SoftwareX 9 (2019) 1–6

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

alphaBetaLab: Automatic estimation of subscale transparencies for the Unresolved Obstacles Source Term in ocean wave modelling

Lorenzo Mentaschi [a,*], Michalis Vousdoukas [a,b], Giovanni Besio [c], Luc Feyen [a]

[a] *European Commission, Joint Research Centre (JRC), Directorate for Space, Security and Migration, Italy*
[b] *University of the Aegean, Department of Marine Sciences, Greece*
[c] *Università di Genova, Dipartimento di Ingegneria Chimica, Civile e Ambientale, Italy*

A R T I C L E I N F O

Article history:
Received 7 August 2018
Received in revised form 19 November 2018
Accepted 21 November 2018

Keywords:
Wave modelling
Subscale modelling
Unresolved Obstacles Source Term (UOST)
WAVEWATCH III

A B S T R A C T

The Unresolved Obstacles Source Term (UOST) is a general methodology to parameterize the dissipative effects of subscale islands, cliffs and unresolved coastal features in ocean wave models. It can be applied to any numerical scheme and modulates the dissipation with spectral direction. Its applicability to practical contexts is made possible by the development of the software package alphaBetaLab, which given a mesh and a high-resolution bathymetry is able to automatically estimate the cell-dependent transparency coefficients needed by UOST (Mentaschi et al., 2018). Here we provide the documentation of the package, of its architecture and flow, and a couple of illustrative applications.

Code metadata

Current code version	1.0.0
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX_2018_119
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	Dependencies: Python2.7, shapely=1.5, basemap=1.0.7 netcdf4=1.2 numpy=1.11. See the wiki.
If available Link to developer documentation/manual	https://github.com/menta78/alphaBetaLab/wiki
Support email for questions	lorenzo.mentaschi@ec.europa.eu

1. Motivation and significance

In large-scale modelling of ocean waves, subscale islands, cliffs and features are a major source of local error if neglected. This can affect large portions of the domain [1]. The Unresolved Obstacles Source Term (UOST) is a general approach to parameterize such effects. It comes with advantages on numerical-scheme-based approaches that are usually limited to regular grids and do not consider the spatial/directional layout of the obstructions. The potential of UOST was first shown by Mentaschi et al. [2], who introduced the method and implemented it in the spectral wave model WAVEWATCH III (WW3, [3]). Mentaschi et al. [4] first applied the methodology with real-world bathymetry and input data. Mentaschi et al. [5] showed that on global scale UOST has a positive impact on the model skill compared to other approaches, thanks to the modulation of wave dissipation with spectral direction.

For each partially obstructed cell of the mesh, UOST needs several parameters, such as the direction-dependent transparency coefficients of the cell. Therefore, it was necessary to develop a software (the alphaBetaLab package) for the automatic estimation of such parameters from meshes and bathymetries. This manuscript offers a brief overview of the UOST methodology (Section 2), discusses the flow and architecture of alphaBetaLab (Section 3), and describes a couple of illustrative examples included in the source code (Section 4). The conclusions are drawn in Section 5.

2. Overview on the Unresolved Obstacles Source Term (UOST)

UOST relies on the hypothesis that any mesh can be considered as a set of polygons, called cells, and that the model estimates the average spectrum of each cell. Given a cell (labelled as A in Fig. 1a and b), UOST estimates for each spectral component the effect of

* Corresponding author.
E-mail address: lorenzo.mentaschi@ec.europa.eu (L. Mentaschi).

https://doi.org/10.1016/j.softx.2018.11.006

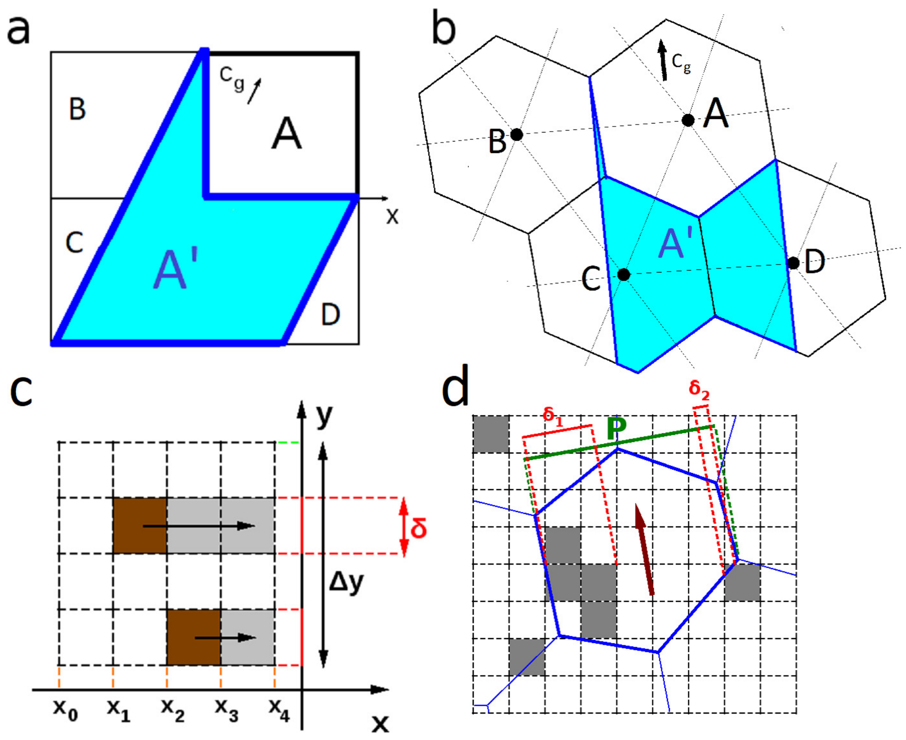

Fig. 1. (a) a square cell (A) and its upstream polygon (A', delimited by blue line, in light blue colour) for a spectral component propagating with group velocity c_g; (b) same as (a) but for a triangular mesh (the hexagons approximate the median dual cells, i.e. the cells of a triangular mesh); (c) computation of the transparency coefficient for a square cell and a spectral component propagating along the x-axis (the brown squares represent unresolved obstacles); and (d) like (c) but for a hexagonal cell and for a tilted spectral component (here the grey squares represent unresolved obstacles).

(a) the unresolved features located in A (local dissipation); (b) the unresolved features located upstream of A and their shadow on A (shadow effect). For the estimation of the shadow, an upstream polygon A' is defined for each cell/spectral component as the intersection between the joint cells neighbouring A (cells B, C and D in Fig. 1ab) and the flux upstream of A. Two different transparency coefficients are needed for each cell/polygon and for each spectral component:

1. The overall transparency coefficient α, computed from the ratio between the cross section of the obstacles and the one of the polygon containing them (Fig. 1c and d). For an obstacle-free cell $\alpha = 1$.
2. A layout-dependent transparency β, defined as the average transparency of cell sections starting from the cell upstream side. If the obstacles are located close to the upstream side of the cell then $\beta \sim \alpha$, if the obstacles are located close to the downstream side of the cell then $\beta \sim 1$.

An important limitation of UOST is that, in order to work properly on a given cell, the time step for the source terms (also called the global time step in WW3) should be less than or equal to the critical Courant–Friedrichs–Lewy (CFL) time step of the cell, i.e., the amount of time needed by the fastest spectral component to entirely cross the cell. Otherwise, part of the energy will leak through the cell without being blocked. For a more detailed description of UOST the reader is referred to [2,4].

3. Software description

alphaBetaLab comes as a Python2.7 library. Its use requires an operating system fully supported by Python. An installation guide valid for unix operating systems is provided in the wiki page. Running alphaBetaLab on large domains can be computationally expensive and the required time can be significantly reduced by running it in parallel on multicore workstations.

In the remainder of this section, we present an overview of the software flow and architecture. To fully understand this topic, some knowledge of object-oriented programming is required. For more detailed information, the reader is referred to the related section in the wiki.

The flowchart of alphaBetaLab is shown in Fig. 2. In synthesis, the input of the system consists of a model mesh and high-resolution bathymetric data. alphaBetaLab assumes that a mesh (represented by an object of class abGrid) can be considered as a set of polygons, called cells. Utility functions create instances of abGrid objects for different types of mesh (e.g. regular, triangular or Spherical Multi Cell – SMC – mesh). The main routine of alphaBetaLab is in the module abEstimateAndSave, which launches the algorithm and saves the output. The algorithm loops on the cells of the mesh and estimates the transparency coefficients for the local dissipation and for the shadow. The output is finally saved to the file format that can be read by the UOST source term implemented in the wave model (see the wiki page for a detailed explanation).

3.1. *High-resolution matrix of transparency*

alphaBetLab needs a high-resolution map of the features (small islands, cliffs, coastal features) that are unresolved in the mesh. This definition is provided through the class abHighResAlphaMatrix. The elements of the matrix are coefficients of transparency to the waves, and have a value between 1 (no obstruction) and 0 (land pixel). At high resolution, the transparency coefficients are assumed to be independent from the wave direction.

Instances of abHighResAlphaMatrix based on ETOPO1 [6] can be built using the function loadBathy found in the module abEtopo1BathyLoader. Future releases will also support the General Bathymetric Chart of the Oceans (GEBCO, [7]) and allow to load multiple Digital Elevation Models (DEM) with different resolutions, giving priority to higher resolution DEMs.

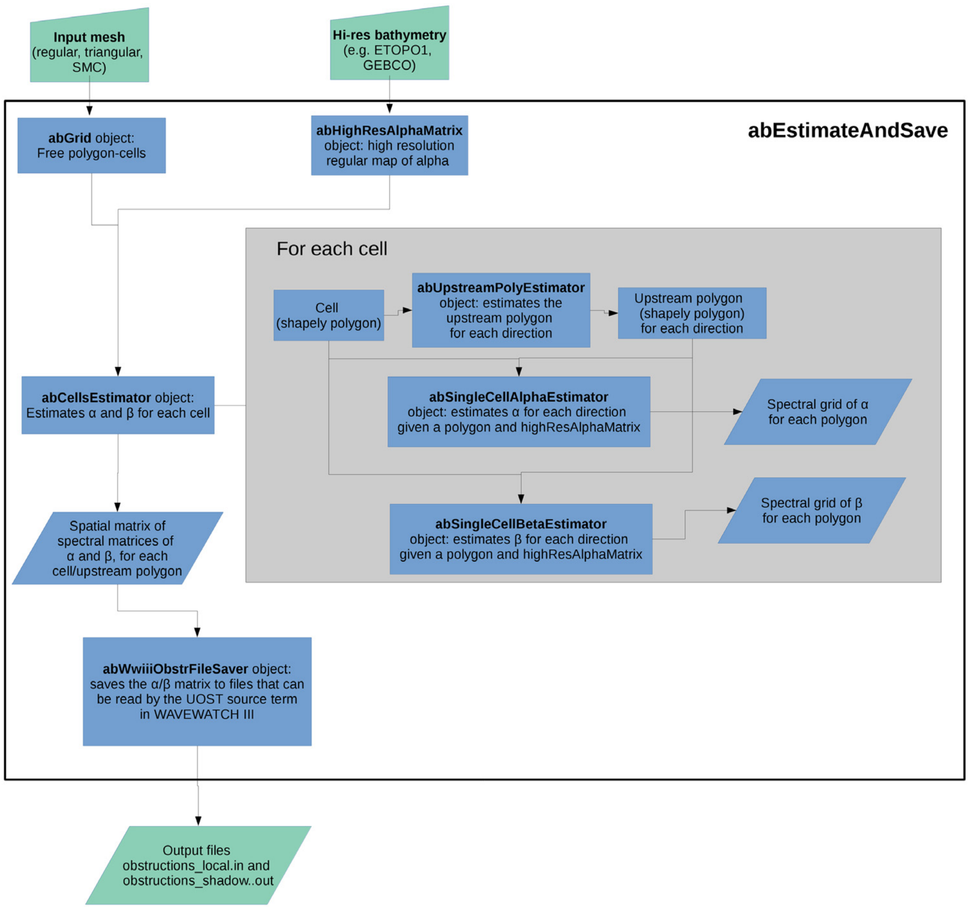
Input mesh (regular, triangular, SMC)
Hi-res bathymetry (e.g. ETOPO1, GEBCO)
abEstimateAndSave
abGrid object: Free polygon-cells
abHighResAlphaMatrix object: high resolution regular map of alpha
For each cell
Cell (shapely polygon)
abUpstreamPolyEstimator object: estimates the upstream polygon for each direction
Upstream polygon (shapely polygon) for each direction
abCellsEstimator object: Estimates α and β for each cell
abSingleCellAlphaEstimator object: estimates α for each direction given a polygon and highResAlphaMatrix
Spectral grid of α for each polygon
abSingleCellBetaEstimator object: estimates β for each direction given a polygon and highResAlphaMatrix
Spectral grid of β for each polygon
Spatial matrix of spectral matrices of α and β, for each cell/upstream polygon
abWwiiiObstrFileSaver object: saves the α/β matrix to files that can be read by the UOST source term in WAVEWATCH III
Output files obstructions_local.in and obstructions_shadow..out

Fig. 2. Flowchart of alphaBetaLab.

3.2. *Model mesh definition*

The class _abGrid, in module abGrid, defines a model mesh as a collection of polygons, each representing a cell. For the definition of the polygon objects, alphaBetaLab relies on the library shapely. _abGrid objects have knowledge of the position of each cell and of its neighbours, enabling a fast estimation of the upstream polygon for each wave direction. Only the cells containing unresolved obstacles are relevant to the algorithms implemented in alphaBetaLab, while empty and land cells are excluded from _abGrid instances. Furthermore, in cells intersecting large land bodies UOST would usually conflict with the resolved portion of coastal dynamics. Therefore, also the coastal cells are generally excluded from the instances of _abGrid (the logics of exclusion of coastal cells are defined in class abCoastalCellDetector, see the wiki for more details).

The instances of _abGrid for specific types of mesh are generated by specialized utility classes and functions: (a) abRectangularGridBuilder for regular grids; (b) abTriangularGridBuilder for triangular meshes, which supports the input formats gmesh (used by WW3 and by Wind Wave Model — WWM [8]) and gr3 (used by the hydraulic model SCHISM [9]); and (c) abSMCGridBuilder for Spherical Multi-Cell (SMC) meshes [10,11], which is still under development.

3.3. *Upstream polygon*

To be able to estimate the shadow effect, an upstream polygon needs to be computed for each cell-polygon and spectral component. The upstream polygon of a cell A is defined as the intersection between the joint cells neighbouring A and the flux of the spectral component upstream of A. In alphaBetaLab this is computed by function getUpstreamPoly of the module abUpstreamPolyEstimator. This function takes as arguments a cell-polygon, a neighbourhood polygon and a direction. The neighbourhood polygon is estimated within the class _abGrid and is given by the union of the neighbour polygons. It can be retrieved with the method _abGrid.getNeighbors.

3.4. *Estimation of α and β in a single polygon*

For each single polygon (that can be a cell or an upstream polygon) the value of the directional α transparency coefficients are computed by instances of the class abSingleCellAlphaEstimator.

The method computeAlpha computes α for a given spectral component as a function of the cross section of the unresolved features inside the polygon along the propagation direction. Inside this function, the problem is first rotated and reflected to transform it to the first octant (class abFirstOctantTransformation in module abFirstOctantTransformation). Then the computation is done on the first octant (private method _getAlphaFirstOctant of class abSingleCellAlphaEstimator).

The computation of the β coefficients is executed by abSingleCellBetaEstimator objects in the method computeBeta. Similar to the method computeAlpha of abSingleCellAlphaEstimator, the problem is transformed to the first octant before the actual computation. The algorithm implemented in _getBetaFirstOctant follows the definition of β: estimate n polygon slices starting from the upstream side of the cell and then for each of these slices estimate the α coefficient. The value of β is then given by the average of the

α coefficients over the slices [2,4]. This approach provides accurate estimations of β but is computationally expensive. A future, more optimized implementation will start from the value of α and will change it as a function of the distribution of the obstacles in the polygon. If the distribution mean lies close to the upstream side of the cell then $\beta \sim \alpha$. If it lies at the downstream side then $\beta \sim 1$. If the distribution mean lies at a certain position between the two sides then β will be increased with respect to α by the corresponding fraction.

3.5. Cell sizes

UOST needs the sizes of the cell-polygon along the propagation directions of each spectral component. These distances (that must be estimated in meters) are computed within the class abCellSize. The algorithm finds the main axes of the cell-polygon and approximates it as an ellipse.

3.6. Assembling all together

Given an instance of _abGrid and one of abHighResAlphaMatrix, the computation of α and β for all the cells/upstream polygons/directions is performed within the class abCellsEstimator.

To speed up the computation a few directional bins are considered (8 by default) and the estimated α and β are subsequently interpolated to the directions of the spectral grid.

To attain better performances alphaBetaLab is extensively parallelized using the Python multiprocessing approach that can be successfully employed in shared memory environments.

3.7. Saving the output

The output of the algorithms, described in the previous sections, consists of two sets of α and β coefficients, one for the local dissipation and one for the shadow effect. These two sets are saved to two separate files (class abWwiiiObstrFileSaver, method saveFiles) in the format that can be loaded by the UOST module of WW3. The default names of the output files are obstructions_local.<gridname>.in and obstructions_shadow.<gridname>.in. For a description of the format of the output files the reader is referred to the wiki and to the examples provided in the source repository.

3.8. End user commands

Wave modellers should not handle directly the modules/classes/algorithms described in the previous sections. Rather, they should invoke simple functions with simple parameters that accomplish all the requested activities. This end-user Application Programming Interface (API) is implemented in the module abEstimateAndSave and includes the possibility to pass parameters to the different components of the algorithm (module abOptionManager, see the wiki for a list of available parameters and their default values). In Section 4 two examples showing the use of such API are illustrated.

3.9. Shortcomings

A practical issue of alphaBetaLab when working on large domains is that it is rather slow. A development that could significantly boost the execution speed is the optimization of the computation of β, as explained in Section 3.4.

Another shortcoming is that in presence of unresolved breakwater intersecting several consecutive cells, the approach implemented in abSingleCellAlphaEstimator and abSingleCellBetaEstimator based on the estimation of the cross section of the obstacle in each single cell is not enough. In such cases abSingleCellAlphaEstimator would detect a total block only for spectral components normal to the breakwater, while if all the cells crossed by the breakwater are considered, the block should be total for all the spectral components propagating from one side of the breakwater to the other. Tackling this problem becomes important at high resolution. A prototype of an algorithm to deal with this problem is developed in module abLongBreakWaterLocAlphaAdjust. However, it was not applied in the case study analysed by [4] and will require further testing before being available for use. Moreover, it is unoptimized and not parallelized.

Finally, the classes/functions handling the SMC meshes are still under development. Addressing this point will be necessary to fully support all the numerical schemes available in WW3.

4. Illustrative examples

The examples below are available in the project repository and can be used as a guideline on how to prepare simple Python scripts that launch alphaBetaLab.

To run the illustrative examples, the user must have downloaded alphaBetaLab (let us assume to the directory /src/alphaBetaLab/) and installed it in an instance of Python (which we assume can be called with the command ablPython). For instructions on how to install alphaBetaLab the reader is referred to the wiki.

4.1. A global regular domain

In this example alphaBetaLab and WW3/UOST are run on a global regular domain at a resolution of 1.5°.

The script to generate the transparency coefficients needed by UOST for this example is

/src/alphaBetaLab/examples/ww3/regularMesh/bathyAndObstructionsGlobal/obstFileBuilder.py.

A line-by-line explanation of the script is provided in the wiki. After a set of instructions that generate the input of the algorithm (the spectral grid, the specifications of the regular grid, the path of the bathymetric file, the output directory, the number of cores for the parallelization, an optional set of algorithm parameters), the core of the script is the last instruction:

abEstimateAndSaveRegularEtopo1(dirs, freqs, gridname, regularGridSpec, etopoFilePath, outputDestDir, nParWorker, opt)

This launches the computation and saves the output.

To generate the input files for UOST the user should use the console commands:

```
$ cd /src/alphaBetaLab/examples/ww3/regularMesh/
bathyAndObstructionsGlobal/
```

```
$ ablPython obstFileBuilder.py.
```

The run can take some time. A faster example can be found in directory
/src/alphaBetaLab/examples/ww3/regularMesh/
bathyAndObstructionsCaribbean, which estimates α and β on a subset (the Caribbean Sea) of the same domain.

The two output files are located in
/src/alphaBetaLab/examples/ww3/regularMesh/
bathyAndObstructionsGlobal/output.

UOST is available in the official version 6.x of WW3. To enable it, the user has to enable the UOST switch before the compilation (see the manual of WW3 for details).

An example of WW3 setup using the obstruction files generated with this illustrative example can be found in /src/alphaBetaLab/examples/ww3/regularMesh/wwiiiRunCFSR. The script to launch the simulation is run_multi_restarts.sh, in which the following variables need to be modified:

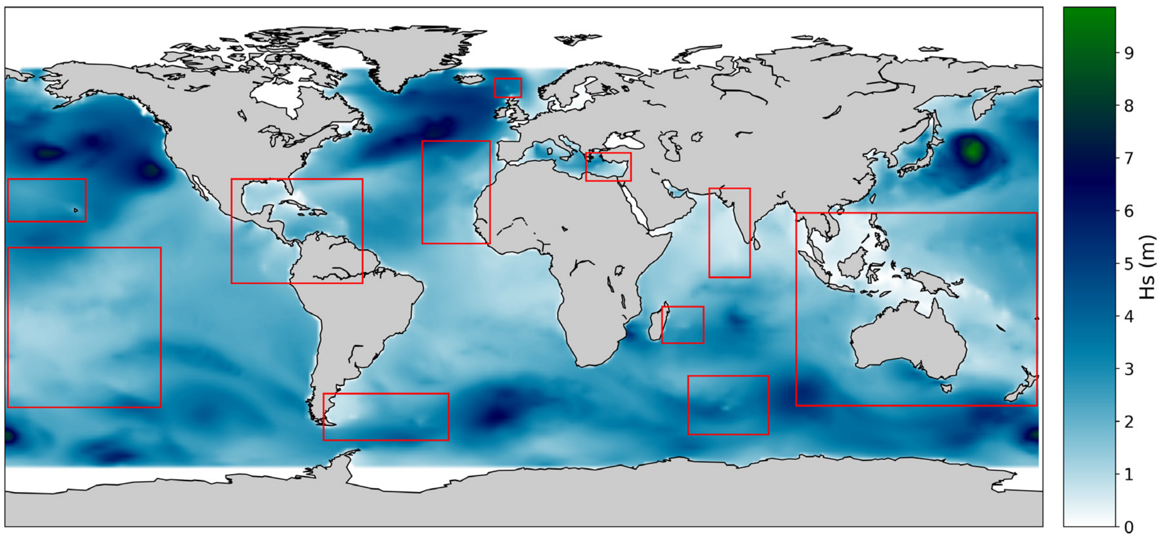

Fig. 3. Significant wave height simulated in the illustrative case on a global regular grid for February 21st 2000, 3:00. The red rectangles highlight some of the areas where the sheltering effect of the unresolved small islands is clear.

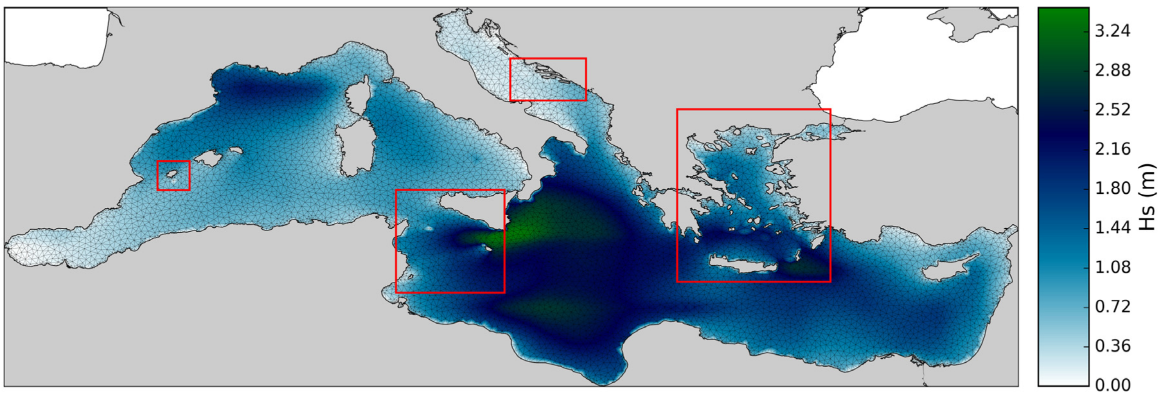

Fig. 4. Significant wave height simulated in the illustrative case on a Mediterranean triangular mesh for January 15th 2000, 13:00. The red rectangles highlight some areas where the sheltering effect of the unresolved small islands is clear.

1. the variable exepath must be set to the directory containing the ww3 executables (e.g. /myww3/v6.x/exe).
2. nParallelProc should be set to the number of desired parallel processes.
3. mpicmd must be set to the mpirun command (e.g. /mympi/bin/mpirun). If nParallelProc is 1 this variable is not required.

This simulation by default runs for the months of January and February 2000, and saves the output as netcdf in the same directory. The significant wave height (Hs) simulated for February 21st 2000 is shown in Fig. 3. In many areas (highlighted with red rectangles), the dissipation due to unresolved islands parameterized by UOST is clearly visible, as a decrease of Hs.

4.2. A triangular mesh on the Mediterranean Sea

In this example, alphaBetaLab and WW3/UOST are run on a triangular mesh covering the Mediterranean Sea with a resolution variable between 20 km and 2 km.

The sample script that generates the obstruction files is /src/alphaBetaLab/examples/ww3/regularMesh/bathyAndObstructions/obstFileBuilder.py. This script is similar to the one described in the previous example and a line-by-line explanation can be found in the wiki.

The main instructions of the script are:

triMeshSpec = triMeshSpecFromMshFile('med.msh')

which instantiates a specification object for the triangular mesh loaded from file med.msh. The subsequent line

abEstimateAndSaveTriangularEtopo1(...)

performs the elaboration and saves the output. In this case, the cells smaller than 3 km are skipped by alphaBetaLab due to the parameter minSizeKm = 3. Such small cells would require a too small global time step in WW3 to be adequately resolved and would slow down the computation of alphaBetaLab.

A setup of WW3 to run this case study can be found in /src/alphaBetaLab/examples/ww3/triangularMesh/wwiiiRunCFSR and is similar to the one of the previous example. A map of Hs simulated for January 15th 2000 is shown in Fig. 4. In the areas highlighted with red rectangles, the unresolved islands parameterized by UOST have a clear sheltering effect.

5. Impact and conclusions

alphaBetaLab allows the employment of UOST in real-world wave modelling applications with clear advantages for the wave community. Considering the directionality of the unresolved features and their geometrical layout inside each cell, UOST has a positive impact on the model skill both in terms of mean wave parameters and the wave spectrum, with mean results close to the ones of higher resolution models [4,12]. The independence of UOST from the numerical scheme for energy propagation will allow the parameterization of the unresolved obstacles with any mesh supported by alphaBetaLab. The availability of UOST will simplify the generation process of unstructured meshes (refinement at any small island will not be needed) and improve their computational economy (the flux of waves in complex areas like archipelagos will be satisfactorily parameterized at a much lower resolution). Therefore, it will impact positively on the use of flexible meshes in large-scale contexts.

Acknowledgements

The research leading to these results has received funding from the JRC of the European Commission as part of the COAST and ADAPTATION projects.

References

[1] Tolman HL. Treatment of unresolved islands and ice in wind wave models. Ocean Model. 2003;5:219–31.
[2] Mentaschi L, Pérez J, Besio G, Mendez FJ, Menendez M. Parameterization of unresolved obstacles in wave modelling: A source term approach. Ocean Model. 2015;96:93–102.
[3] The WAVEWATCH III Development Group (WW3DG), User manual and system documentation of WAVEWATCH III version 516 Tech. Note 329, NOAA/NWS/NCEP/MMAB, College Park, MD, USA Tech. Note, MMAB Contrib., 2016.
[4] Mentaschi L, Kakoulaki G, Vousdoukas M, Voukouvalas E, Feyen L, Besio G. Parameterizing unresolved obstacles with source terms in wave modeling: A real-world application. Ocean Model. 2018;126:77–84.
[5] Mentaschi L, Kakoulaki G, Vousdoukas MI, Voukouvalas E, Feyen L, Pereira P. The effect of changing spatial resolution in global dynamic wave models, ESSOAr. 2018, http://dx.doi.org/10.1002/essoar.10500014.1.
[6] Amante C, Eakins BW. ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis. 2009.
[7] Becker JJ, Sandwell DT, Smith WHF, Braud J, Binder B, Depner J, Fabre D, Factor J, Ingalls S, Kim SH, Ladner R, Marks K, Nelson S, Pharaoh A, Trimmer R, von Rosenberg J, Wallace G, Weatherall P. Global bathymetry and elevation data at 30 Arc seconds resolution: SRTM30_PLUS. Mar. Geod. 2009.
[8] Roland A. Development of WWM II: Spectral Wave Modelling on Unstructured Meshes. Technische Universitat Darmstadt, Institute of Hydraulic and Water Resources Engineering; 2008.
[9] Zhang YJ, Ye F, Stanev EV, Grashorn S. Seamless cross-scale modeling with SCHISM. Ocean Model. 2016;102:64–81.
[10] Li JG. Propagation of ocean surface waves on a spherical multiple-cell grid. J. Comput. Phys. 2012;8:262–77.
[11] Li J-G. Global transport on a spherical multiple-cell grid. Mon. Weather Rev. 2011;139:1536–55.
[12] Mentaschi L, Kadoulaki G, Vousdoukas MI, Voukouvalas E, Pereira P, Feyen L. The Effect of Changing Spatial Resolution in Global Dynamic Wave Models. 2018.