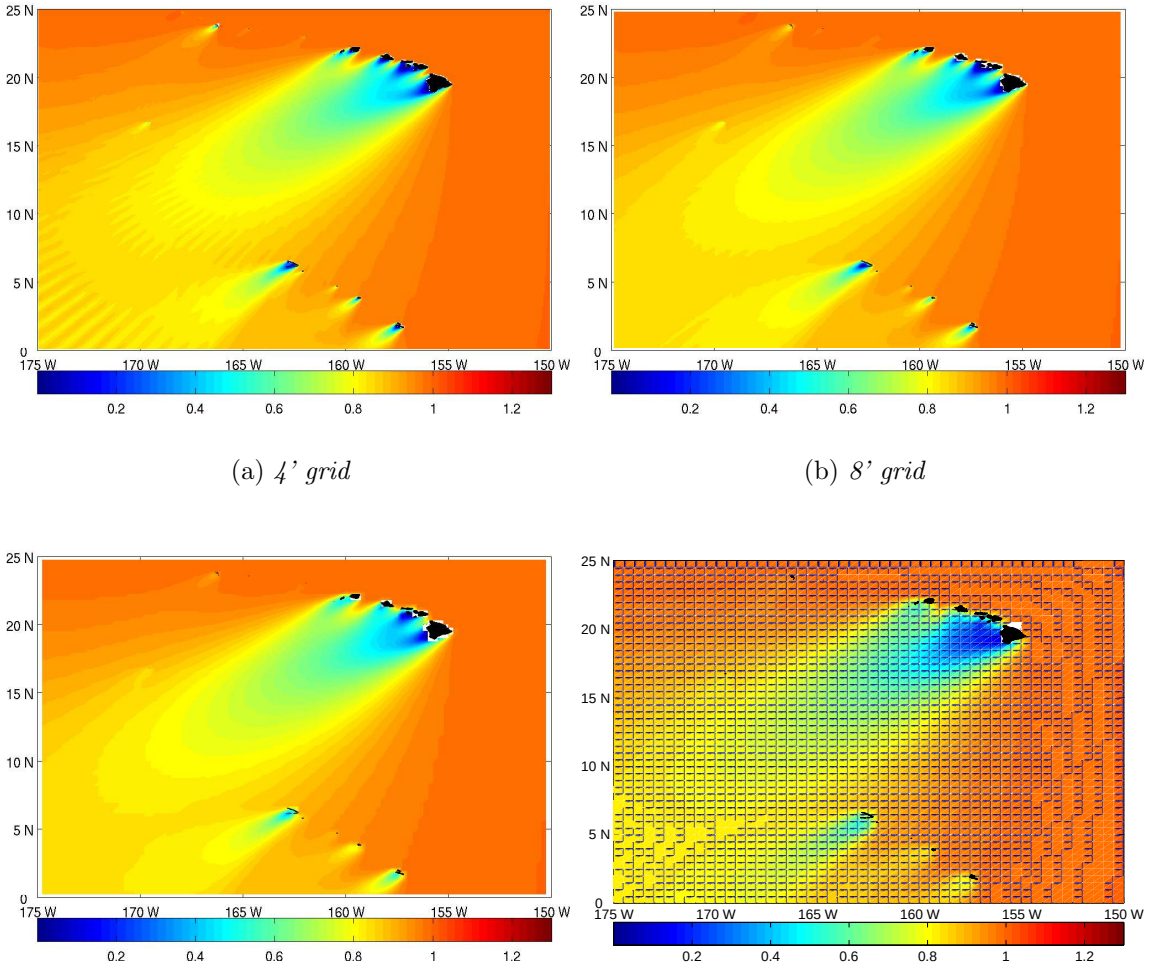
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.12 : Normalized Significant wave heights for Test NO-OB



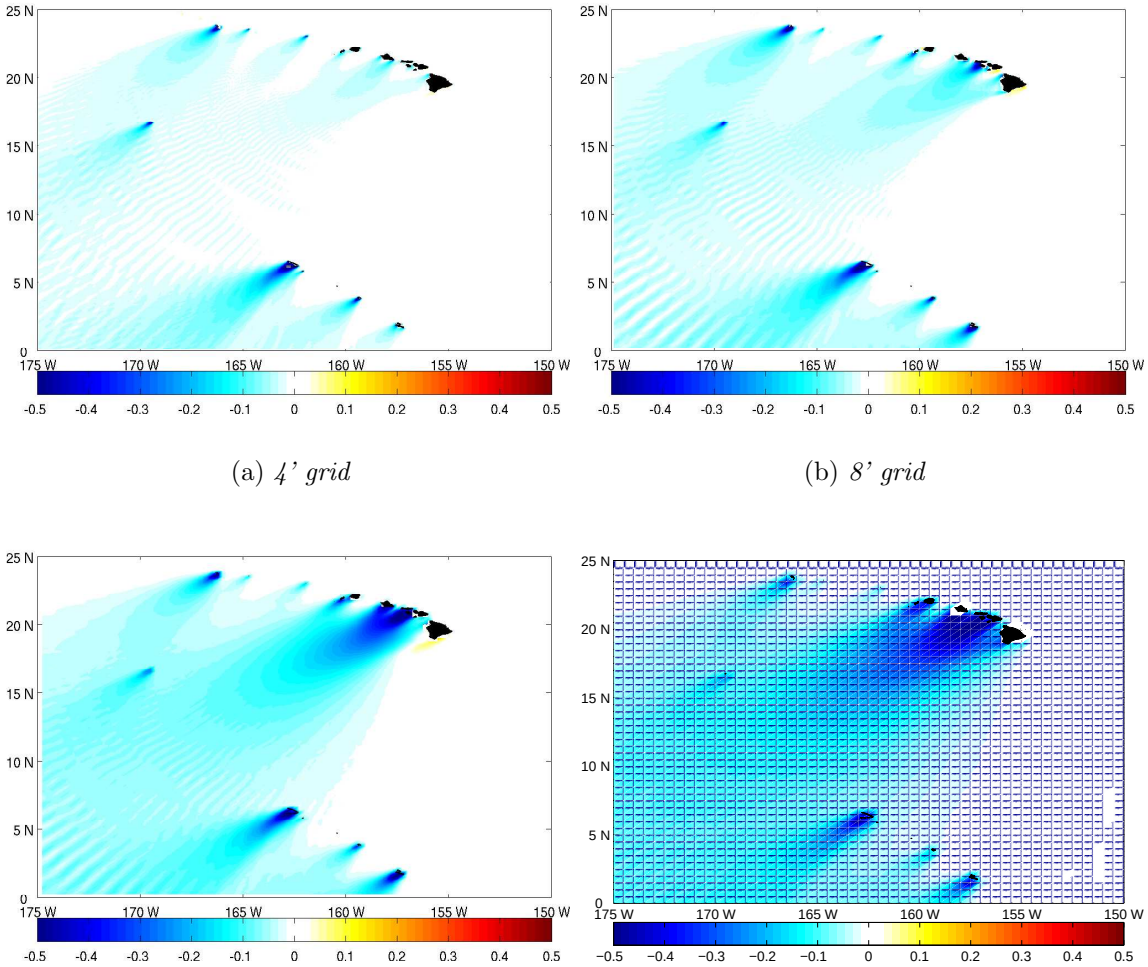
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.13 : Normalized Significant wave heights for Test OB-2



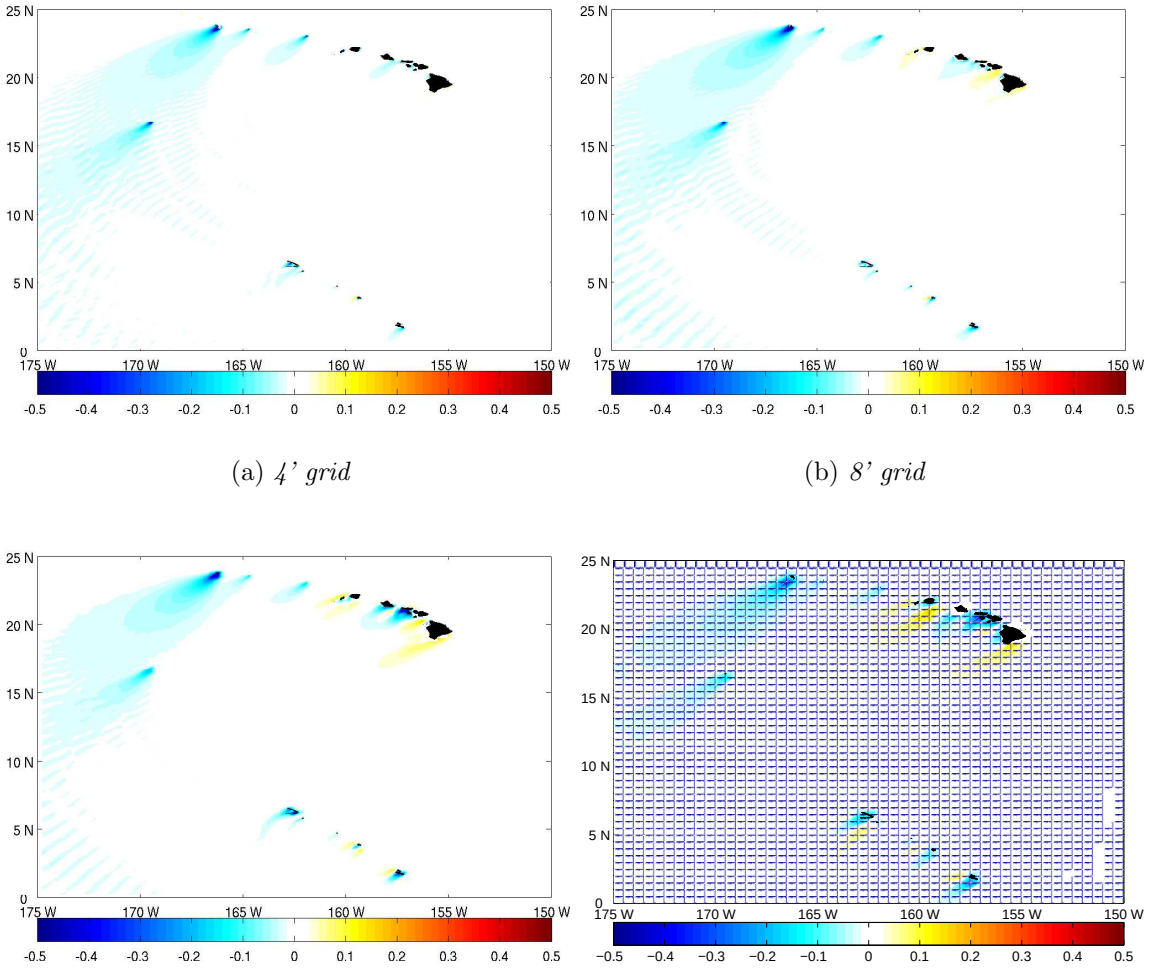
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.14 : Differences between 2' and lower resolution grids (Test NO-OB)



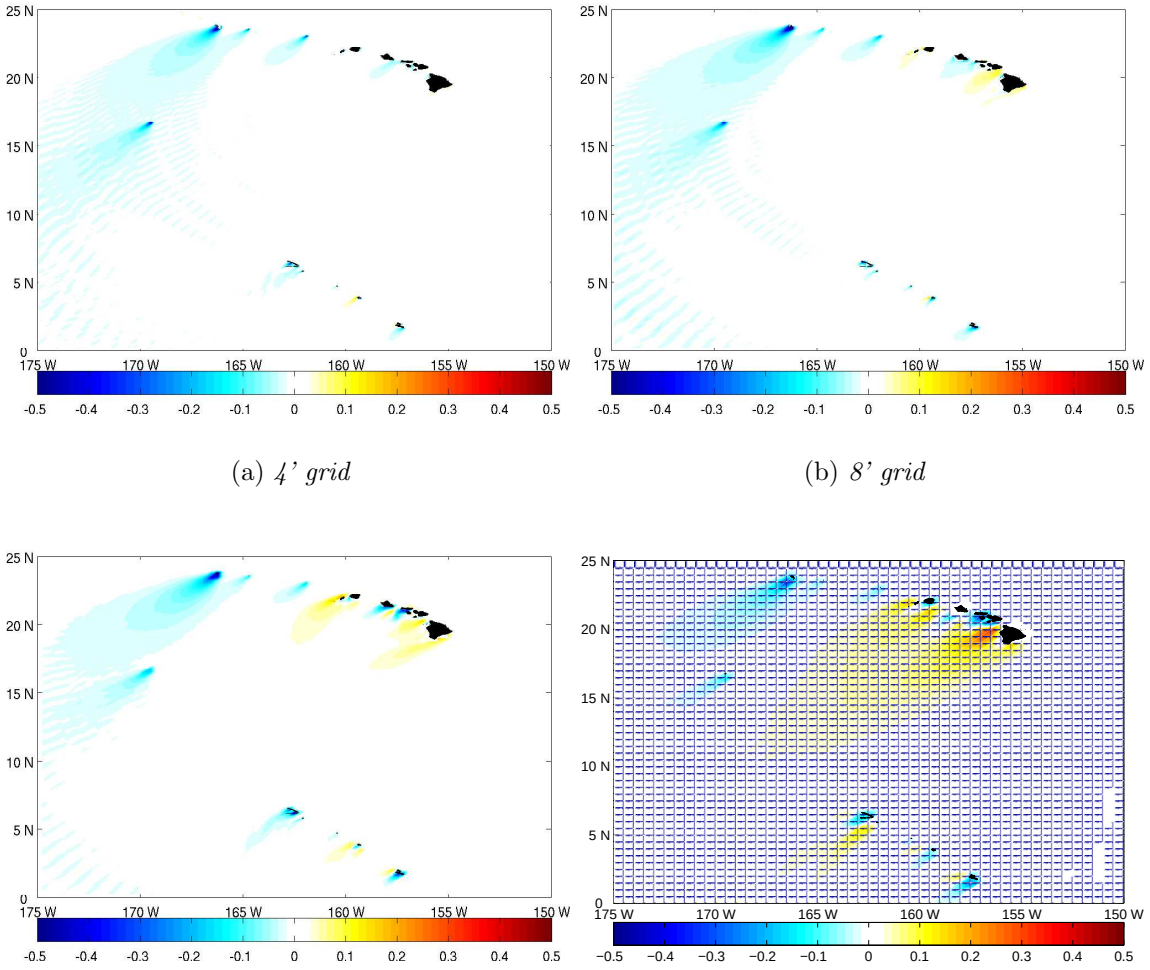
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.15 : Differences between 2' and lower resolution grids (Test OB-0)



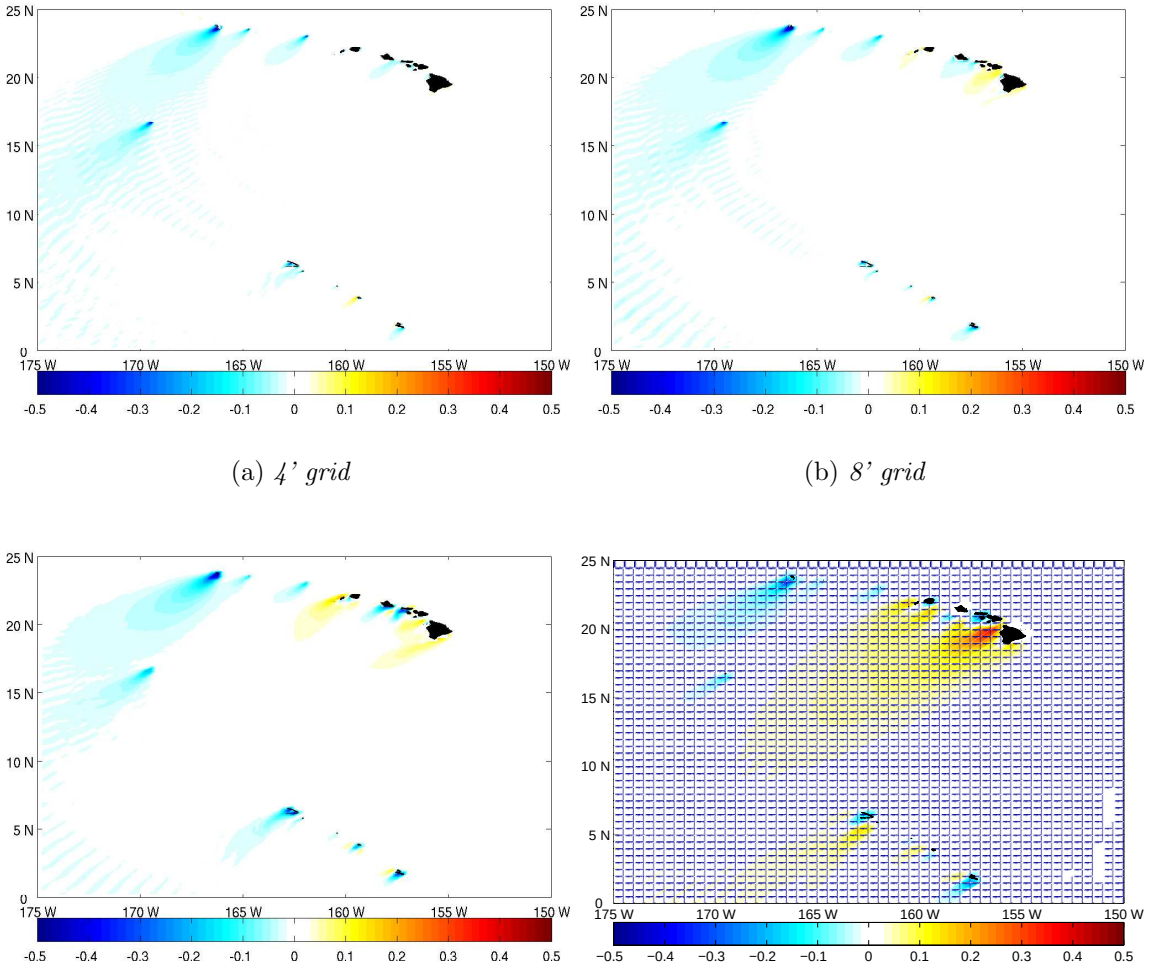
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.16 : Differences between 2' and lower resolution grids (Test OB-1)



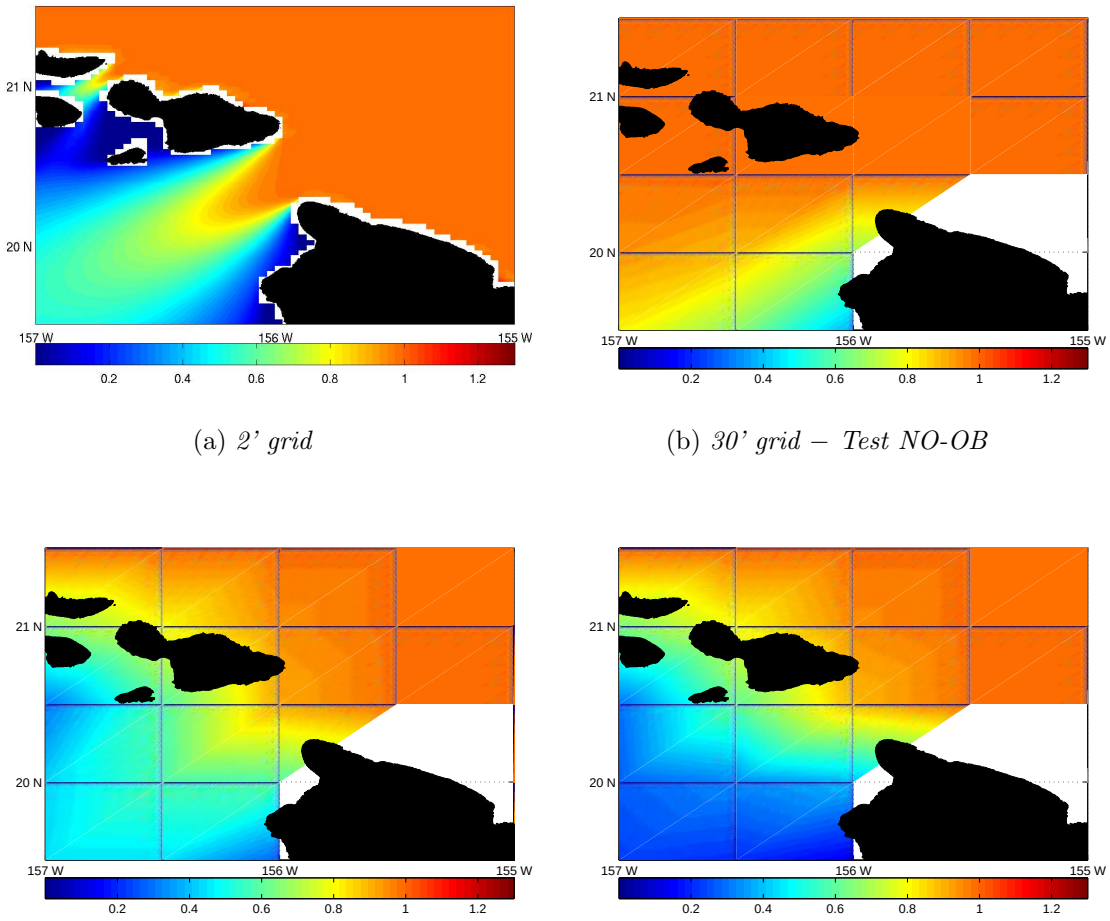
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.17 : Differences between 2' and lower resolution grids (Test OB-2)



(a) *2' grid*

(b) *30' grid – Test NO-OB*

(c) *30' grid – Test OB-0*

(d) *30' grid – Test OB-1*

Fig. 3.18 : Swell propagation through the Alenuihaha Channel (gap between the Island of Hawaii and Maui)

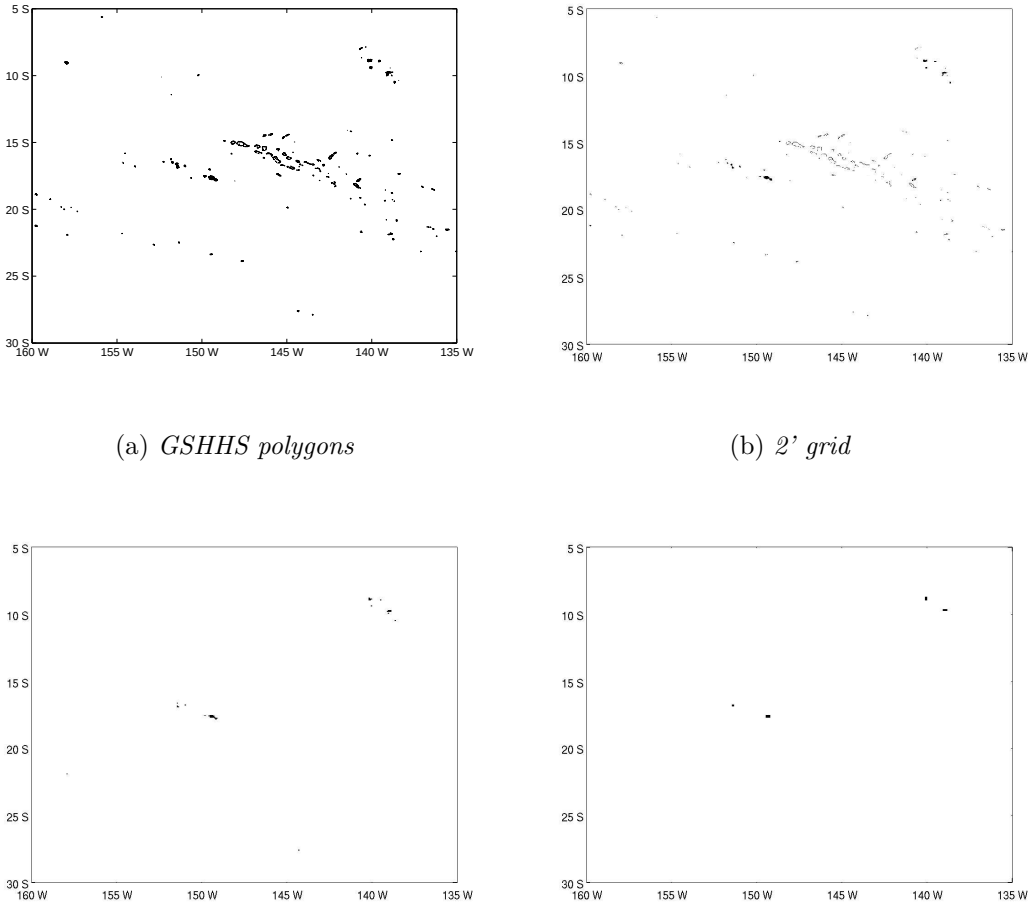
3.3 French Polynesian Islands

The French Polynesian island chains of Marquesas and Tuamotu in the South Pacific (North East of Australia) make up the third and final test case for the grid generation algorithm. The individual islands are mostly atolls that are not well resolved even in the highest resolution grids (Fig 3.19) but still prove to be very effective in blocking swells. Fig 3.19 shows that both the Marquesas (in the North-Eastern part of the grid) and the Tuamotu Archipelago (in the South-Western part of the grid) are poorly resolved at the 4' and 8' resolutions, and not resolved at all in the lower resolutions. Since most of the swell blocking occurs via obstructions this test case provides us with an ideal benchmark to test the obstruction grid algorithm.

Since the islands are spread over a large area this is an ideal place to test what is the impact of accounting for neighboring cell boundary information in the obstruction grids. Fig 3.20 shows the swell propagation at the highest resolution grid. At this resolution, obstruction grids have some impact on swells propagating through the Tuamotu archipelago as some of the atolls here can only be represented as obstructions.

With decreasing grid resolution obstructions become more important (Figs 3.21 and 3.22). The blocking effect of the Tuamotu archipelago cannot be represented even at the 4' grid resolution, and at the 15' or lower resolution the blocking effect of both Marquesas and Tuamotu cannot be represented at all. This test case highlights the importance of having obstruction grids and using **GSHHS** polygons to define the islands. Together the small island chains provide an effective barrier to ocean swells, the effects of which are well reproduced in the obstruction grid algorithms.

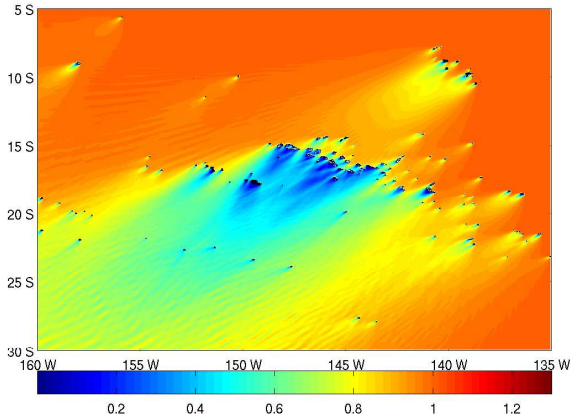
Detailed comparison between the different ways of generating obstruction grids can be seen in the difference plots in Figs 3.23 to 3.26. As expected, having an obstruction grid is more important than the details of the obstruction grid. Since this test case consists of a series of overlapping island chains which work collectively to obstruct the swells accounting for neighboring cell information will be more important here. This can be seen in the weaker blocking of swells through the Tuamotu archipelago in Test OB-0 (Fig 3.24). Taking into account the obstruction information of neighboring cells (Figs 3.25 and 3.26) does a better job in reproducing the energy propagation patterns of the highest resolution run. Though we can see signs of over obstruction when we take into account all the neighbors instead of being selective, particularly at the lower resolution grids. Nonetheless, the impact of taking into account neighbors (as opposed to no neighbors) in the obstruction calculations is greater than how the neighbors are chosen. For this reason we will prefer the methodology of using both neighbors in building obstruction grids as it precludes the need for knowing the direction of swell propagation when applying the obstruction grids.



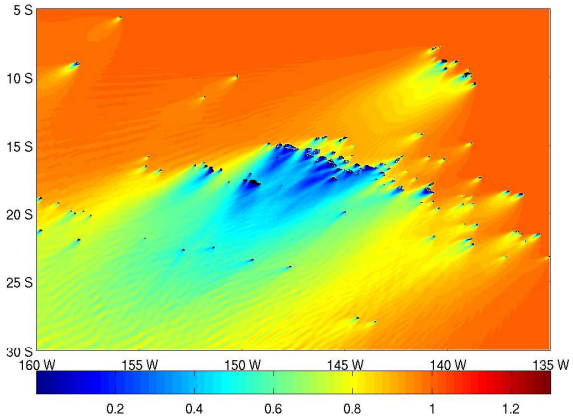
(a) *GSHHS polygons* (b) *2' grid*

(c) *4' grid* (d) *8' grid*

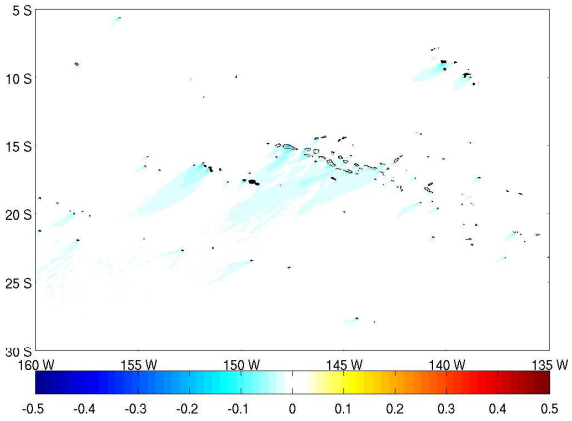
Fig. 3.19 : Land masks for the French Polynesian Islands test case



(a) *Test NO-OB*

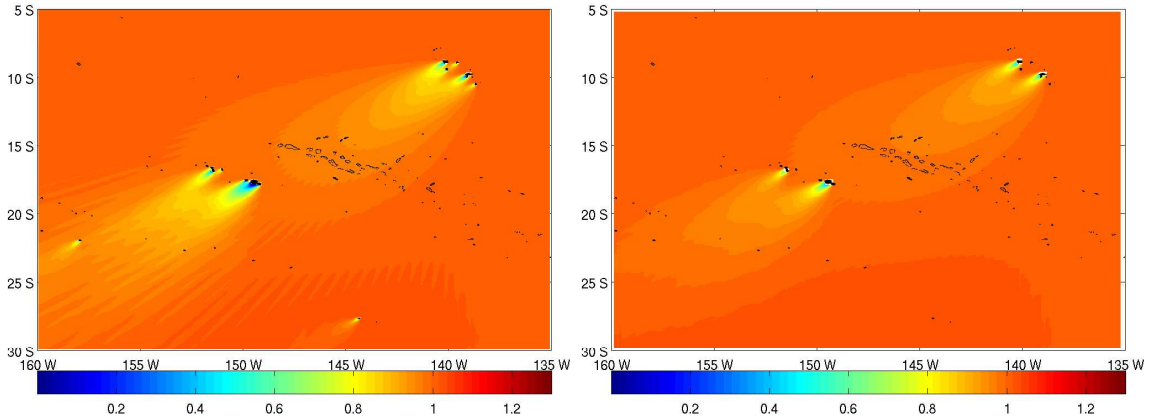


(b) *Test OB-2*



(c) *Difference (with obstruction - without obstruction)*

Fig. 3.20 : Swell propagation through the 2' grid



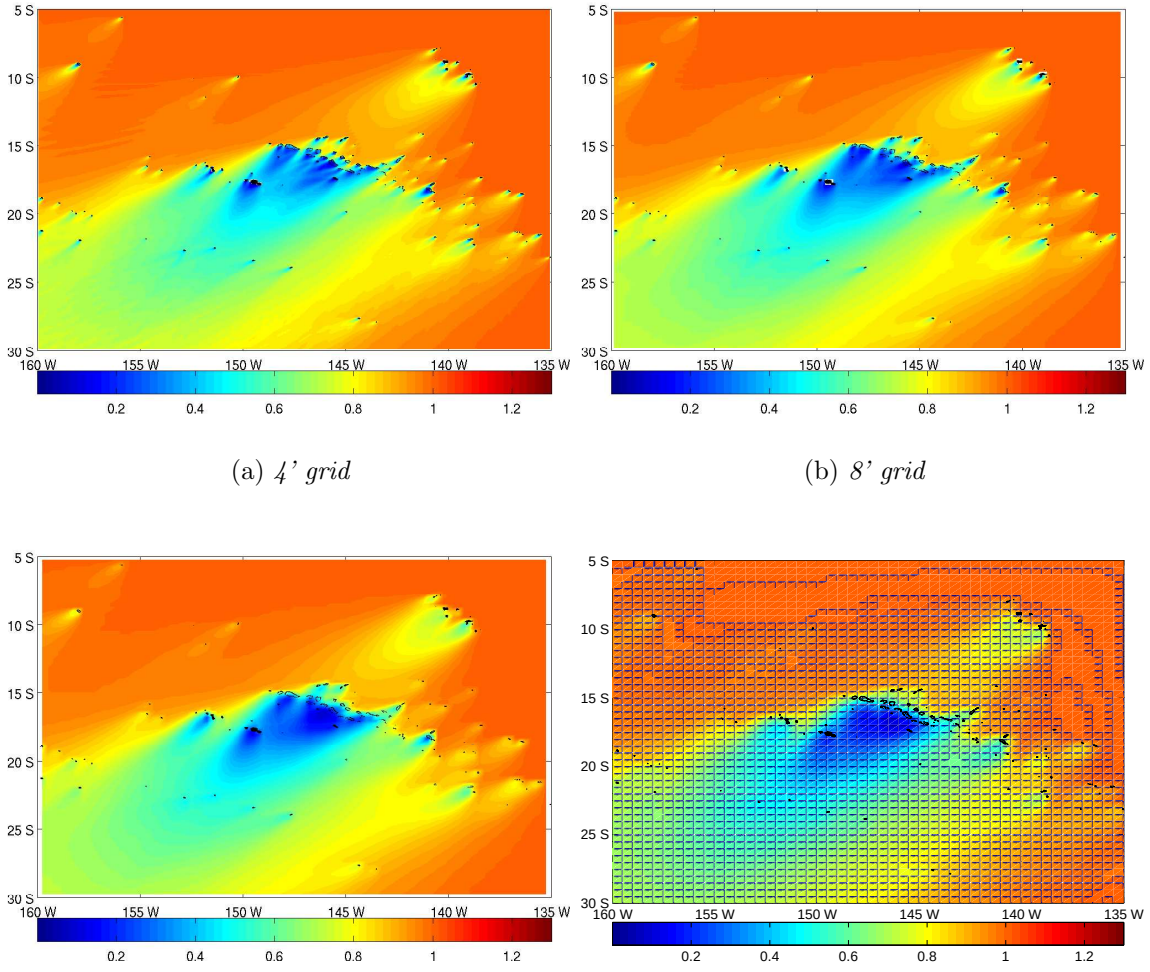
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

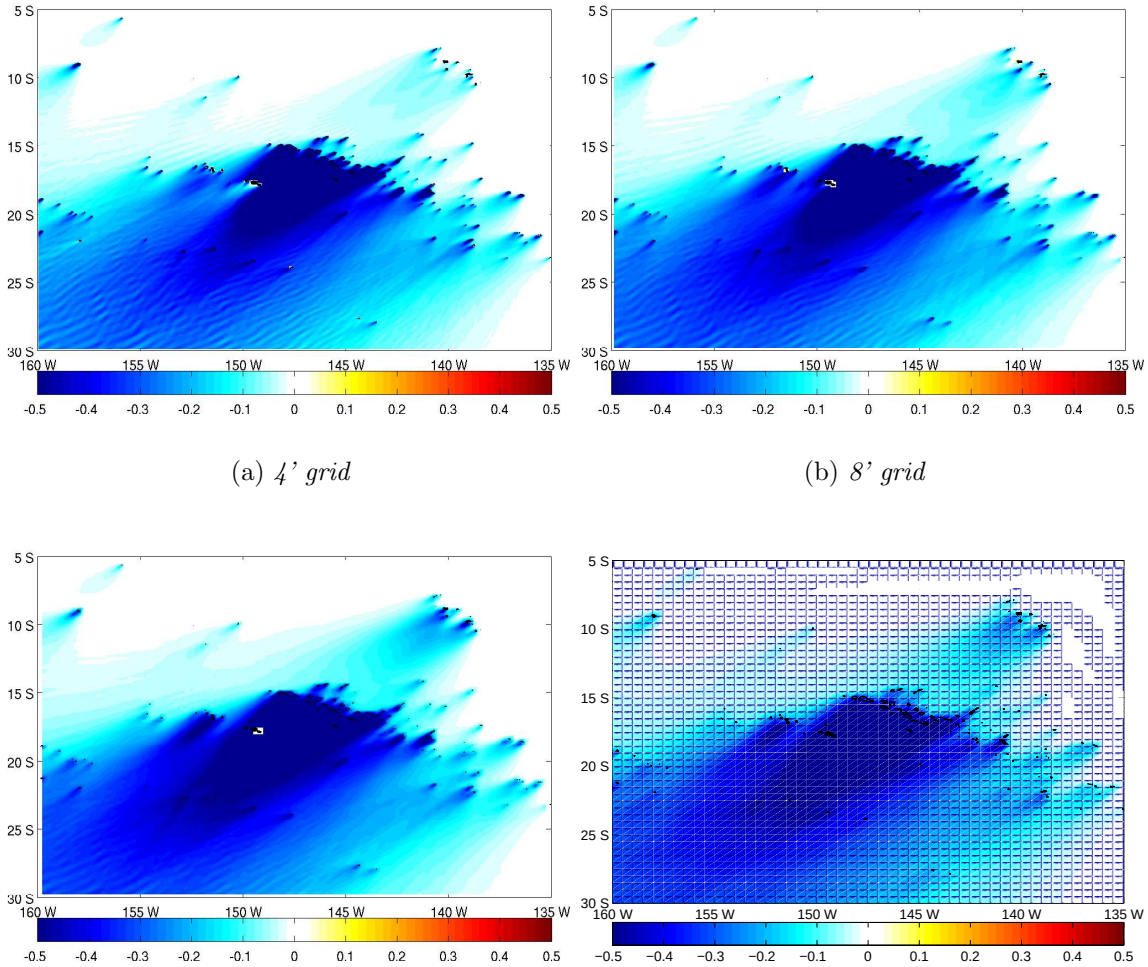
Fig. 3.21 : Normalized Significant wave heights for Test NO-OB



(a) *4' grid* (b) *8' grid*

(c) *15' grid* (d) *30' grid*

Fig. 3.22 : Normalized Significant wave heights with obstruction for Test OB-2



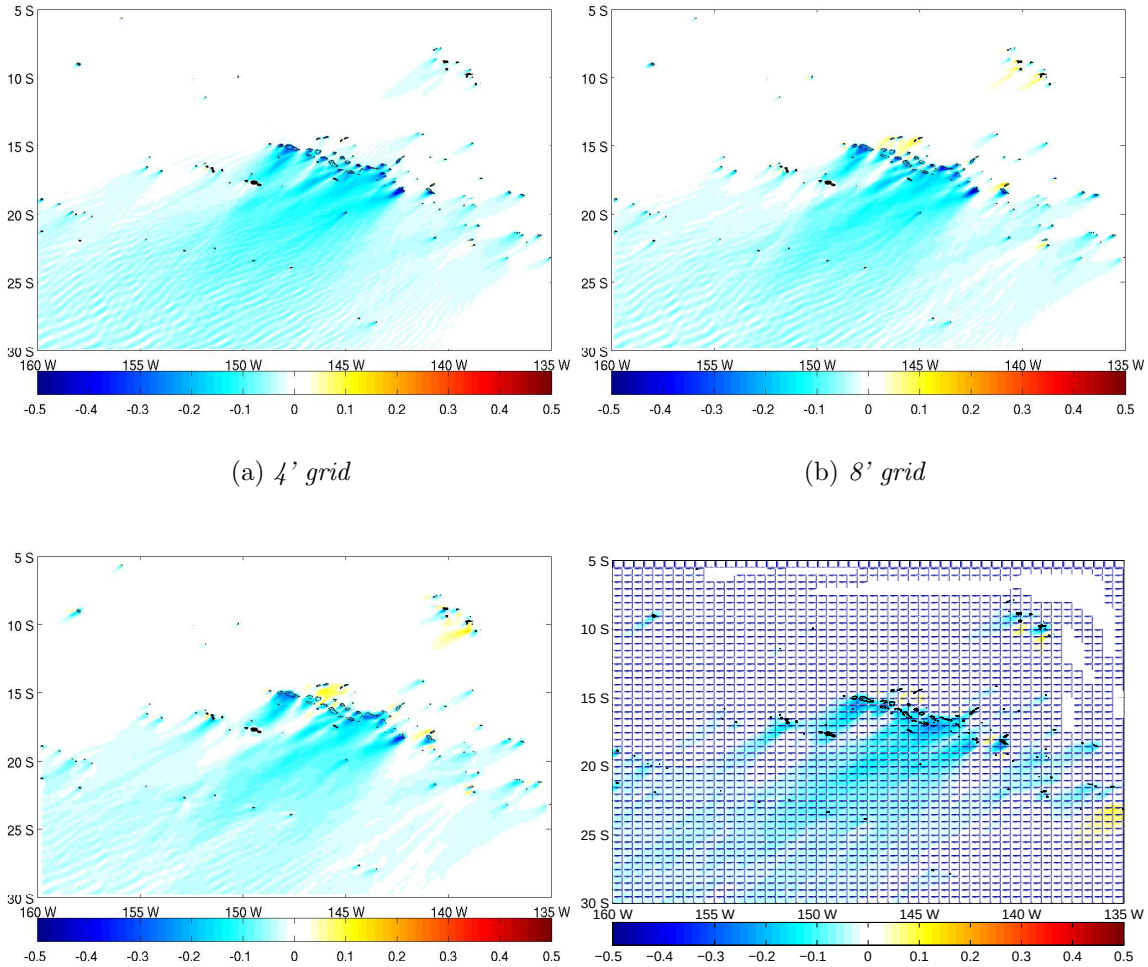
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.23 : Differences between 2' and lower resolution grids (Test NO-OB)



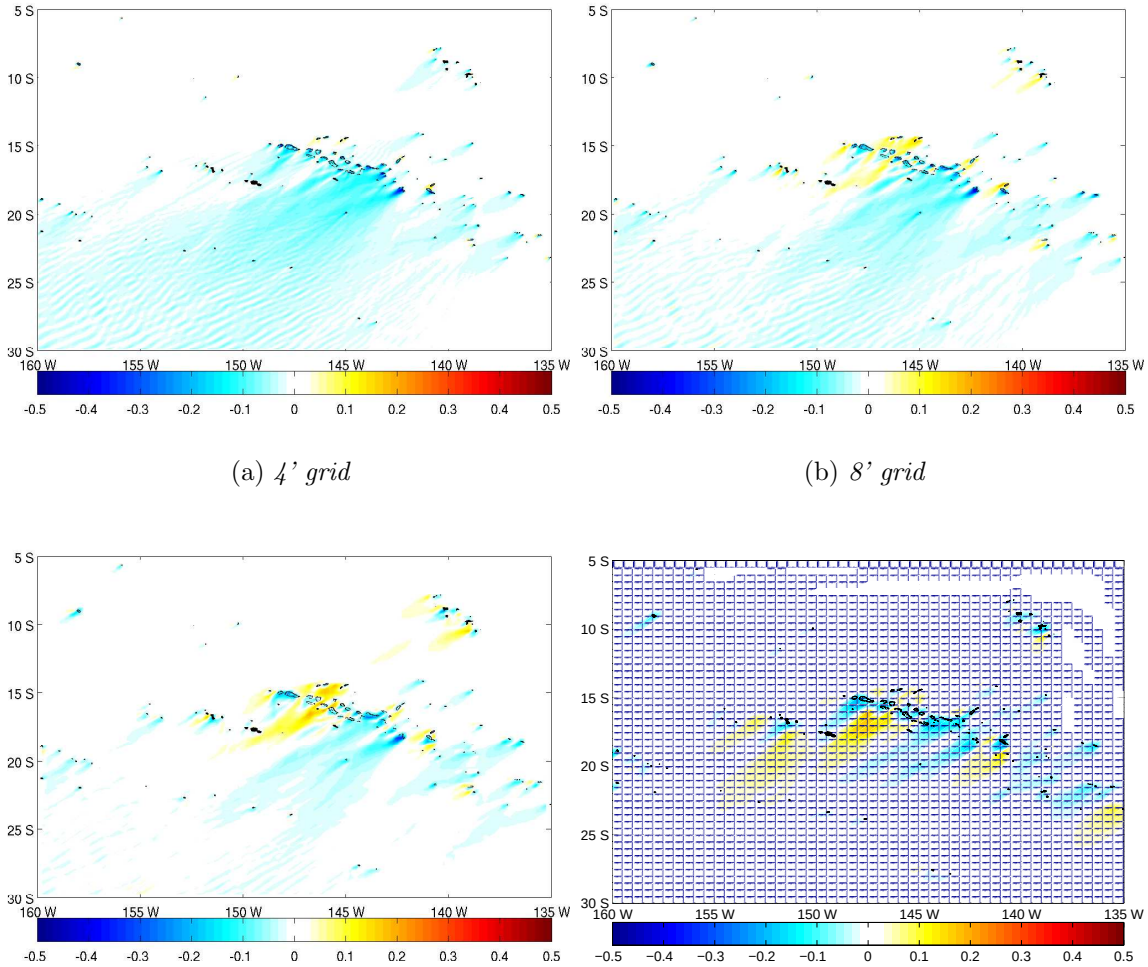
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.24 : Differences between 2' and lower resolution grids (Test OB-0)



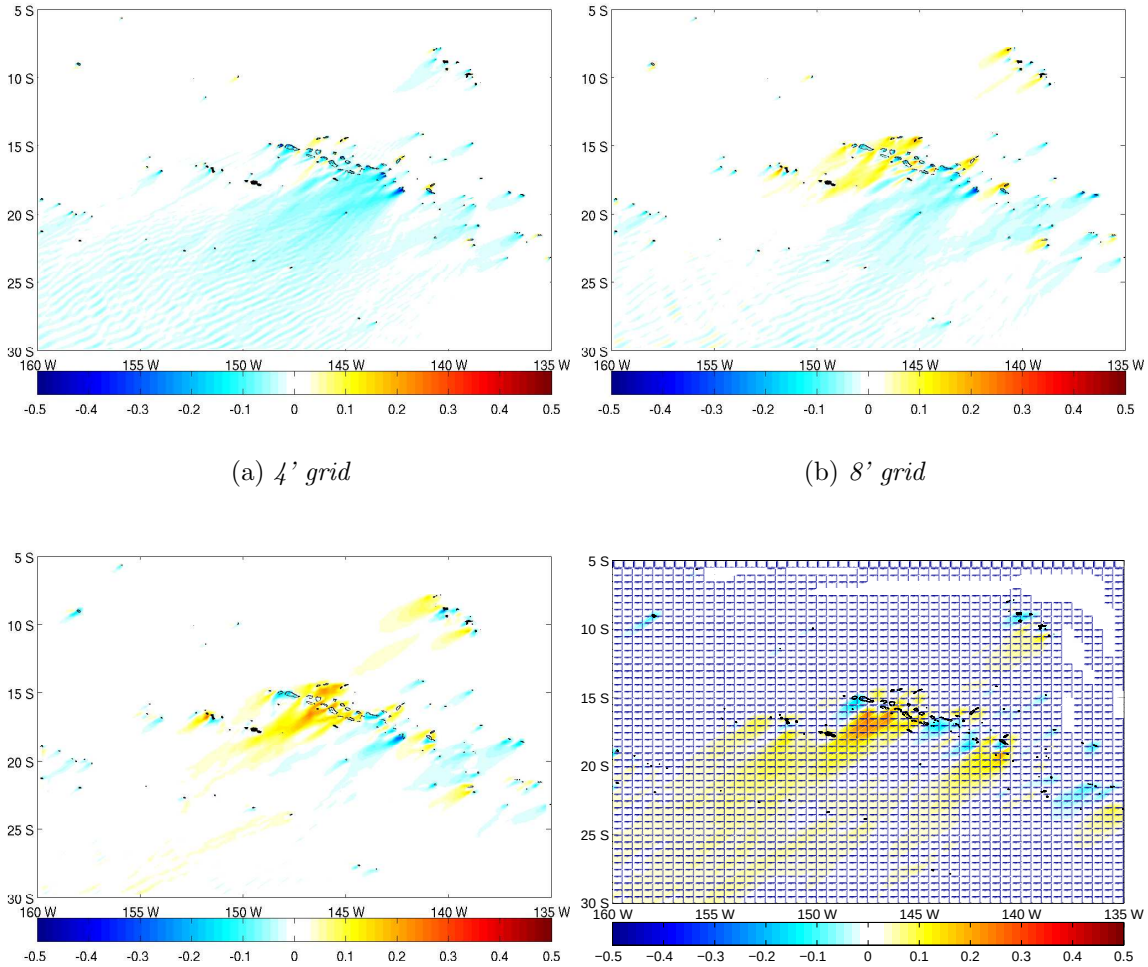
(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.25 : Differences between 2' and lower resolution grids (Test OB-1)



(a) *4' grid*

(b) *8' grid*

(c) *15' grid*

(d) *30' grid*

Fig. 3.26 : Differences between 2' and lower resolution grids (Test OB-2)

4 Conclusions

A grid generation package for use with the WAVEWATCH III wave model has been developed in the MATLAB environment. To allow for as much flexibility as possible the package has been developed as a series of functions that can be called from a master script. This provides the users with the option to customize the package to their own needs, using only the parts that are desired.

The grid generation package consists of 5 main modules − a grid generation module, a boundary module, a land mask module, a wet cell module and a sub - grid module − at the end of which the package provides us with the grid (latitudes, longitudes and depth), land mask and obstruction (along x and y directions at the grid points) information that along with swell parameters and nest information (if applicable) make up the input for WAVEWATCH III simulations. Apart from these there is an additional module for modifying masks. This module is only applicable for the multi-grid version of WAVEWATCH III (version 3.10 or higher) and is needed for grids which get information from over-lapping grids.

The motivation for this work was two fold − first to develop a methodology to accurately account for sub-grid obstructions, and second, to automate the process of grid generation from start to finish (including providing options for masking out unnecessary water bodies). The package uses the 2' global high resolution grid as the reference bathymetric data (choice is between NGDC's **ETOPO2** and NRL's **DBDB2** databases) to build the grids and the Global Self - Consistent Hierarchical High Resolution Shoreline **GSHHS** polygons to build the land-sea masks and obstruction data sets.

The obstruction algorithm developed here has been tested at three different locations (Caribbean, Hawaii and the French Polynesian islands) with grids of several different resolutions. The obstruction algorithm allows for 3 different options − account for obstructions in individual cells alone, account for obstructions in both neighboring cells and finally, account for obstructions only in the cell in the path of the propagating cells. All 3 options were compared with the highest resolution runs.

Overall the patterns of swell propagation at lower resolutions (where significant portions of the coastal features have to be modeled as sub-grid obstructions) compare well with the swell propagation patterns at higher resolution when obstruction grids are taken into account. Accounting for obstructions improved the solution by more than 50 % in comparison to not accounting for obstruction. There were small differences in the results depending upon how the neighboring cell information was handled. In regions with a number of small island chains and atolls (Tuamotu archipelago in the South Pacific and the Bahamas in the Caribbean) accounting for obstruction in the neighboring cells lead to better estimates of the blocking effect. In flows through the channels between large islands

not accounting for neighboring cell information works a little better because the under blocking of swells is compensated by the coarser resolution of the channels. For developing practical forecasts suppression of true swell penetration is more desirable than spurious swell penetration, making OB-2 the preferable option for building obstruction grids. In general though, these differences in wave heights from the three different obstruction options is minimal (10-15 %) in comparison to not accounting for obstruction. Hence underscoring the importance of having a robust obstruction algorithm for practical applications.

References

Booij, N. and L. H. Holthuijsen, 1987: Propagation of ocean waves in discrete spectral wave models. *Journal of Computational Physics*, pp. 307–326.

Chawla, A., 2007: Global bathymetry validation study. Technical Bulletin 253, NCEP/NOAA/NWS, National Center for Environmental Prediction, Washington DC.

NGDC, 2006: 2-minute gridded global relief data (ETOPO2). http://www.ngdc.noaa.gov/mgg/fliers/01mgg04.html.

NRL, 2006: Digital bathymetry data base 2-minute resolution v. 3 (DBB2). http://www7320.nrlssc.navy.mil/DBDB2_WWW/NRLCOM_dbdb2.html.

Tolman, H. L., 2002a: Alleviating the garden sprinkler effect in wind wave models. *Ocean Modelling*, **4**, 269–289.

Tolman, H. L., 2002b: User manual and system documentation of wavewatch-iii version 2.22. Technical note 222, NCEP/NOAA/NWS, National Center for Environmental Prediction, Washington DC.

Tolman, H. L., 2003: Treatment of unresolved islands and ice in wind wave models. *Ocean Modelling*, **5**, 219–231.

Tolman, H. L., 2006: Toward the third release of wavewatch iii: A multi - grid model version. in 9^{th} *International Workshop on Wave Hindcasting and Forecasting*, Victoria, BC, Canada.

Wessel, P. and W. Smith, 1996: A global self-consistent hierarchical high-resolution shoreline database. *J. Geophys. Res.*, **101**(B4), 8741 – 8743.