

## Preface

WaveRayModel is a Matlab™ App to examine wave refraction and shoaling over coastal bathymetries. Using a ray tracing method forward and backward tracking rays can be generated for a range of wave periods (frequencies), directions and water levels. Forward tracking provides a spatial visualisation of how waves approach the shore. Backward tracking is used to construct transfer tables that allow inshore wave conditions to be estimated from an offshore timeseries of waves (height, period and direction). Several different definitions of the offshore wave spectrum are provided and depth saturation can also be included (TMA spectrum). The App includes options to plot the offshore and inshore spectrums and the various transfer coefficients as a function of wave period, mean directions and water level. Note, however, that this implementation does not include processes of wave reflection, wave diffraction and wave breaking, although the latter is implicitly captured by including depth saturation using the TMA spectrum.

## Requirements

The model is written in Matlab™ and provided as Open Source code (issued under a GNU General Public License) and runs under v2016b or later. WaveRayModel uses the muitoolbox and dstoolbox.

The option to use a mesh instead of a Cartesian grid uses Mesh2D which is available to download from: <https://github.com/dengwirda/mesh2d>. Installation details are provided in Section 2.1.3.

## Resources

The WaveRayModel App and two toolboxes (muitoolbox and dstoolbox) can be downloaded from [www.coastalsea.uk](http://www.coastalsea.uk).

*Cite as:*

Townend, I.H., 2021, WaveRayModel manual, CoastalSEA, UK, pp65, [www.coastalsea.uk](http://www.coastalsea.uk).

## Bibliography

Abernethy, C.L., Gilbert, G., 1975. Refraction of wave spectra. INT 117, Wallingford, UK.

Townend, I.H., Savell, I.A., 1985. The application of ray methods to wave refraction studies. In: P.P.G. Dyke, Moscardini, A.O., Robson, E.H. (Ed.), Offshore and Coastal Modelling. Lecture Notes on Coastal and Estuarine Studies No12. Springer-Verlag, pp. 137–164.

## Acknowledgements

The original development of this type of model took place in the 1980's. The team working on the model at Sir William Halcrow & Partners was led by Chris Fleming and at various times included Ian Savell, Graham Copeland, Ian Townend, Geof Gilbert and Dominic Reeve.

The meshing tool uses the following code from the Matlab™ forum or GitHub.

[mesh2d](#): © Darren Engwirda, 2017.

[curvspace](#): © Yo Fukushima, 2005.

[trigradient](#): © Mick Warehime, 2013.

Details of all functions used (and source where relevant) are provided in Section 6.



## Revision history

Version	Date	Changes
1.1	May 2023	Option to use mesh added.
1.0	Mar 2023	First release of model as a beta test version



## Contents

1	Introduction .....	1
2	Getting started .....	2
2.1	Configuration.....	2
2.1.1	Installing the toolboxes .....	2
2.1.2	Installing the App .....	2
2.1.3	Installing Mesh2D .....	2
2.2	Opening WaveRayModel .....	3
2.3	Typical Workflow .....	3
2.4	Sample project.....	5
3	Application Menus .....	6
3.1	File.....	6
3.2	Tools.....	6
3.3	Project.....	6
3.4	Setup.....	7
3.5	Run .....	9
3.6	Analysis.....	12
3.6.1	Plotting .....	12
3.6.2	Statistics.....	14
3.6.3	Ray Plots.....	18
3.6.4	Spectral Plots .....	18
3.7	Help .....	20
3.8	Tabs .....	20
3.9	UI Data Selection .....	21
3.10	Accessing data from the Command Window .....	22
4	Supporting Information .....	24
4.1	Quality Checks .....	24
4.1.1	Waves .....	24
4.1.2	Water levels.....	24
4.2	Grid and Mesh options .....	25
4.3	Nearshore waves.....	25
4.3.1	Ray tracing .....	27
4.3.2	Spectral transfer.....	27
4.3.3	Wave spectra .....	28
4.3.4	Direction spreading .....	30
4.3.5	Wave buoy directional spreading .....	30
4.4	Output tables.....	31



---

4.4.1	Ray models .....	31
4.4.2	Transfer Table .....	31
4.4.3	Wave transfer model outputs.....	32
4.5	Model and program assumptions.....	32
4.6	Taylor diagram .....	33
4.6.1	Taylor diagram theory .....	33
4.6.2	Skill score .....	34
4.7	Derive Output .....	35
4.7.1	Calling an external function .....	36
4.7.2	Input and output format for external functions.....	37
4.7.3	Pre-defined functions .....	39
4.7.4	Adding variables to peak and cluster time series .....	43
5	User functions.....	44
5.1	User Statistics .....	44
5.2	User Plots .....	44
6	Program Structure.....	45
7	Bibliography.....	50
Appendix A – Input Data File Formats .....		52
Appendix B – Data set properties (DSproperties) .....		59
Appendix C - Date and Time Locale.....		60
Appendix D – Mesh parameters.....		61

## 1 Introduction

When computing power was scarce and before the advent of more sophisticated discretisation methods, ray tracing methods provided a powerful way of analysing wave propagation problems. Indeed, the practice of forward tracking wave rays as they approached the coast was originally done manually by coastal engineers to understand how energy was being focussed on the shore (USACE, 1984). Since then, advanced spectral wind-wave models, known as third-generation models, have been developed. These include models such as TOMAWAC (Benoit et al., 1997) and SWAN (Booij et al., 1999). These models solve the spectral action balance equation, without any a priori restrictions on the spectrum for the evolution of wave growth ([SWAN manual](#)). However, these remain computationally demanding and may be more than is needed for coastlines that are not too complex, especially where wave records at a few points rather than detailed spatial descriptions will suffice. For these applications, which included many design problems, the backward ray tracing method still has a role to play. The method was proposed by Abernathy and Gilbert as a way of circumventing some of difficulties when using forward tracking methods to estimate changes in wave height (Abernathy and Gilbert, 1975). Examples of its application include various coastal design problems at the time, with extensions to include reflections and wave breaking (Townend and Savell, 1985a) and wave diffraction (Southgate, 1981; Townend and Savell, 1985b).

WaveRayModel is a Matlab<sup>TM</sup> App to examine wave refraction and shoaling over coastal bathymetries. Using a ray tracing method forward and backward tracking rays can be generated for a range of wave periods (frequencies), directions and water levels. Forward tracking provides a spatial visualisation of how waves approach the shore. Backward tracking is used to construct transfer tables that allow inshore wave conditions to be estimated from an offshore timeseries of waves (height, period and direction). Several different definitions of the offshore wave spectrum are provided and depth saturation can also be included (TMA spectrum). The App includes options to plot the offshore and inshore spectrums and the various transfer coefficients as a function of wave period, mean directions and water level. Note, however, that this implementation does not include processes of wave reflection, wave diffraction and wave breaking, although the latter is implicitly captured by including depth saturation using the TMA spectrum.



## 2 Getting started

### 2.1 Configuration

WaveRayModel is installed as an App and requires `muitoolbox` and `dstoolbox` to be installed. The download for each of these includes the code, documentation and example files. The files required are:

`dstoolbox`: `dstoolbox.mltbx`

`muitoolbox`: `muitoolbox.mltbx`

The App file: `WaveRayModel.mlappinstall`

#### 2.1.1 Installing the toolboxes

The two toolboxes can be installed using the *Add-Ons > Manage Add-Ons* option on the Home tab of Matlab™. Alternatively, right-click the mouse on the ‘`mltbx`’ files and select install. All the folder paths are initialised upon installation and the location of the code is also handled by Matlab™. The location of the code can be accessed using the options in the *Manage Add-Ons* UI.

#### 2.1.2 Installing the App

The App is installed using the Install Apps button on the APPS tab in Matlab™. Alternatively, right-click the mouse on the ‘`mlappinstall`’ file and select install. Again, all the folder paths are initialised upon installation and the location of the code is handled by Matlab™.

Once installed, the App can be run from the APPS tab. This sets the App environment paths after which the App can be run from the Command Window using:

```
>> WaveRayModel;
```

The App environment paths can be saved using the Set Path option on the Matlab™ Home tab.

Documentation can be viewed from the App Help menu, or the Supplemental Software in the Matlab™ documentation. The location of the code can be accessed by hovering over the App icon and then finding the link in the pop-up window.

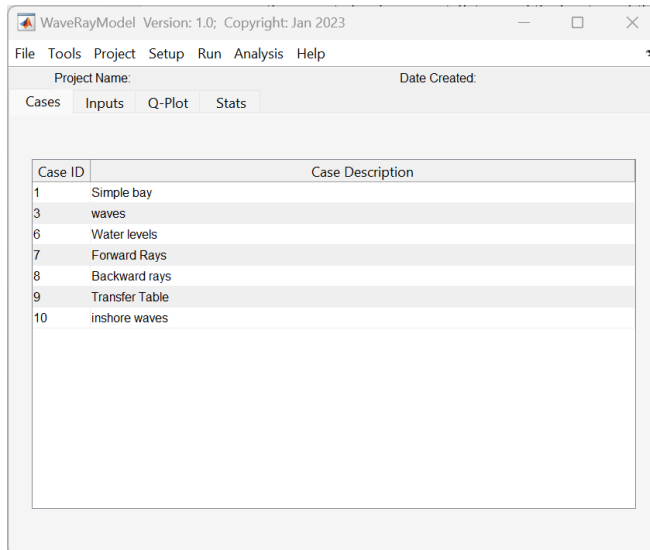
#### 2.1.3 Installing Mesh2D

An alternative to using a very detailed Cartesian grid is to use a triangular mesh that is detailed nearshore and less refines towards the offshore boundary. The ‘Create Mesh’ option provides a workflow to do this but requires that Mesh2D is downloaded and installed with the WaveRayModel App. Mesh2D is available to download from: <https://github.com/dengwirda/mesh2d>. Once downloaded the zip file should be extracted into the ‘`mesh2D`’ folder of the App. To find the location of the App, hover over the App icon, or use Home > Manage Add-ons on the Matlab ribbon and then select the options (3 dots) for the WaveRayModel App.

## 2.2 Opening WaveRayModel

A graphical user interface (GUI) is used to set-up, run cases, plot results and export model output.

A splash screen crediting the developers appears for a few seconds before being replaced by the WaveRayModel interface.



Version number

Drop-down menus

Project title and date

Display Tabs

Description of data loaded and model runs (Cases)

## 2.3 Typical Workflow

The following steps explain how to set-up a model using the various tools provided. Text in *Red italic* refers to drop down menus. Text in *Green italic* refers to Tab titles.

*File>New* to create a new project space.

*Setup>Input Data>Bathymetry*

The UI requests the user to select a file. Once added the grid can be viewed using the *Inputs* tab.

Changes to the grid can be made using the various tools provided in the *Setup>Grid Tools* menu. An idealised grid can be created by defining the grid dimensions using *Setup>Input Data>Grid Parameters* and then using *Run>Test Grid*.

*Setup>Run Parameters>Run Conditions*

The UI allows the conditions to be used for forward or backward tracking to be defined (range or wave periods and water levels, cut-off depth and shoreline angle).

*Then for Forward tracking:*

*Setup>Run Parameters>Forward Tracking:* The UI enables the start conditions for a set of forward tracking rays to be defined.

*Run> Check Start Points:* To ensure that the start point definition is as required.

*Run>Forward Tracking:* To generate the set of rays for the periods and water levels specified.

*Or for Backward tracking:*

*Setup>Run Parameters> Backward Tracking*

The UI enables the start conditions for a set of backward tracking rays to be defined.

*Run> Check Start Depth:* To see the location and depth of the start position.

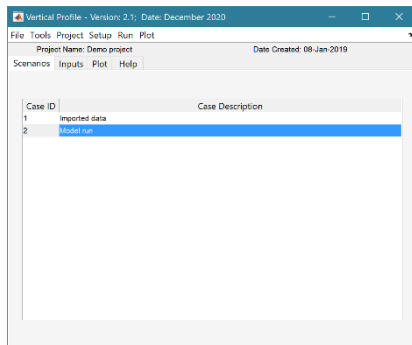
*Run>Backward Tracking:* To generate the set of rays for the periods and water levels specified.

When the run has completed the user is prompted to provide a description of the model run (case).

The run is listed on the *Cases* tab and the ray plots for a selected case can be viewed on the *Q-Plot* tab.

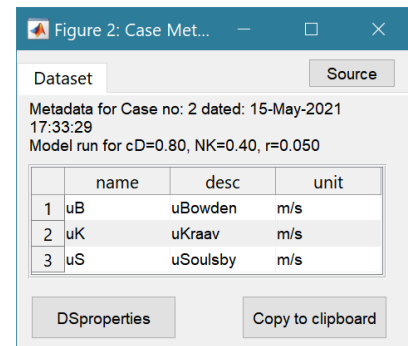
### Analysis>Plots

The results from a run can be selected and plotted. By using the Add button additional model runs can be included on the plot, allowing different Cases to be compared.



### Default User Interface

Left click mouse on a Case ID to see the associated meta-data (as shown on the right)



### Analysis>Ray Plots

User can select a Case and generate a stand-alone figure of the rays for selected run conditions.

### Run> Transfer Table

Uses a backtracking ray Case to generate tables of inshore and offshore conditions (wave direction, celerity, etc). The results can be viewed using the *Q-Plot* tab, or the *Analysis>Spectral Plots>Transfer Table* for a stand-alone figure.

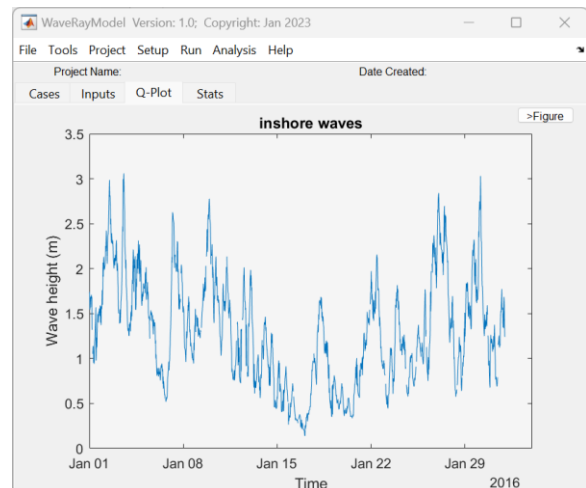
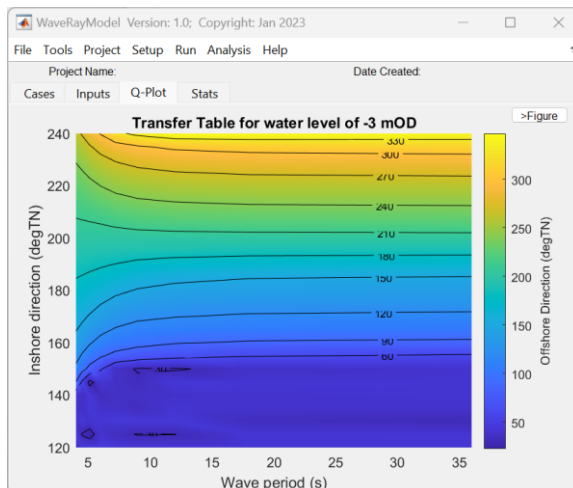
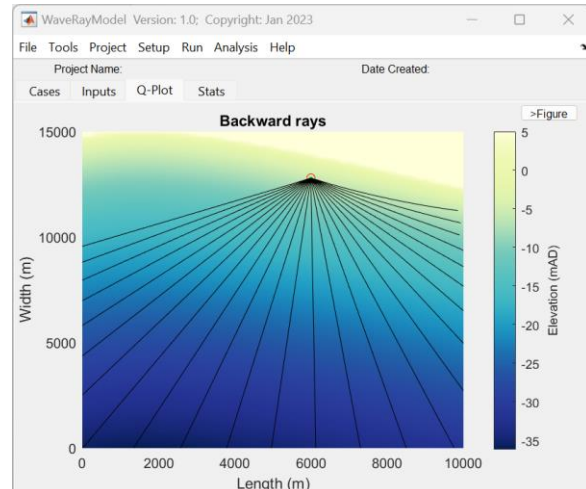
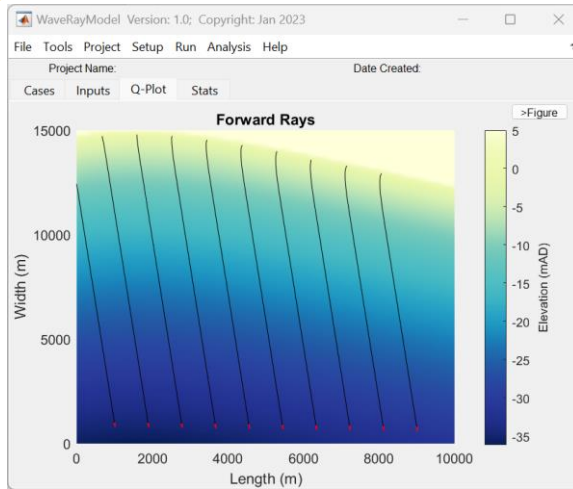
At this stage other output options can be used to examine the resulting transfer coefficients for a range of wave periods, mean offshore directions and water levels (using a unit wave height).

*Analysis>Spectral Plots>Transfer Coefficients*. And the offshore and inshore spectra for a defined wave or wind condition can be plotted using *Analysis>Spectral Plots>O/I Spectrum*.

To model a wave or wind time series, the offshore data must first be loaded using *Setup>Input Data*, and selecting Wave, Wind or Water Levels as appropriate. The imported data can be viewed using the *Q-Plot* tab, and quality controlled using the *Setup>Input Data*, option for *Quality Control*, or the *Setup>Data Clean-up*, to trim, concatenate or resample a time series. If a water level data set is loaded this is used, if not, then still water level is set zero datum. An inshore time series is created using *Run>Run Timeseries* and again the results can be viewed using the *Q-Plot* tab.

Sample output:





Note: when running time series there is an option to save just the wave properties or the wave properties and spectra. When saving both, the arrays can be large and result in a large file if saved (see Section XX for more details).

An alternative option is to load a CCO wave spectrum file using the *Setup>Input Data>Waves>Load* option and selecting the CCO wave spectrum format option to import a set of files. The raw data can be viewed using the *Q-Plot* tab. If spectrum data is selected to *Run>Run Timeseries*, the offshore and inshore spectra are computed, as well as the inshore wave properties. As with wave data input, the user has the option to save just the properties or the properties and spectra (but see Note about file size!).

## 2.4 Sample project

The *example* folder contains sample data files for some of the different types of data that can be loaded, and a demonstration project file.

### 3 Application Menus

The UI comprises a series of drop down menus that provide access to a number of commonly used functions such as file handling, management of run cases, model setup, running and plotting of the results. In addition, Tabs are used to display set-up information of the Cases that have been run. In this manual text in *Red italic* refers to drop down menus and text in *Green italic* refers to Tab titles.

#### 3.1 File

*File>New*: clears any existing model (prompting to save if not already saved) and a popup dialog box prompts for Project name and Date (default is current date).

*File>Open*: existing models are saved as \*.mat files. User selects a model from dialog box.

*File>Save*: save a file that has already been saved.

*File>Save as*: save a file with a new or different name.

*File>Exit*: exit the program. The close window button has the same effect.

#### 3.2 Tools

*Tools>Refresh*: updates *Cases* tab.

*Tools>Clear all>Project*: deletes the current project, including setup parameters and all Cases.

*Tools>Clear all>Figures*: deletes all results plot figures (useful if many plots have been produced).

*Tools>Clear all>Cases*: deletes all cases listed on the *Cases* tab but does not affect the model setup.

#### 3.3 Project

*Project>Project Info*: edit the Project name and Date.

*Project>Cases>Edit Description*: select a Case description to edit.

*Project>Cases>Edit Data Set*: edit a data set. Initialises a data selection UI to define the record to be edited and then lists the variable in a table so that values can be edited. The user can also limit the data set retrieved based on the variable range and the independent variable (X) or time. This can be useful in making specific edits (eg all values over a threshold or values within a date range).

**Tip:** Using the Copy to Clipboard button also provides a quick way of exporting selected data.

*Project>Cases>Save*: select the Case to be saved from the list of Cases, select whether to save the Case as a *dstable* or a *table* and name the file. The dataset *dstable* or *table* are saved to a mat file.

*Project>Cases>Delete*: select the Case(s) to be deleted from the list of Cases and these are deleted (model setup is not changed).

*Project>Cases>Reload*: select a previous model run and reload the input values as the current input settings.

*Project>Cases>View settings*: display a table of the model input parameters used for a selected Case.

*Project>Import/Export>Import*: load a Case class instance from a Matlab binary 'mat' file. Only works for data sets saved using Export.

*Project>Import/Export>Export*: save a Case class instance to a Matlab binary 'mat' file.

These last two functions can be used to move Cases between projects or models.

**NB:** to export the data from a Case for use in another application (eg text file, Excel, etc), use the *Project>Cases>Edit Data Set* option to make a selection and then use the 'Copy to Clipboard' button to paste the selection to the clipboard.

### 3.4 Setup

The setup menu provides a series of menus to enable different components of the model to be defined.

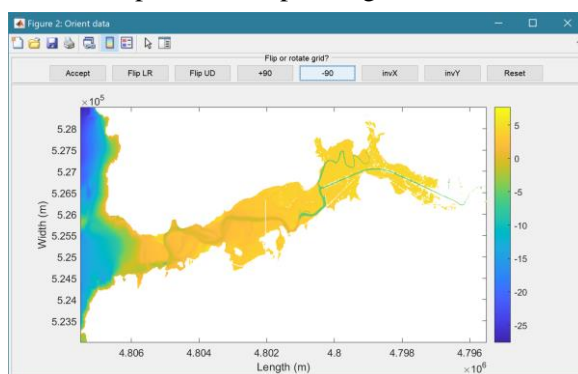
**Setup>Import Data>\*\*\*:** dialog with sub-menu options to Load, Add, Delete, Quality Control Bathymetry x,y,z data, wave, wind and water level time series data. The options (especially QC) vary depending on what is defined in the data specific format file. File formats currently provided for are detailed in Appendix A – Input Data File Formats.

Select one or more files to load. When the data has been loaded, the user is prompted to provide a description of the data set (Case) and is listed on the *Cases* tab and can be viewed on the *Q-Plot* tab. The source file(s) area defined in the meta-data, which can be viewed by selecting the Case ID field on the *Cases* tab for the record of interest and using the *Source* button on the top right of the table figure that is displayed.

**Setup>Import data> Load data:** prompts for file format to be loaded. The options available vary with Data type. The data is then loaded and the user is prompted for a description (working title) for the data set.

For bathymetry data the user is prompted for some additional information about the grid that can help control what is imported and given the option to flip or rotate the grid for use in the model.

UI to manipulate an imported grid



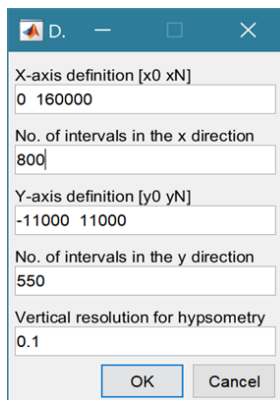
#### Button options:

Accept – uses the current settings to load grid  
 FlipLR – flips the grid left to right (horizontally)  
 FlipUD – flips the grid up-down (vertically)  
 +90 – rotates grid clockwise  
 -90 – rotates grid anti-clockwise  
 invX – reverses the direction of the X-axis  
 invY – reverses the direction of the Y-axis  
 Reset – restores the settings as loaded from file

**Setup>Import data > Add data:** prompts for file to be added (only one file at a time can be added) and the Case to use (if more than one Case). Only files with the format used to create the data set can be used to Add data to a data record and this is selected when the first file is loaded using the Load menu option.

**Setup>Import data > Delete data:** prompts for Case from which some part of the data is to be deleted.

**Setup>Import data > Data QC:** runs a series of checks on the data. This is only available if defined for the specific data format.



*Setup>Grid Parameters*: enter and edit the grid parameters used to define the model domain

Upper and lower limits for the x co-ordinates

Grid spacing along the x-axis

Upper and lower limits for the y co-ordinates

Grid spacing along the y-axis

Not used in this model. D

Definition used to generate synthetic bathymetry using *Run>Test Grid*.

A range of tools to manipulate cartesian grids.

*Setup>Grid Tools>Translate Grid*: interactively translate grid x-y coordinates.

*Setup>Grid Tools>Rotate Grid*: interactively flip or rotate grid<sup>1</sup>.

*Setup>Grid Tools>Re-Grid*: re-grid a gridded dataset to match another grid or to user specified dimensions.

*Setup>Grid Tools>Sub-Grid*: interactively define a sub-grid and save grid as a new Case.

*Setup>Grid Tools>Combine Grids*: superimpose one grid on another based on maximum or minimum set of values.

*Setup>Grid Tools>Add Surface*: add horizontal surface to an existing grid.

*Setup>Grid Tools>To curvilinear*: map grid from cartesian to curvilinear coordinates.

*Setup>Grid Tools>From curvilinear*: map grid from curvilinear to cartesian coordinates.

*Setup>Grid Tools>Display Dimensions*: display a table with the dimensions of a selected grid.

*Setup>Grid Tools>Difference Plot*: generate a plot of the difference between two grids.

*Setup>Grid Tools>Plot Sections*: interactively define section on a grid and plot as sections.

*Setup>Grid Tools>Digitise Line*: interactively digitise a line (with option to add elevations) using selected grid as base map.

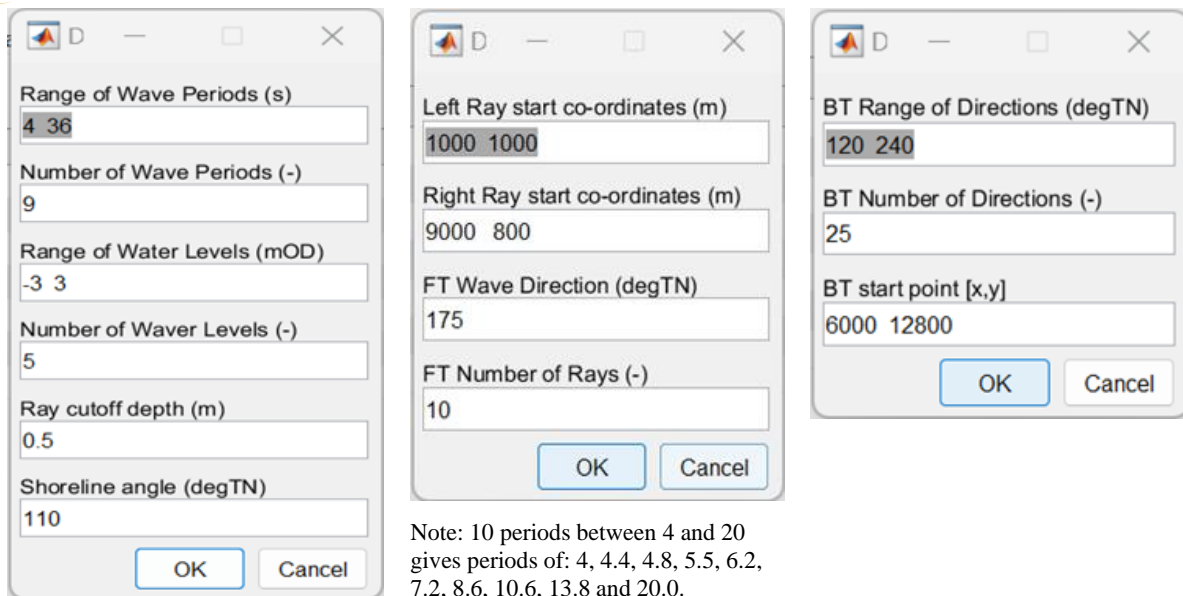
*Setup>Grid Tools>Export xyz Grid*: select a Case and export grid as xyz tuples.

*Setup>Run Parameters>Run Conditions*: used to define the range of wave periods and water levels, together with the number of intervals to use (see Section 4.3.1 for details of frequency spacing). The ray cut-off water depth and the shoreline angle are also defined in this menu.

*Setup>Run Parameters>Forward Tracking*: used to define the coordinates of a transect along which the rays start, together with the wave direction that the waves are FROM and the number of rays.

*Setup>Run Parameters>Backward Tracking*: used to define the range of directions that the waves are FROM (if straight from offshore) and the number of rays, along with the coordinates of the start point.

<sup>1</sup> Rotation re-orientates the z grid and swaps x and y axes when rotating +/-90°. However, it does not change the order of the x and y axes. Consequently, rotating +90° or flipping left-right reverses the co-ordinate values on the x-axis and rotating -90° or flipping up-down reverses the co-ordinate values on the y-axis.



The image shows three dialog boxes for configuring the WaveRayModel. The first dialog, titled 'D', contains fields for 'Range of Wave Periods (s)' (4 36), 'Number of Wave Periods (-)' (9), 'Range of Water Levels (mOD)' (-3 3), 'Number of Waver Levels (-)' (5), 'Ray cutoff depth (m)' (0.5), and 'Shoreline angle (degTN)' (110). The second dialog, also titled 'D', contains fields for 'Left Ray start co-ordinates (m)' (1000 1000), 'Right Ray start co-ordinates (m)' (9000 800), 'FT Wave Direction (degTN)' (175), and 'FT Number of Rays (-)' (10). The third dialog, titled 'D', contains fields for 'BT Range of Directions (degTN)' (120 240), 'BT Number of Directions (-)' (25), and 'BT start point [x,y]' (6000 12800). Each dialog has 'OK' and 'Cancel' buttons.

Note: 10 periods between 4 and 20 gives periods of: 4, 4.4, 4.8, 5.5, 6.2, 7.2, 8.6, 10.6, 13.8 and 20.0.

*Setup>Data clean-up>Concatenate two timeseries:* allows two timeseries data sets to be joined. Two records, or two variables, of a similar type can be joined to form a single timeseries or timeseries collection. The user is prompted to select the timeseries records to be used (only records of the same type are shown for the second selection). If there is an overlap in time, the user is prompted to select which timeseries to use for the overlap. The option is then provided to create a new timeseries based on selecting a Single variable or a timeseries collection that concatenates All variables. For single variables they can have different variable names. For All variables, the selected records must have the same set of variables with the same names. This is useful to merge data sets from different sites to extend the record.

*Setup>Data clean-up>Resample timeseries:* allows a selected timeseries to be resampled at user specified interval, using a user specified method (e.g., mean/max/min over the interval). This can be useful to reduce size of a data set (e.g., long tidal records before using the data in models).

*Setup>Data clean-up>Patch timeseries:* allows gaps in a selected timeseries to be patched using the data from another timeseries that overlaps the primary timeseries (at least for some or all the gaps). The function assumes that the primary data set has time intervals defined with no data (NaNs). These times are matched with coincident times in the timeseries used to make the patch (there is currently no interpolation in time) and the data values of the patch timeseries are added to the primary timeseries.

*Setup>Data clean-up>Trim timeseries:* allows the start and end dates of a timeseries to be modified.

*Setup> Model Constants:* various constants are defined for use in models, such as the acceleration due to gravity, viscosity and density of sea water, and density of sediment. Generally, the default values are appropriate (9.81, 1.36e-6, 1025, 2650 respectively) but these can be adjusted and saved with the project if required.

### 3.5 Run

*Run> Check Start Points:* utility generates a plot showing the bathymetry, start points and initial ray directions based on the current input settings.

*Run>Forward Rays:* uses the defined run parameters (*Setup>Run Parameters>Run Conditions*) and start points (*Setup>Run Parameters>Forward Tracking*) to generate a set of rays. These can then be viewed on the *Q-Plot* tab, or using the *Analysis>Ray Plots* option.



**Run> Check Start Depth:** utility display the minimum and maximum depth at the start point for backward tracking based on the current input settings.

**Run>Backward Rays:** uses the defined run parameters (*Setup>Run Parameters>Run Conditions*) and start point (*Setup>Run Parameters>Backward Tracking*) to generate a set of rays. These can then be viewed on the *Q-Plot* tab, or using the *Analysis>Ray Plots* option.

**Run> Transfer Table:** generates the Transfer Table from a set of Backward Tracking rays. Summary output from the tables can then viewed on the *Q-Plot* tab, or using the *Analysis>Spectral Plots>Transfer Table* option.

**Run> Run Wave Timeseries:** uses a selected Transfer Table and imported wind or wave data to compute the inshore conditions which are saved as a timeseries with the following variables:

Hsi - Inshore wave height	kw - Wave transfer coefficient
T2i - Inshore mean period	kt2 - Mean period coefficient
Diri - Inshore wave direction	ktp - Peak period coefficient
Tpi - Inshore peak period	kd - Mean direction shift
Diripk - Inshore peak direction	swl - Still water level
	depi - Inshore depth

Alternatively, imported directional wave spectra data can be selected to estimate the same set to wave properties. Regardless of the input source, there is also an option to save the offshore and inshore directional-frequency spectra as well as the wave properties. However, for a timeseries the spectra arrays rapidly become very large. Typically, a month of data at 3 hour intervals will generate 600 Mb of output. For this reason, there is an option to sub-sample the arrays down to 1-degree and 1 second direction-frequency intervals and this reduces the size to around 150 Mb/month.

[The interpolation interval of 0.5 degrees and 0.5 seconds is a property defined in the SpectralTransfer class constructor. The downsizing of the arrays to 1 degree and 1 second intervals is done in WRM\_WaveModel.runModel].

**WARNING:** only save wave spectra timeseries of short periods unless there is a specific need for the full time series of spectral data.

**Run>Create Mesh:** generate an unstructured triangular mesh. A detailed Cartesian grid can be used to generate a triangular mesh. This can be computationally more efficient than a very detailed Cartesian grid because the resolution can be varied across the model domain from fine nearshore to coarse offshore.

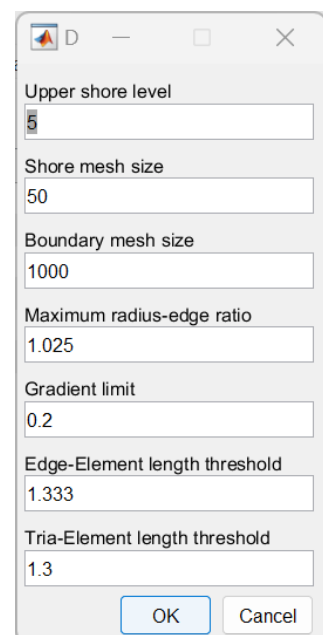
**NB:** this option requires mesh2D to have been installed – see Section 2.1.3.

The shore boundary of the mesh is defined by the Upper shore level and the target mesh size along the shoreline is defined by the Shore mesh size. The Boundary mesh sizes allow some mesh refinement but, if set too fine, can result in a fine mesh around the domain boundary and a coarser mesh in the middle of the domain.

The Maximum radius-edge ratio can be used to refine the mesh (min 1.0)

The Gradient limit and the Edge-Element and Tria-Element normalised length thresholds all provide additional ways of refining the mesh (generally making the mesh denser and hence more computationally demanding).

For further details on mesh2D parameter settings see Section 4.2 and the mesh2D example files, inline documentation and the references detailed on the download site (<https://github.com/dengwirda/mesh2d>).



The screenshot shows a dialog box titled 'D' with the following parameters and values:

- Upper shore level: 5
- Shore mesh size: 50
- Boundary mesh size: 1000
- Maximum radius-edge ratio: 1.025
- Gradient limit: 0.2
- Edge-Element length threshold: 1.333
- Tria-Element length threshold: 1.3

At the bottom, there are 'OK' and 'Cancel' buttons.

Initially the extracted shoreline is plotted with an option to smooth the line if desired. The boundary points are then plotted, showing the distribution of points that will be used to generate the mesh. (Note: some other parameters are set by default in the WRM\_Mesh class, which is used to call mesh2D). If these are not suitable, 'Quit' and restart the process setting different mesh parameters until a suitable resolution and distribution of elements is obtained. A final plot shows the triangulated mesh superimposed on a surface plot of the bathymetry derived from the mesh and which can be saved or deleted. Some further information about grids and meshes is provided in Section 4.2.

**Run>Test Grid:** generate an idealised bathymetry for a grid as defined in **Setup>Grid Parameters**. The options include a planar or Dean beach profile cross-shore form and a straight or crenulate embayed shoreline. There is also an option to include a mound in the bathymetry.

**Run> Derive Output:** data that has been added (either as data or modelled values) can be used to derive new variables. The UI allows the user to select data and use a chosen selection of data/variable/range to define either a Variable, XYZ dimension, or Time. Each data set is sampled for the defined data range. If the data set being sampled includes NaNs the default is for these to be included (button to right of Var-limits is set to '+N'). To exclude NaNs press the button so that it displays '-N'.

The selection is assigned by clicking one of the X, Y or Z buttons. The user is prompted to assign a Variable, XYZ dimension, or Time (the options available varies with the type of variable selected) – see Section 3.9 for details of how this works.

An equation is then defined in the text box below using the x, y, z or t variables<sup>2</sup>. Based on the user selection the routine applies the defined variable ranges to derive a new variable. In addition text inputs required by the call and the model object (mobj) can also be passed.

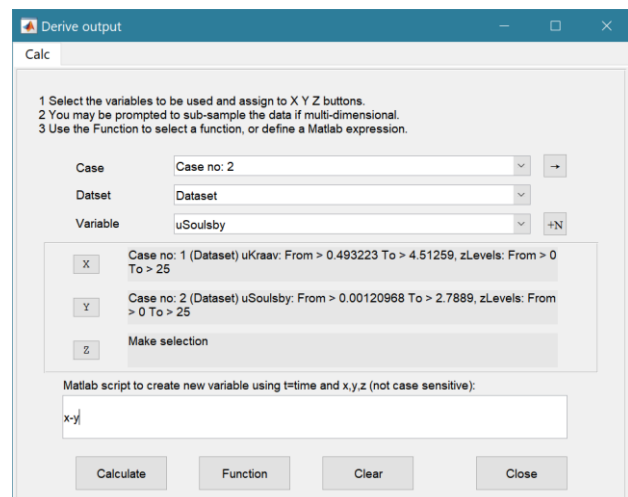
Adding the comment %time or %rows, allows the the row dimension to be added to the new dataset. For example if x and y data sets are timeseries, then a Matlab<sup>TM</sup> expression, or function call, call can be used to create a new time series as follows:

```
x^2+y %time
```

The output from function calls can be figures or tables, a single numeric value, or a dataset to be saved (character vectors, arrays or dstables). External functions should return the table RowNames (e.g., time or location) as the first variable (or an empty first variable), followed by the other variables to be saved.

If there is no output to be passed back the function should return a string variable.

If `varout = 'no output'`; this suppresses the message box, which is used for single value outputs. For expressions that return a result that is the same length as one of the variables used in the call, there



<sup>2</sup> Various pre-defined function templates can be accessed using the 'Function' button. Alternatively, text can be pasted into the equation box from the clipboard by right clicking in the text box with the mouse.

is the option to add the variable to the input dataset as a new variable. In all there are three ways in which results can be saved:

1. As a new dataset;
2. As an additional variable(s) to one of the input datasets;
3. As an additional variable(s) to some other existing dataset.

For options 2 and 3, the length of the new variables must be the same length (numbers of rows) as the existing dataset.

An alternative when calling external functions is to pass the selected variables as dstables, thereby also passing all the associated metadata and RowNames for each dataset selected. For this option up to 3 variables (plus time if defined for a selected variable) can be selected but they are defined in the call using dst, for example:

```
[time,varout] = myfunction(dst, 'usertext', mobj);
```

```
dst = myfunction(dst, 'usertext', mobj);
```

This passes the selected variables as a struct array of dstables to the function. Using this syntax the function can return a dstable, or struct of dstables, or a number of variables. When a dstable, or struct of dstables is returned, it is assumed that the dsproperties have been defined in the function called and dstables are saved without the need to define the meta-data manually.

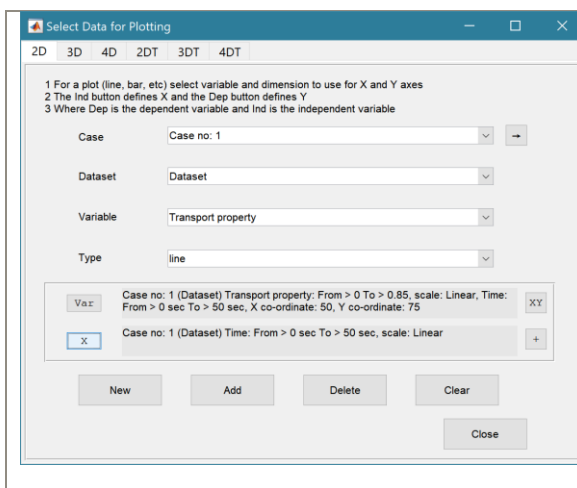
Some further details on using this option and the **'Function'** library available are provided in Section 4.7.

## 3.6 Analysis

Plotting and Statistical Analysis both use the standard Data selection UI. These both require Case, Dataset and Variables to be selected from drop-down lists and assigned to a button. Further details of how this works are given in Section 3.9.

### 3.6.1 Plotting

**Analysis>Plot menu:** initialises the Plot UI to select variables and produce several types of plot. The user selects the Case, Dataset, and Variable to be used and the plot Type from a series of drop-down lists. There are then buttons to create a New figure, or Add, or Delete variables from an existing figure for 2D plots, or simply a Select button for 3D and 4D plots. The following figures illustrate the options available.



#### 2D plot

For each selection choose the Case, Dataset and Variable to be used.

> Assign a variable, or a dimension, to the Var and X buttons to set the Y and X axes, respectively

Each selection can be scaled (log, normalised, etc) and the range to be plotted can be adjusted when assigning the selection to a button.

> Select plot type (line, bar, scatter, stem, etc)

#### Control Buttons:

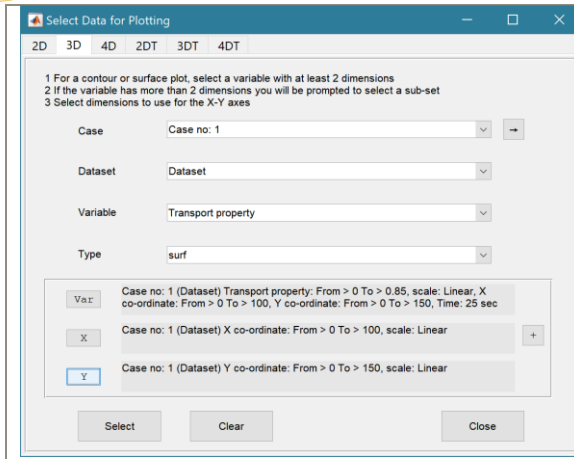
→ : updates the list of Cases

XY : swaps the X and Y axes

+ : switches between cartesian and polar plot type

*If polar selected then Ind assumed to be in degrees.*





### 3D plot

For each selection choose the Case, Dataset and Variable to be used.

> Assign selections to the Var, X and Y buttons

Take care to ensure that the assignments to X and Y correctly match the dimensions selected for the variable (including any adjustment of the dimension ranges to be used).

> Select plot type.

Control Buttons: see 2D plot above.

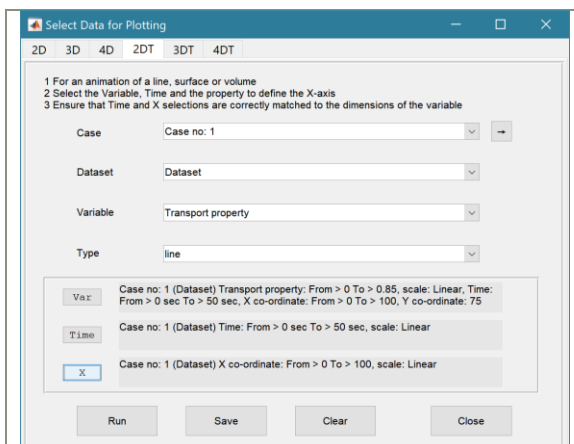
For all plot types, when the data has more dimensions than the plot or animation the user is prompted to sub-select from the data (by selecting sampling values for the dimensions that are not being used).

Animations follow a similar workflow. There are buttons at the bottom of each tab to:

**Run** the selection and create an animation,

**Save** the animation to a file (the animation needs to have been run first) . There is also an option to save on the bottom left of the animation figure.

**Clear** the current selection.



### 2DT animation

For each selection choose the Case, Dataset and Variable to be used.

> Assign a variable, or a dimension, to the Var, Time and X buttons.

Each selection can be scaled (log, normalised, etc) and the range to be plotted can be adjusted when assigning the selection to a button.

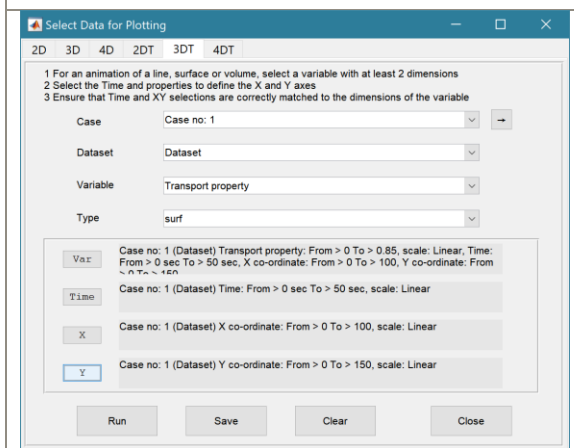
> Select plot type (line, bar, scatter, stem, etc)

Control Buttons:

→ : updates the list of Cases

+ : switches between cartesian and polar plot type

*If polar selected, then X assumed to be in degrees and when prompted select Polar and NOT Rose.*



### 3DT animation

For each selection choose the Case, Dataset and Variable to be used.

> Assign selections to the Var, Time, X and Y buttons

Take care to ensure that the assignments to Time, X and Y correctly match the dimensions selected for the variable (including any adjustment of the dimension ranges to be used).

> Select plot type.

Control Buttons: see 2DT plot above.

### Selection of User plot type

Calls the user\_plot.m function, where the user can define a workflow, accessing data and functions already provided by the App, or the mui toolbox. The sample code can be found in the psfunctions folder and illustrates the workflow to a simple line plot using x-y data from the 2D tab and a surface plot using x-y-z data from the 3D tab.

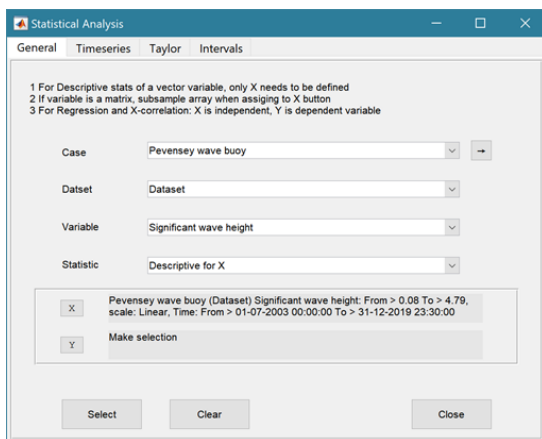
### 3.6.2 Statistics

**Analysis > Statistics:** several statistical analysis options have been included within the Statistical Analysis GUI. The tabs are for **General** statistics, **Timeseries** statistics, and model comparisons using a **Taylor Plot**.

#### General tab

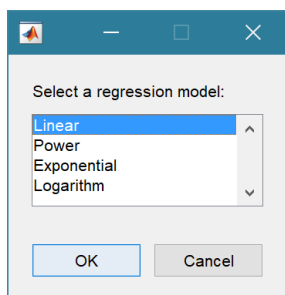
The General tab allows the user to apply the following statistics to data loaded in ModelUI:

- 1) **Descriptive for X:** general statistics of a variable (mean, standard deviation, minimum, maximum,



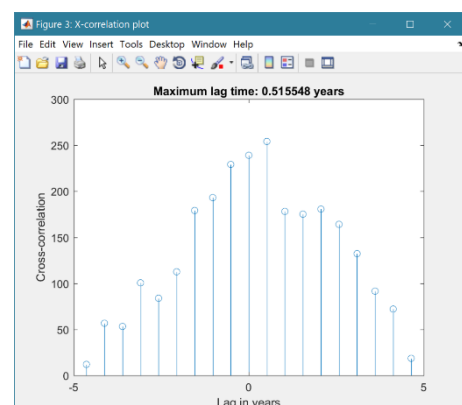
sum and linear regression fit parameters). Only X needs to be defined. The range of the variable can be adjusted when it is assigned to the X button (see Section 3.9). If the variable being used is a multi-dimensional matrix (>2D), the user is prompted to define the range or each additional dimension, or select a value at which to sample. The function can return statistics for a vector or a 2D array.

The results are tabulated on the **Stats > General** tab and can be copied to the clipboard for use in other applications.



- 2) **Regression:** generates a regression plot of the dependent variable, Y, against the independent variable, X. For time series data, the default data range is the maximum period of overlap of the two records. For other data types the two variables must have the same number of data points. After pressing the Select button, the user is prompted to select the type of model to be used for the regression. The results are output as a plot with details of the regression fit in the plot title.

- 3) **Cross-correlation:** generates a cross-correlation plot of the reference variable, X, and the lagged variable, X (uses the Matlab 'xcorr' function). For time series data, the default data range is the maximum period of overlap of the two records. For other data types the two variables must have the same number of data points. This produces a plot of the cross-correlation as a function of the lag in units selected by the user.



- 4) **User:** calls the function `user_stats.m`, in which the user can implement their own analysis methods and display results in the UI or add output to the project Catalogue. Currently implements an analysis of clusters as detailed for Timeseries data below.

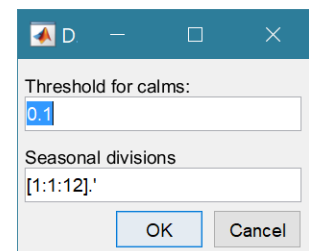
### Timeseries tab

The Timeseries tab allows the user to select a single Timeseries variable and apply any of the following statistics:

- 1) **Descriptive:** general statistics of a variable (mean, standard deviation, minimum, maximum, sum and linear regression fit parameters). The results are tabulated in a new window and can be copied to the clipboard for use in other applications.

Various ‘seasonal’ sub-divisions can be defined. The required option is selected from the table in the UI, by selecting a Syntax cell and then closing the UI.

The next UI prompts for a threshold for calms (values below threshold are deemed to be “calm” conditions) and allows the selected ‘seasonal’ divisions to be changed (if the desired option is not in the default list), or edited. The divisions can be expressed in several ways, as detailed below:



Script	Result
1	Descriptive statistics for the full-time series
[1:1:12].'	Descriptive statistics for the full-time series and monthly values (the '.' creates a column vector).
[12,1,2; 3,4,5; 6,7,8; 9,10,11]	Descriptive statistics for the full-time series and seasons based on groupings – Dec-Feb, Mar-May, Jun-Aug, Sep-Nov shown.

When seasonal statistics are produced with more than 2 seasons a plot is generated. This can be a cartesian or polar plot of the mean values with error bars used to depict +/- one standard deviation. The polar plot maps the year as one revolution.

- 2) **Peaks:** generates a new timeseries of peaks over a defined threshold. There are three methods that can be selected:

- 1 = all peaks above the threshold;
- 2 = the peak value within each up-down crossing of threshold; and
- 3 = peaks that have a separation of at least ‘*tint*’ hours.

For option 3, the separation between peaks (‘*tint*’) is also defined in the pop-up gui. This can be used to try and ensure that peaks are independent. The peaks are marked on a plot with the defined threshold. If rejected, new values can be defined. If accepted a new timeseries is added. This has the class of the Data Type that was used as the source timeseries but is not appended to that timeseries because the date/times are a subset of the source.

- 3) **Clusters:** The selection process is similar to peaks, where the user defines a threshold, selection method and time between peaks (for method 3). In addition, the cluster interval is defined in days. This is the period of time separating two peaks for them to be no longer considered part of a cluster (e.g. if a sequence of storms occurs every few days they will form a cluster. If there is then a gap of, say, 31 days to the next storm, with a cluster time interval of 30 days this would be considered as part of the next cluster). Once a selection has been made, a plot is generated that shows the peaks for each cluster with a different symbol. The user can either choose a different definition, or accept the definition. Once accepted, the results are

added as a new timeseries, with the class of the Data Type that was used as the source timeseries. Two values are stored at the time of each peak in the clusters: the magnitude of the peak; and the number of the cluster to which it belongs (numbered sequentially from the start). This allows the data for individual clusters to be retrieved, if required.

- 4) **Extremes:** The selection process is similar to peaks, where the user defines a threshold, selection method and time between peaks (for method 3). A figure is generated with two plots. The left-hand plot shows the peaks for the defined threshold and the right hand plots shows the mean excess above the threshold (circles), the 95% confidence interval (dotted red lines) and the number of peaks (vertical bars + right hand axis) as a function of threshold. This plot can be used to help identify a suitable threshold for the peak-over-threshold extremes analysis method. The user can either choose a different definition, or accept the definition. Once accepted, the user is prompted to select a plot type. Options are: None; Type 1 – a single return period plot; Type 2 – a composite plot showing the probability, quantile, return period and density plots. See Coles (2001) for further details of the method used and the background to these plots. The results are tabulated on the *Stats/Extremes* tab and can be copied to the clipboard for use in other applications.
- 5) **Poisson Stats:** user is prompted to select a threshold, method and peak separation (see Peaks above) and the function generates a plot of the peak magnitude, time between peaks (interarrival time) and the duration above the threshold for each peak. The plot shows a histogram of each variable and the exponential pdf derived from the data, along with the  $\mu$  value for the fit.
- 6) **Hurst Exponent:** user is prompted to select from one of 3 methods, which are based on different computation routines taken from the Matlab Forum, as follows:
  - 1 = Chiarello matrix method,
  - 2 = Abramov loop code,
  - 3 = Aalok-Ihlen code and
  - 4 = Aste using unweighted option.

Methods 1 and 2 are similar, whereas method 3 explores the effect of scale and method 4 derives the unweighted generalized Hurst exponent. The main difference between the first two methods is that Abramov uses a different form for  $S$ , rather than the Matlab standard deviation function (std).

The Hurst parameter  $H$  is a measure of the extent of long-range dependence in a time series (while it has another meaning in the context of self-similar processes).  $H$  takes on values from 0 to 1. A value of 0.5 indicates the absence of long-range dependence. The closer  $H$  is to 1, the greater the degree of persistence or long-range dependence.  $H$  less than 0.5 corresponds to a lack of persistence, which is the opposite of LRD indicates strong negative correlation so that the process fluctuates violently.  $H$  is also directly related to fractal dimension,  $D$ , where  $1 < D < 2$ , such that  $D = 2 - H$ .

This is experimental code (for code see `.../muitoolbox/psfunctions/hurst_exponent.m`, `hurst_aalok_ihlen.m` and `genhurstw.m`) and the user should refer to the background literature for further details. (Di Matteo et al., 2003; Pacheco et al., 2008; Ihlen, 2012; Morales et al., 2012; Sutcliffe et al., 2016; Abramov and Khan, 2017; Antoniadis et al., 2021; Brandi and Di Matteo, 2021).

- 7) **User:** calls `user_stats.m` function, where the user can define a workflow, accessing data and functions already provided by the particular App, or the `muitoolbox`. The sample code can be found in the `psfunctions` folder and illustrates the workflow to produce a clusters plot. Some code in the header (commented out) shows how to get a time series using the handles passed

to the function (obj and mobj). This code would get the same timeseries as the one passed to the function. However, by modifying the 'options' variable it is possible to access other timeseries variables.

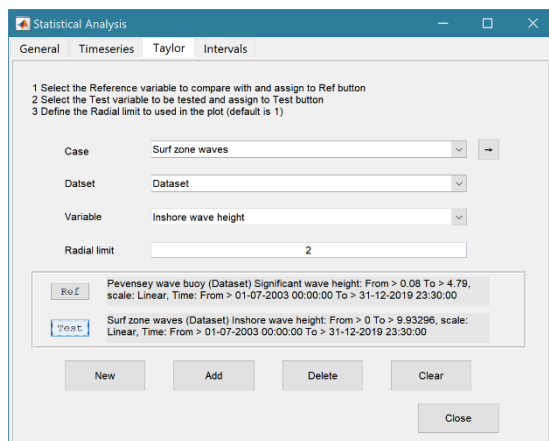
## Taylor tab

The Taylor tab allows the user to create a Taylor Plot using 1D or 2D data (e.g timeseries or grids):

A Reference dataset and a Test dataset are selected. Datasets need to be the same length if 1D, or same size if 2D. If the data are timeseries they are clipped to a time-period that is common to both, or any user defined interval that lies within this clipped period. The statistics (mean, standard deviation, correlation coefficient and centred root mean square error) are computed, normalized using the reference standard deviation and plotted on a polar Taylor diagram (Taylor, 2001).

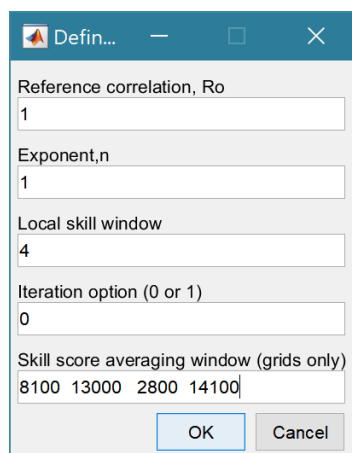
*[The ModelSkill App provides additional tools to test data and the ModelSkill App manual provides further details of the methods used.]*

Selecting New generates a new Taylor Plot. Selecting the Add button adds the current selection to an existing plot and the Delete button deletes the current selection. The Clear button resets the UI to a blank selection.



Once New or Add are selected, the user is asked whether they want to plot the skill score (Yes/No). If Yes, then the user is prompted to set the skill score parameters. As further points are added to the plot, this selection remains unchanged (i.e., the skill score is or is not included). To reset the option, it is necessary to close and reopen the Statistics UI.

If the number of points in the Reference and Test datasets are not the same the user is prompted to select which of the two to use for interpolation.



This is the maximum achievable correlation (see Taylor (2001) for discussion of how this is used).

Exponent used in computing the skill score (see ModelSkill manual for details).

Number of points (+/-W) used to define a local window around the ith point. If W=0 (default) the local skill score is not computed.

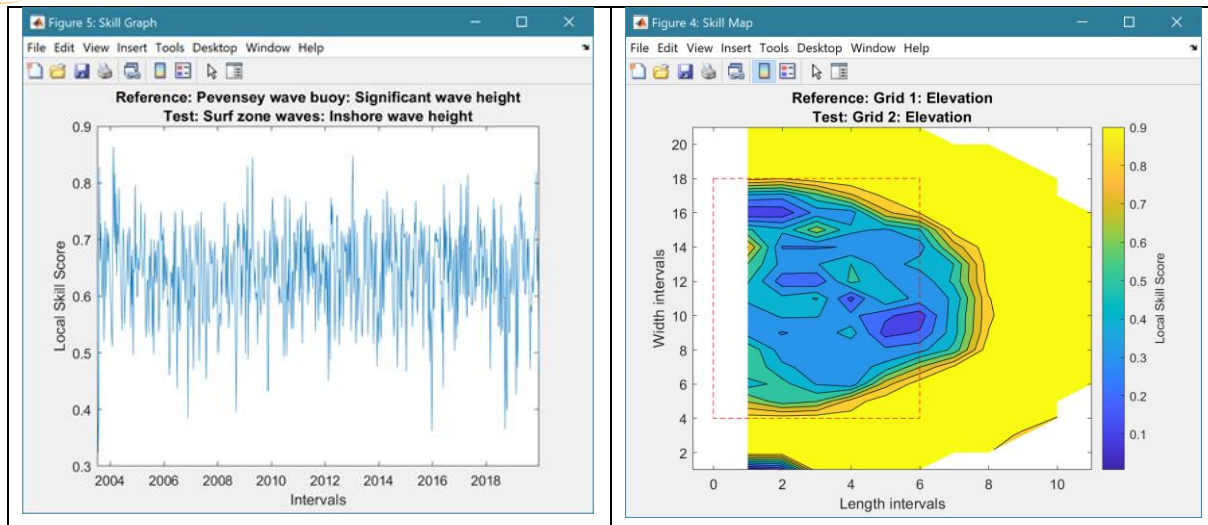
Local skill score is computed for window around every grid cell (=1), or computes score for all non-overlapping windows (=0)

Window definition to sub-sample grid for the computation of the average **local** skill score. Format is [xMin, xMax, yMin, yMax].

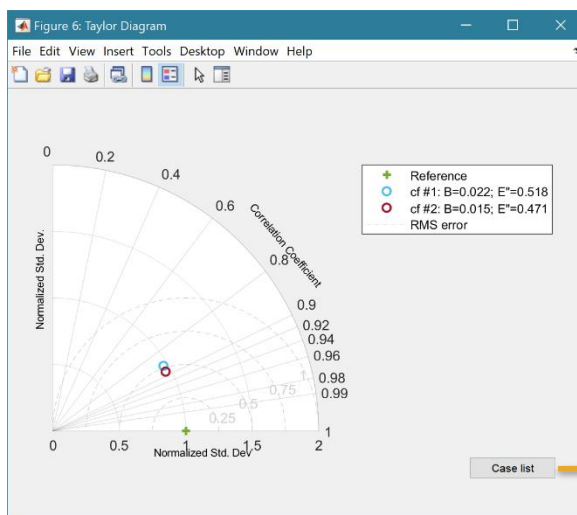
(a) time series skill score plot

(b) grid skill score plot



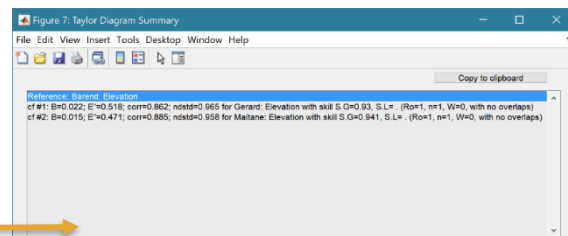


The Taylor Plot shows the Reference point as a green cross and the Test points as coloured circles. The legend details the summary statistics and the Case List button generate a table figure listing all the results. These can be copied to the clipboard.



Taylor diagram legend includes: B – bias; E'' – normalised RMS difference

The normalised standard deviation and correlation coefficient are also given in the Case List table, along with the global skill score, Sg, and the average local skill score, Sl.

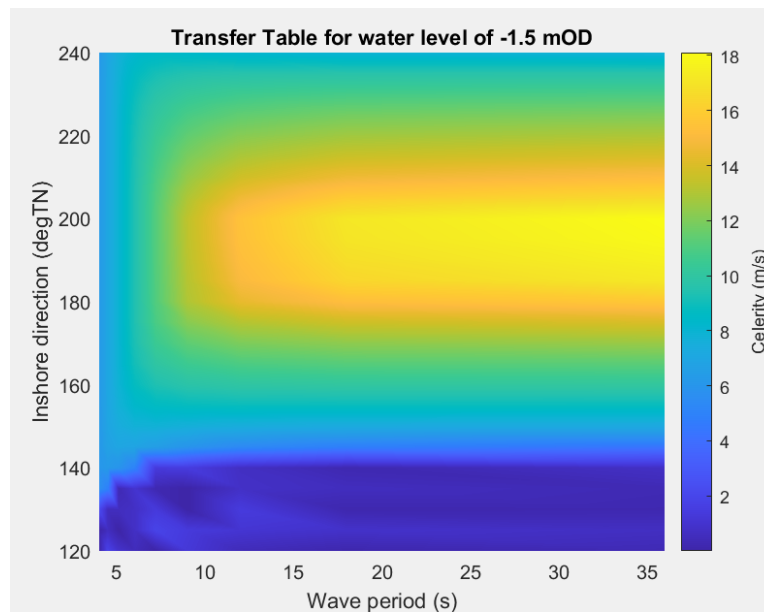


### 3.6.3 Ray Plots

**Analysis>Ray Plots:** used to select a case and plot the rays for a selected wave period and water level, or all rays for all periods and levels. There is also the option to plot the celerity or group celerity variation along a single set of rays.

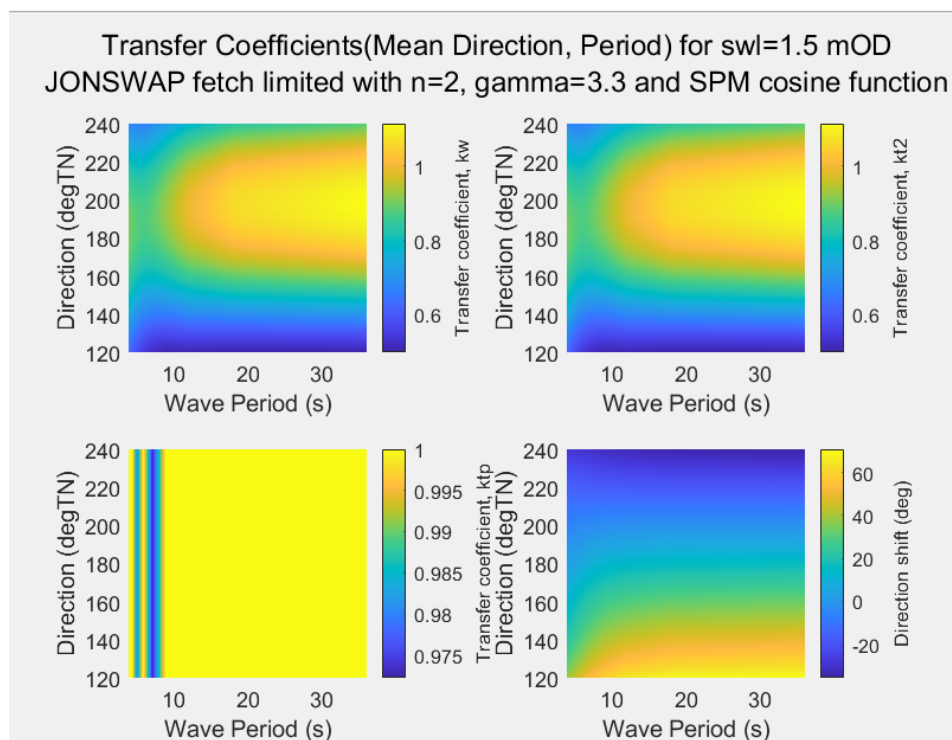
### 3.6.4 Spectral Plots

**Analysis>Spectral Plots>Transfer Table:** uses a Transfer Table created using **Run> Transfer Table** to plot data from the inshore or offshore Transfer Tables. For a selected water level the options are for celerity or group celerity inshore and direction, celerity, group celerity, offshore depth (seaward end of ray), average depth along rays and minimum depth along rays.

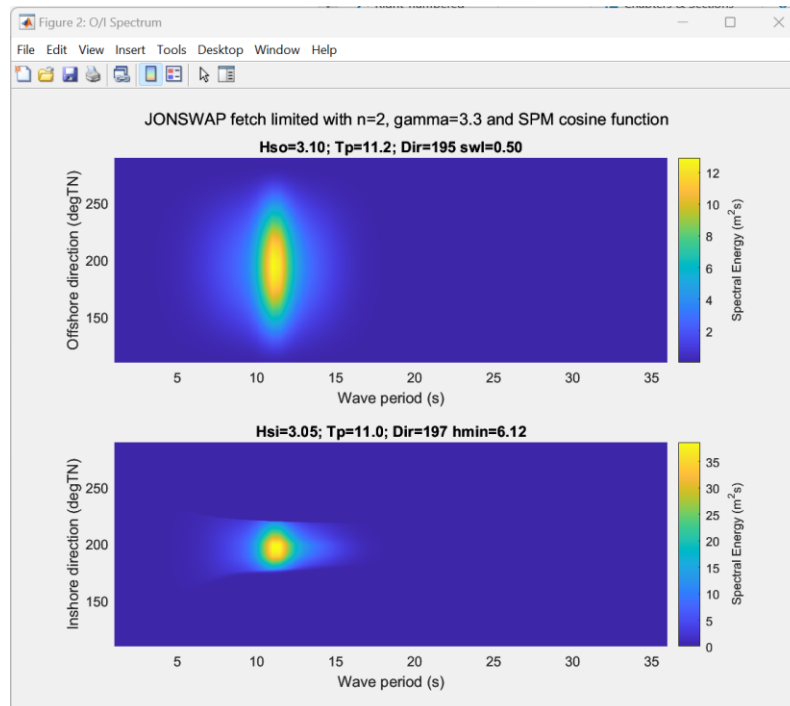


At this stage other output options can be used to examine the resulting transfer coefficients for a range of wave periods, mean offshore directions and water levels (using a unit wave height).

*Analysis>Spectral Plots>Transfer Coefficients:* uses a selected Transfer Table and a unit wave height ( $H_s=1\text{m}$ ) to derive the inshore wave parameters and return a set of transfer coefficients. These can then be plotted as a function of wave period and mean offshore direction for a selected water level.



*Analysis>Spectral Plots>O/I Spectrum:* uses a selected Transfer Table and user defined input conditions for wind or wave forcing to derive a 2D offshore spectrum and the associated inshore spectrum, which are then plotted along with details of the conditions. This includes an option to plot the output as an X-Y plot or a polar plot.



**Analysis>Spectral Plots>O/I Animation:** uses a selected Transfer Table and an imported time series of wind or wave conditions, or spectral data, to generate an animation of offshore and inshore spectrum (frames are similar to the above image). The user is prompted to select the type of spectrum to use and the range of data to use. If the record selected has more than 5000 data points the user is asked to confirm. This is because the plotting is data intensive and can quickly create large arrays. However, it does provide a useful way of checking what is happening in the transfer process.

As with the single plot the user is prompted to select an X-Y plot or a polar plot. If the latter is used, there is some additional processing required to interpolate the data onto the polar surface (uses polarplot3d) and this can take some time.

### 3.7 Help

The help menu provides options to access the App documentation in the Matlab™ Supplemental Software documentation, or the App manual.

### 3.8 Tabs

To examine what has been set-up the Tabs provide a summary of what is currently defined. Note: the tabs update when clicked on using a mouse but values displayed cannot be edited from the Tabs.

**Cases:** lists the cases that have been run with a case id and description. Clicking on the first column of a row generates a table figure with details of the variables for the case and any associated metadata. Buttons on the figure provide access the class definition metadata and any source information (files input or models used).

**Inputs:** tabulates the system properties that have been set (display only).

**Q-Plot:** displays a quick-plot defined for the class of the selected case (display only).

**Stats:** displays a table of results for any analyses that have been run (can be copied to clip board).



### 3.9 UI Data Selection

Functions such as Derive Output (3.5), Plotting (3.6.1) and Statistics (3.6.2) use a standardised UI for data selection. The Case, Dataset and Variable inputs allow a specific dataset to be selected from drop down lists. One each of these has been set to the desired selection the choice is assigned to a button. The button varies with application and may be X, Y, Z, or Dependent and Independent, or Reference and Sample, etc. Assigning to the button enables further sub-sampling to be defined if required. Where an application requires a specific number of dimensions (e.g., a 2D plot), then selections that are not already vectors will need to be subsampled. At the same time, the range of a selected variable can be adjusted so that a contiguous window within the full record can be extracted. In most applications, any scaling that can be applied to the variable (e.g., linear, log, relative, scaled, normalised, differences) is also selected on this UI. The selection is defined in two steps:

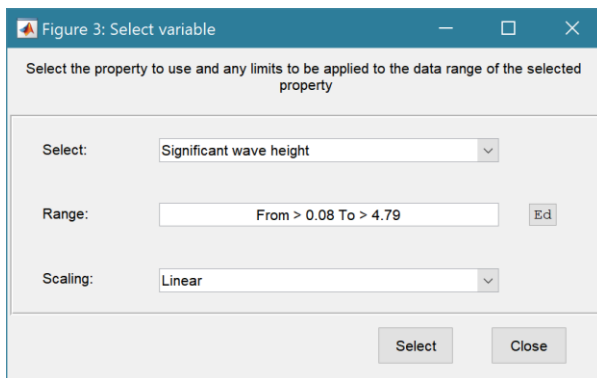


Figure 3: Select variable

Select the property to use and any limits to be applied to the data range of the selected property

Select: Significant wave height

Range: From > 0.08 To > 4.79 Ed

Scaling: Linear

Select Close

#### Step 1.

Select the attribute to use. This can be the variable or any of its associated dimensions, which are listed in the drop-down list.

The range for the selection can be adjusted by editing the text box or using the Edit (Ed) button.

Any scaling to be applied is selected from the drop-down list.

Press Select to go to the next step or Close to quit.

The number of variables listed on the UI depends on the dimensions of the selected variable. For each one Select the attribute to use and the range to be applied.

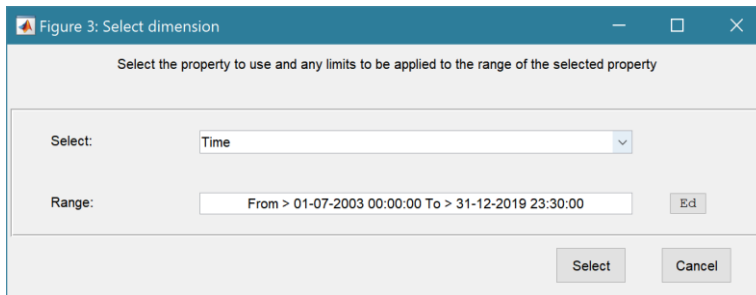


Figure 3: Select dimension

Select the property to use and any limits to be applied to the range of the selected property

Select: Time

Range: From > 01-07-2003 00:00:00 To > 31-12-2019 23:30:00 Ed

Select Cancel

#### Step 2 - Variable only has dimension of time.

No selection to be made.

Edit range if required.

Figure 4: Select dimension

Select the property to use and any limits to be applied to the range of the selected property

Select: X-axis

Range: From > 7900 To > 16900 Ed

Select: Y-axis

Range: From > 0 To > 16900 Ed

Time: 0 yrs 500 yrs 1000 yrs Ed

Select Cancel

## Step 2 - Variable has 3 dimensions but only 2 are needed for the intended use.

Select the 1<sup>st</sup> variable to use as a dimension.  
Edit range if required.

Select the 2<sup>nd</sup> variable to use as a dimension.  
Edit range if required.

Use the slider or the Edit (Ed) button to set the value of the dimension to use. (A value of t=500 is selected in the example shown).

Press Select to accept the selection made.

[NB: Only unused dimensions can be selected from the Select drop-down lists. To adjust from the default list this can sometimes require that the second Select list-box is set first to allow the first Select list-box to be set to the desired value.]

The resulting selection is then detailed in full (including the ranges or values to be applied to all dimensions) in the text box alongside the button being defined.

Note where a variable is being selected as one property and a dimension as a second property, any sub-selection of range must be consistent in the two selections. This is done to allow variables and dimensions to be used as flexibly as possible.

### 3.10 Accessing data from the Command Window

In addition to the options to save or export data provided by the **Project>Cases>Save** and **Project>Import/Export** options, data can also be accessed directly for use in Matlab™, or to copy to other software packages. This requires use of the Command Window in Matlab™, and a handle to the App being used. To get a handle, open the App from the Command Window as follows:

```
>> myapp = <AppName>; e.g., >> as = Asmita;
```

Simply typing:

```
>> myapp
```

Which displays the results shown in the left column below with an explanation of each data type in the right hand column.

myapp = <AppName> with properties:	Purpose
Inputs: [1×1 struct]	A struct with field names that match all the model parameter input fields currently
Cases: [1×1 muiCatalogue]	muuiCatalogue class with properties DataSets and Catalogue. The former holds the data the latter the details of the currently held records.
Info: [1×1 muiProject]	muiProject class with current project information such as file and path name.
Constants: [1×1 muiConstants]	muiConstants class with generic model properties (e.g. gravity, etc).

To access current model settings, use the following:

```
>> myapp.Inputs.<InputClassName>
```

To access the listing of current data sets, use:

```
>> cs.Cases.Catalogue
```

To access imported or model data sets, use:

```
>> cs.Cases.DataSets.<DataClassName>
```

If there are more than one instance of the model output, it is necessary to specify an index. This then provides access to all the properties held by that data set. Two of these may be of particular interest, `RunParam` and `Data`. The former holds the input parameters used for that specific model run.

`RunParam` is a struct with fields that are the class names required to run the model (similar to `Inputs` above). The `Data` property is a model specific struct with field names defined in the code for the model class. If there is only a single table assigned this will be given the field name of 'Dataset'. To access the *dstable* created by the model, use:

```
>> cs.Cases.DataSets.<DataClassName>(idx).Data.Dataset
```

```
>> cs.Cases.DataSets.<DataClassName>(idx).Data.<ModelSpecificName>
```

To access the underlying *table*, use:

```
>> cs.Cases.DataSets.<DataClassName>(idx).Data.Dataset.DataTable
```

The result can be assigned to new variables as required. Note that when assigning *dstable*s it may be necessary to explicitly use the copy command to avoid creating a handle to the existing instance and potentially corrupting the existing data.

## 4 Supporting Information

This section provides some background on the methods used in the quality control of data and the models included within WaveRayModel. Input files formats that are already defined are detailed in Appendix A – Input Data File Formats.

### 4.1 Quality Checks

Once the Data QC option has been run for a data set, ‘qc’ is displayed in the QC column on the right-hand side of the Data tab. If the Data QC is re-run this applies the quality checks to the currently saved timeseries and overwrites the values. The following checks are currently run when Data QC is selected.

#### 4.1.1 Waves

The CCO checks for Hs, Tz and Tp are as follows:

- Flag = 0 - all data pass
- Flag = 1 - either Hs or Tz fail, so all data fail
- Flag = 2 - Tp fail + derivatives
- Flag = 3 - Direction fail + derivatives
- Flag = 4 - Spread fail + derivatives
- Flag = 5 - Tp fail Jump test only (exclude on advice from Travis Morgan at CCO)
- Flag = 7 - Buoy adrift
- Flag = 8 - Sea Temperature fail
- Flag = 9 - Missing data (already assigned as NaN)

If any of flags 1-4 are set, the data are changed to NaN. In addition, the following checks are performed and again data that fail are set to NaN:

- ~  $0 > \text{Direction} < 360$
- ~  $\text{Hs} < \text{Hmax}$
- ~  $\text{Hs-Tz steepness} < 1/7$  (using deep water steepness)
- ~  $\text{Hmax-Tp steepness} < 1/7$  (ditto)
- ~  $\text{Tp} < \text{MaxTp}$  (defined in *Setup>Run parameters>Model constants* but the user is also prompted to confirm or adjust this value).

#### 4.1.2 Water levels

Some BODC file formats have an error flag for bad data. This is written as a 9 when the timeseries is loaded. Running the Data QC for water levels sets data values with a 9 flag to NaNs. In addition, a time series plot is generated and the user is prompted to define minimum and maximum values.

Values outside this range are assigned NaN values.

For CCO tidal data the flags used are loaded with the data. The flags are used in the Data QC tool to remove suspect data. Flags 3, 4, 8 and 9 are set to NaN when Data QC is applied.

- Flag = 1 correct value
- Flag = 2 interpolated
- Flag = 3 doubtful
- Flag = 4 isolated spike or wrong value
- Flag = 5 correct but extreme
- Flag = 6 reference change detected
- Flag = 7 constant value
- Flag = 8 out of range
- Flag = 9 missing

## 4.2 Grid and Mesh options

Imported data are loaded from a file as X,Y, Z tuples and define a Cartesian grid. The grid tools provide enable this initial grid to be manipulated (rotation, re-sampling, translation, etc). However, with very detailed source data (e.g., sub 10m) this can result in a very large grid. This can be sub-sampled to derive a more manageable and computationally efficient but this is at the expense of reduced grid resolution. An alternative is to construct a mesh which has varying sized triangular elements across the domain. Typically, these will have a more detailed grid in the shallower water near shore and larger element further offshore.

The option provided in the WaveRayModel App uses Mesh2D and is available to download from: <https://github.com/dengwirda/mesh2d>. Once installed (see Section 2.1.3), the mesh generator can be run using *Run>Create Mesh* (see Section 3.5).

The effect of the various parameter settings is illustrated in Appendix D – Mesh parameters. The mesh2d functions are called from the WRM\_Mesh class and specifically in the get\_mesh2d method. Some properties such as ‘kind’ and ‘disp’ are hard coded to ‘delfront’ and ‘inf’ respectively. Using delfront generates slightly more optimal meshes than the ‘delaunay’ alternative and disp turns the command window display off.

The other parameters that are enabled, are set when Create Mesh is called. Broadly these do the following:

Upper shore level, zhw, defines the elevation used to extract the shore boundary from the existing grid

Shore mesh size, minshore, defines the spacing of point sampling along the shoreline

Boundary mesh size, minbound, defines the spacing of point sampling long the open boundary

Maximum radius-edge ratio, rho2, refines the grid density and quality of mesh (minimum value =1)

Gradient limit, dhdx, adjusts mesh size across domain and boundaries (more influence on mesh size)

Edge-Element length threshold, siz1, refines the edge size along the boundary

Tria-Element length threshold, siz2, refines the internal domain (more influence on mesh size)

Adding internal points (with or without edges) is also an option and allows certain areas of the mesh to be controlled/refined.

## 4.3 Nearshore waves

Forward tracking wave rays as they approached the coast was originally done manually by coastal engineers to understand how energy was being focussed on the shore (USACE, 1984). This was replaced by computer simulations when a ray tracing method was proposed by Abernathy and Gilbert (Abernathy and Gilbert, 1975). Although this used equilateral triangles, it was found to be readily adapted for use with isosceles triangles derived from a Cartesian grid. The method has the further advantage that it can be run in reverse from a nearshore point (backward tracking) and used to construct a functional relationship between the offshore and nearshore wave spectra.

The fundamental relationship describing the curvature of a wave orthogonal, expressed in cartesian co-ordinates  $x$  and  $y$ , is:

$$\frac{d\alpha}{ds} = \frac{1}{c} \left( \frac{dc}{dx} \sin \alpha - \frac{dc}{dy} \cos \alpha \right) \quad (1)$$

where  $s$  is directed along the orthogonal and  $\alpha$  is the angle between the wave orthogonal and the  $x$ -axis. The wave speed,  $c$ , for sinusoidal waves of small amplitude, is given by the dispersion equation:

$$c = \frac{gT}{2\pi} \tanh\left(\frac{2\pi h}{cT}\right) \quad (2)$$

where  $T$  is the wave period,  $h$  the water depth and  $g$  the acceleration due to gravity.

A cartesian grid defines the seabed topography covering the entire area of interest. Starting from a known point and a known direction, Equation (1) may be solved repeatedly to calculate the path of the wave ray across the element. The solution technique used is a development of the "circular arc" method originally developed at the Hydraulics Research Station (Abernethy and Gilbert, 1975). Briefly, the method assumes that within a small triangular element the variation of wave speed is linear in both  $x$  and  $y$  coordinate directions. Using this assumption, the ray path within an element is described by a circular arc. This arc is centred on the zero celerity line in the local celerity plane and is tangential to the corresponding arcs in adjacent elements (Section 4.3.1).

The solution requires definition of the initial values so that wave rays may be tracked from any point to some unknown destination. Wave ray diagrams may be constructed by tracking several rays from the offshore boundary to cover the inshore area of interest. Whilst this method provides a good picture of the general wave refraction patterns in an area, associated methods of calculating wave refraction coefficients are unsatisfactory for several reasons which are discussed by Abernethy and Gilbert (1975).

An alternative to this method is to track wave rays "backwards" from an inshore point of particular interest to the offshore boundary of the model. This is done for a number of wave directions and periods from each point of interest. This allows complete two-dimensional frequency direction wave spectra to be transferred to the inshore points and has been found to give more stable and realistic. The spectrum refraction equation may be stated as:

$$S(f, \phi) = \frac{c_0 \cdot c_{g0}}{c \cdot c_g} S_0(f, \theta) \quad (3)$$

Where  $S$  is the spectrum function,  $f$  is the wave frequency in shallow water,  $\phi$  is the direction in deep water,  $f$  is the wave frequency ( $=1/T$ ),  $c_g$  is the wave group velocity and subscripts 0 refer to deep water.

The wave refraction model output associates a deepwater wave direction with each inshore wave direction for each period considered. Using a suitable spectral description of the deepwater wave conditions, the energy density associated with the deepwater end of each wave ray may be transferred to the inshore spectrum at the appropriate direction and period. Equation (3) is used to modify the energy density to account for refraction effects. From the inshore wave spectrum, relevant quantities such as wave height, mean direction and frequency may be calculated (Section 4.3.2).

The offshore two-dimensional frequency-direction spectrum is usually assumed to be the product of two mutually independent components  $S'(f)$  and  $G(\theta)$  for which the former is a one dimensional frequency spectrum and the latter is a suitable direction distribution function. A number of well-known spectra are provided for, as summarised in Section 4.3.3. Similarly different definitions of the directional spreading of the spectrum are provided and these are outlined in Section 4.3.4.

The model set-up uses a range of wave periods and water levels to define a ray parameter space with dimensions of [number of rays, period, water level] for forward tracking and [inshore direction, period, water level] for backward tracking. The wave periods are defined at log10 (frequency) intervals and the water levels are spaced linearly over the specified range. The log10 frequency spacing provides a more efficient coverage of the spectrum, thereby reducing the number of ray sets

that need to be run. However, the range and interval needs to be selected to adequately cover the range of peak periods in the wave climate being studied. The following are some indicative settings:

T=4-18s, n=8: 4.0, 4.5, 5.1, 6.0, 7.2, 9.0, 12.0, 18.0.

T=4-20s, n=10: 4.0, 4.4, 4.8, 5.5, 6.2, 7.2, 8.6, 10.6, 13.8, 20.0.

T=5-25s, n=9: 5.0, 5.6, 6.3, 7.1, 8.3, 10.0, 12.5, 16.7, 25.0.

Note: T≤3s are assumed to be approximated by minimal refraction and are added by the code.

#### 4.3.1 Ray tracing

A detailed derivation of the method is given by Abernethy and Gilbert (1975). The following summarises the key components used to implement the ray tracing in the WaveRayModel.

$$\text{Ray trajectory equation: } \frac{d\alpha}{ds} = -\mathbf{N} \cdot \frac{\nabla c}{c} \quad (4)$$

$$\text{Ray separation equation: } c \frac{d\gamma/c}{ds} + \frac{\beta}{c} \frac{dc}{dN^2} = 0 \quad (5)$$

$\mathbf{N}$ ,  $\mathbf{T}$  are unit vectors normal and tangential to a ray at  $\mathbf{r}_a$

$\alpha$  is the bearing of the ray w.r.t the reference direction

$c$  is the wave celerity and  $\tilde{N}c$  is the gradient of the celerity

$b = b_1/b_0$  where  $b$  is the distance between rays.

$\gamma = b\mathbf{T}/c$

$$\text{Radius of arc is given by } R = \frac{-c}{\mathbf{N} \cdot \nabla c} \quad (6)$$

$$\text{If a ray enters an element at } \mathbf{r}_a, \text{ then the centre of the arc is at } \mathbf{r}_c = \mathbf{r}_a + R\mathbf{N} \quad (7)$$

The iterative process is to identify where a ray enters and exits an element, find the next element and repeat the process. Finding the correct exit point, adjusting the coordinates of the local grid origin and handling rays that lie on a grid node, or are travelling along a grid axis, add a little complexity.

#### 4.3.2 Spectral transfer

The inshore wave height, mean direction and period may be determined by integrating to obtain the moments of the inshore spectrum. The above parameters are then obtained from these, as follows:

$$k_w = \left( \frac{\iint S(f, \phi) df d\phi}{\iint S_0(f, \theta) df d\theta} \right)^{1/2} \quad (7)$$

$$\bar{\phi} = \left( \frac{\iint \phi \cdot S(f, \phi) df d\phi}{\iint S(f, \phi) df d\phi} \right) \quad (8)$$

$$T_2 = \left( \frac{\iint S(f, \phi) df d\phi}{\iint f^2 \cdot S(f, \phi) df d\phi} \right)^{1/2} \quad (9)$$

$k_w$  expresses the inshore wave height as a proportion of the offshore wave height, making the refraction study independent of the deepwater wave height in this simple case.  $\bar{\phi}$  is the mean direction of the inshore spectrum and  $T_2$  is the mean period of the inshore spectrum. The following are saved in the output table:

Inshore wave height:  $H_{si} = k_w \cdot H_{so}$

Inshore mean period:  $T_2$ , equation (9)

Inshore wave direction:  $\bar{\phi}$  equation (8)

Inshore peak period: wave period of maximum value of inshore spectrum

Inshore peak direction: wave direction of maximum value of inshore spectrum

Wave transfer coefficient:  $k_w$ , equation (7)

Mean period coefficient:  $k_{t2} = T_{2i}/T_{2o}$

Peak period coefficient:  $k_{tp} = T_{pi}/T_{po}$

Mean direction shift:  $k_d = \bar{\phi} - \bar{\theta}$

Still water level – still water level coincident with wave record

Inshore depth – depth at inshore point based on still water level.

### 4.3.3 Wave spectra

An excellent summary of the most common wave spectra in common use is provided by Carter (Carter, 1982). The four spectra included are Bretschneider open ocean, Pierson-Moskowitz fully developed, JONSWAP fetch limited, and TMA shallow water. These are detailed below:

#### Bretschneider open ocean

The Bretschneider spectrum was developed to represent open ocean wave conditions and the spectral energy as a function of frequency,  $f$ , has the following form:

$$S(f) = 0.31 H_s^2 \frac{1}{f_p} \left( \frac{f}{f_p} \right)^{-5} \exp \left( -1.25 \left/ \left( \frac{f}{f_p} \right)^4 \right. \right) \quad (10)$$

where  $H_s$  is the significant wave height,  $f_p$ , is the peak frequency of the spectrum ( $1/T_p$ ).

#### Pierson-Moskowitz

The data for the spectrums of fully developed seas for wind speeds from 20 to 40 knots (10.29 to 20.58 m/sec), were used to identify a power spectrum for fully developed seas (Pierson Jr. and Moskowitz, 1964). The spectral energy as a function of frequency,  $f$ , has the following form:

$$S(f) = \frac{\alpha g^2}{(2\pi)^4 f^5} \exp \left( -\frac{5}{4} \left( \frac{f}{f_p} \right)^4 \right) \quad \text{where } \alpha = 0.0081 \quad (11)$$

Given the significant wave height this can be written as:

$$S(f) = 5.10^{-4} f^{-5} \exp \left( -2.10^{-3} / H_s^2 f^4 \right) \quad (12)$$



Given the peak wave period this can be written as:

$$S(f) = 5.10^{-4} f^{-5} \exp\left(-1.25/(T_p f)^4\right) \quad (13)$$

### JONSWAP fetch limited

The JONSWAP formulation is used to represent the deepwater spectrum describes developing seas, which is appropriate for most coastal sites around the UK and has following functional form:

$$S(f) = \frac{\alpha g^2}{(2\pi)^4 f^5} \exp\left(-\frac{5}{4}\left(\frac{f}{f_p}\right)^4\right) \gamma^q \quad (14)$$

$$\text{where } q = \exp\left(-\frac{(f - f_p)^2}{2\sigma^2 f_p^2}\right)$$

and  $\sigma$  is 0.07 when  $f < f_p$  and 0.09 when  $f > f_p$ ,  $\alpha$  is a constant obtained either from measured offshore spectra, or by parameterisation,  $f_p$  is the peak frequency of the spectrum and  $g$  is the acceleration due to gravity. Comparing equations (10) and (11) it is clear that the only differences are the additional term  $\gamma^q$  and the definition of  $\alpha$ . Using the integral moments of the spectrum the value of  $\alpha$  can be estimated from  $H_s$  using the following expression:

$$\alpha = \frac{H_s^2 (2\pi)^4 f_p^4}{16 g^2 I_0(\gamma)} \quad (15)$$

in which  $I_0(\gamma)$  is zero moment integral. Typically,  $\gamma$  is given a value of 3.3 but can range from 1-7.

### TMA shallow water

The TMA spectrum modifies the JONSWAP spectrum to take account of shallow water effects (depth saturation of the energy spectra). To correct for depth-dependent effects, Bouws et al. (1985) also manipulated the linear term  $S(f)$  to reflect the loss of energy due to enhanced dissipation of shallow water. They replace  $S(f)$  with  $S_k(f, h)$ , where  $h$  is the water depth:

$$S_k(f, H) = \frac{\alpha g^2 \varphi_k(\omega_h)}{(2\pi)^4 f^5}; \quad \varphi_k(\omega_h) = \frac{\left[ (k(\omega, h))^{-3} \frac{\partial k(\omega, h)}{\partial f} \right]}{\left[ (k(\omega, \infty))^{-3} \frac{\partial k(\omega, \infty)}{\partial f} \right]} \quad \text{and } \omega_h = 2\pi f \sqrt{h/g} \quad (16)$$

in which  $k$  is the wave number ( $1/L$ ) and other variables are as defined for the JONSWAP spectrum. The solution of the partial derivatives is detailed in Kitaigorodskii et al. (1975) and Bouws et al. give the following approximate solution:

$$\begin{aligned} \varphi_k &\approx 0.5 \cdot \omega_h^2 & \text{for } \omega_h \leq 1 \\ \varphi_k &\approx 1 - 0.5 \cdot (2 - \omega_h)^2 & \text{for } 1 < \omega_h \leq 2 \\ \varphi_k &\approx 1 & \text{for } \omega_h > 2 \end{aligned} \quad (17)$$

Such that:  $S_k(f, H) = S(f) \cdot \varphi_k$  where  $S(f)$  is as defined for the JONSWAP spectrum.

When using wind input the peak frequency is estimated from the peak period using the equation proposed by Donelan (1985):

$$T_p = 0.54 \cdot \frac{U^{0.54} F^{0.23}}{g^{0.77}} \quad (18)$$

$$k_p = \frac{2\pi U^2}{gL_p}; \quad \alpha = 0.078k_p^{0.49} \quad \text{and} \quad \gamma = 2.47k_p^{0.39}$$

where  $U$  is the wind speed at 10m (m/s),  $F$  is the fetch length (m),  $g$  is the acceleration due to gravity (m/s<sup>2</sup>) and  $L_p$  is the wavelength of the peak period.

The wavelength is calculated using the celerity of each frequency for the average depth along the rays. To compute the offshore spectrum the offshore depth is used to determine the saturation (Eq.17), whereas the minimum depth is used to determine the inshore spectrum. In all cases the depth is based on the weighted value for all rays using the directional spreading function (Eq.19), to weight the relevant depth value for each ray based on the mean direction of the wave condition.

#### 4.3.4 Direction spreading

Two methods are available for calculating the effective fetch, as proposed in the Shore Protection Manual and by Donelan (1985).

The functional form of the cosine directional distribution used in the SPM is:

$$G(\theta) = \frac{\cos^n(\beta(\theta - \bar{\theta}))}{\int_{-\pi/2}^{\pi/2} \cos^n(\beta(\theta - \bar{\theta}))d\theta} \quad (19)$$

where  $\bar{\theta}$  is the mean direction of the spectrum,  $\beta$  is a constant and the exponent  $n$  is typically in the range 2-10. An exponent of 2 corresponds to a broad banded distribution which can be associated with locally wind-generated waves, whilst a value of 10 corresponds to a relatively narrow-banded distribution which can be associated with seas generated over a long fetch.

For the SPM method, the trigonometric function is a cosine and the constant,  $\beta$ , is 1.

For the Donelan formulation, developed for wind-wave generation in lakes where the fetches can be long and narrow, the trigonometric function is hyperbolic secant  $\text{sech}(z) = 1/(\cosh(z))$  rather than a cosine and the constant,  $\beta$ , is 2.28.

#### 4.3.5 Wave buoy directional spreading

The SPT file format from the Datawell wave buoy returns spectral energy and directional statistics as a function of 64 frequency bins. The directions properties available in the file are the mean direction, spread, skew and kurtosis. Equations 14.2.24 and 14.2.27 in the Datawell Waves5 Reference Manual, 2023, p224-5 describe how to use these parameters to reproduce an approximation of the directional spreading function. Kuik et al. (1988) works with a similar set of parameters, to those included in the SPT Format output file, namely the mean direction  $Dir$ , the directional spread  $Spr$ , the skewness  $Skew$  and the kurtosis  $Kurt$ . These parameters can also be used to compute the centred Fourier Coefficients as follows:

$$\theta_0 = Dir. \frac{\pi}{180}$$

$$m_1 = 1 - \frac{\sigma^2}{2}, \text{ where } \sigma = Spr. \frac{\pi}{180}$$

$$m_2 = \frac{1}{2} (Kurt. \sigma^4 - 6 + 8m_1)$$

$$n_2 = -Skew \left( \frac{1}{2} (1 - m_2) \right)^{3/2}$$

The directional distribution can then be described in terms of these centred Fourier coefficients as follows:

$$D(\theta, f) = \frac{1}{\pi} \left\{ \frac{1}{2} + m_1 \cos(\theta - \theta_0) + m_2 \cos 2(\theta - \theta_0) + n_2 \sin 2(\theta - \theta_0) \right\}$$

## 4.4 Output tables

Summary of the data held in the output tables for each of the models.

### 4.4.1 Ray models

Forward and Backward ray models are held in a dstable with the following variables:

Name	Description	Units
xr	X-Position	m
yr	Y-Position	m
alpha	Direction	rad
depth	Water depth	m
celerity	Celerity	m/s
cgroup	Group Celerity	m/s

For forward rays there is a table row for each ray start position. For backward rays there is a row for each inshore direction and the table has dimensions of wave period and water level. As the number of points in each ray can vary, the variables for each combination of ray number/direction, period and water level are held as a cell (i.e., each row/variable has a [1 x nper x nwls] cell array).

### 4.4.2 Transfer Table

The transfer table is constructed from a set of backward tracking rays with the following variables:

The Inshore table contains the wave celerity and group celerity at the inshore point, with dimensions period and water level. The Offshore table contains the following variables:

Name	Description	Units
theta	Offshore Direction	degTN
depth	Offshore depth	m
celerity	Celerity	m/s
cgroup	Group Celerity	m/s
avdepth	Average depth	m

mindepth	Minimum depth	m
----------	---------------	---

Each row of the table is an inshore wave direction and the table has dimensions of wave period and water level for each variable (i.e., an [ndir x nper x nwls] array for each variable).

#### 4.4.3 Wave transfer model outputs

##### Wave properties

When a transfer table is used to create a new timeseries at the point used for the backtracking from an offshore timeseries, the following variables are saved:

Name	Description	Units
Hsi	Inshore wave height	m
T2i	Inshore mean period	s
Diri	Inshore wave direction	degTN
Tpi	Inshore peak period	s
Diripk	Inshore peak direction	degTN
kw	Wave transfer coefficient, $H_{si}/H_{so}$	-
kt2	Mean period coefficient, $T_{2i}/T_{2o}$	-
ktp	Peak period coefficient, $T_{pi}/T_{po}$	-
kd	Mean direction shift, $\phi_i - \theta_o$	deg
swl	Still water level	mOD
depi	Inshore depth	m

Each row of the table is a time taken from the input wave data set.

##### Wave spectra

Name	Description	Units
So	Offshore wave spectra	m <sup>2</sup> /Hz
Si	Inshore wave spectra	m <sup>2</sup> /Hz

Each row of the table is a time record and the table has dimensions of direction and frequency for each variable (i.e., an [ndir x nfreq] array for each variable).

#### 4.5 Model and program assumptions

In implementing the model, the following programming selections have been used:

- The resolution of the spectral transfer table depends on the number of inshore directions, periods and water levels that were defined for the backward ray tracing model run.
- When rays are terminated at their “offshore” end because they are in shallow water (i.e. they have refracted on to the shore) they are assigned a stop flag. This is applied when constructing the spectral transfer table to exclude rays that terminate at the shore (offshore direction is set to NaN, depth and celerity set to zero).
- To construct the offshore frequency-direction spectrum the spectral transfer table is interpolated to directions at 0.5 degree intervals and wave periods at 0.5s intervals for the selected water level.

- The Transfer Coefficients plots use the modelled direction, period and water level intervals and the user defined conditions (type of spectrum, etc) with a wave height of 1m to construct the output arrays to be plotted.
- The Transfer Table plots use the modelled direction, period and water level intervals and the user defined conditions (type of spectrum, etc).

Model assumptions include the following:

- The model assumes that the seabed is slowly varying the grid resolution is such that the celerity gradient can be assumed to be constant in any given triangular element.
- In this implementation the occurrence of caustics (rays that cross) is ignored. In backward tracking mode this is generally an acceptable approximation because the inshore spectrum is the integral of all directions. Hence the energy associated with a given caustic only forms a small fraction of the total incident energy.
- The JONSWAP spectrum has an empirical constant, gamma, which needs to be estimated and the TMA spectrum as implemented, uses the approximation of Bouws et al. (1985).
- When reconstructing spectra from measured spectral data sets (e.g., those available from the Channel Coastal Observatory), the directional distribution is obtained using the method set out in the Datawell Reference Manual (Datawell, 2023).
- The direction spreading function proposed by Donelan has been implemented using a constant value of  $\beta$  but this could be made frequency dependent as detailed in the original publication (Donelan et al., 1985).

## 4.6 Taylor diagram

The basis of the Taylor Diagram and associated skill score is explained in Taylor (2001) and summarised here.

### 4.6.1 Taylor diagram theory

For a variable, X we denote:

Bias as: 
$$\bar{E} = \bar{X}_{model} - \bar{X}_{obs}$$

Centred Root Mean Square Difference (or Error) as:

$$E' = \sqrt{\frac{\sum \{(X_{model} - \bar{X}_{model}) - (X_{obs} - \bar{X}_{obs})\}^2}{N}}$$

to give the total Mean Square Difference as; 
$$E^2 = \bar{E}^2 + E'^2$$

As  $E' \rightarrow 0$ , patterns become similar, so it is not possible to determine how much of the error is due to a difference in structure and phase and how much is simply due to a difference in the amplitude of the variations.

Standard deviations for model and observed data sets are given by:

$$\sigma_{model} = \sqrt{\sum (X_{model} - \bar{X}_{model})^2 / N}$$

$$\sigma_{obs} = \sqrt{\sum (X_{obs} - \bar{X}_{obs})^2 / N}$$

and the Correlation coefficient is given by:

$$R = \frac{\sum (X_{model} - \bar{X}_{model})(X_{obs} - \bar{X}_{obs})}{N\sigma_{model}\sigma_{obs}}$$

These measures are related as follows:

$$E'^2 = \sigma_{\text{model}}^2 + \sigma_{\text{obs}}^2 - 2\sigma_{\text{model}}\sigma_{\text{obs}}R$$

And by defining  $R$  as  $\cos(\phi)$  this has the form of the law of cosines equation, where the mean square error and two standard deviations form the sides of a triangle with angle  $\phi$  between the two sides defined by standard deviations.

When comparing different metrics, it can be convenient to use the above in a normalised form, where the normalised variables are denoted by a hat.

Normalised RMS difference: 
$$\hat{E}' = \frac{E'}{\sigma_{\text{obs}}}$$

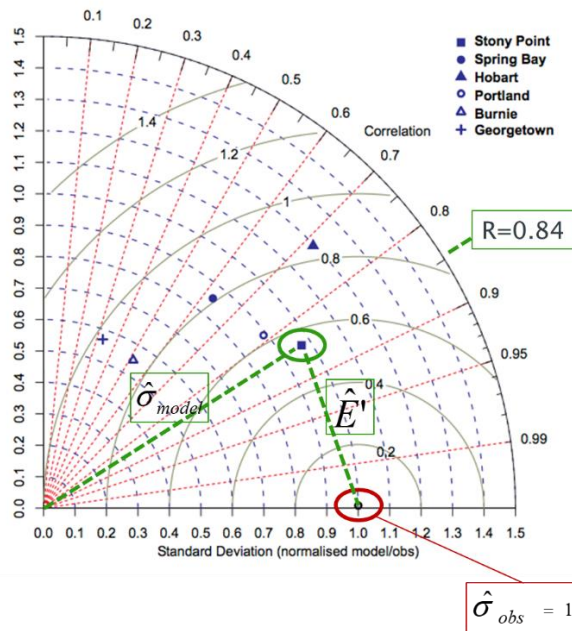
Normalised Standard deviations: 
$$\hat{\sigma}_{\text{model}} = \frac{\sigma_{\text{model}}}{\sigma_{\text{obs}}}$$

$$\hat{\sigma}_{\text{obs}} = 1$$

Leading to: 
$$\hat{E}'^2 = \hat{\sigma}_{\text{model}}^2 + \hat{\sigma}_{\text{obs}}^2 - 2\hat{\sigma}_{\text{model}}\hat{\sigma}_{\text{obs}}R$$

whence: 
$$\hat{E}'^2 = 1 + \hat{\sigma}_{\text{model}}^2 - 2\hat{\sigma}_{\text{model}}R$$

The figure below shows the form of the resultant Taylor diagram for one data set.



#### 4.6.2 Skill score

Taylor defines the basis of a good skill score as:

*“For any given variance the score should increase monotonically with increasing correlation, and for any given correlation the score should increase as the modelled variance approaches the observed variance. Traditionally, skill scores have been defined to vary from zero (least skilful) to one (most skilful).”*

He then proposes a skill score that achieves this without being too complicated as:

$$S = \frac{4(1+R)}{(\hat{\sigma}_{\text{model}} + 1/\hat{\sigma}_{\text{model}})^2 (1+R_0)}$$

where  $R_0$  is the maximum correlation attainable and  $S=0$ -poor skill;  $S=1$ -good skill. This option weights the pattern variance. However, this is just a model and the weighting can be varied based on what is considered to be most important for the application. E.G. to increase penalty for low correlation, the equation of the following form could be adopted:

$$S = \frac{4(1+R)^4}{(\hat{\sigma}_{\text{model}} + 1/\hat{\sigma}_{\text{model}})^2 (1+R_0)^4}$$

Bosboom et al (2014a; 2014b) use similar variations on this basic structure.

In ModelSkill this is implemented as:

$$S = \frac{4(1+R)^n}{(\hat{\sigma}_{\text{model}} + 1/\hat{\sigma}_{\text{model}})^2 (1+R_0)^n}$$

The exponent,  $n$ , and the maximum correlation attainable,  $R_0$ , are defined in Run Parameters.

The foregoing provides an estimate based on values (mean, standard deviation, etc) for the full grid domain; a global skill score, 'Sg'. Bosboom and Reniers (2014) also make use of spatially local estimates of the skill score, 'Sl', which they obtain by applying a weighting based on distance from each point, when computing the mean, standard deviation and correlation. They then average the local skill score estimates, to obtain an average local skill score. When this is done based on the grid points for the surrounding cells (where the window,  $W$ , is the number of grids cells either side of the central point that are to be included) and the weighting is constant across this sub-grid, this is the same as simply sub-sampling the grid. A similar one-dimensional window can be applied to a vector when comparing timeseries data. This is the approach adopted in Coastaltools. The average skill score,  $Sl$ , is added to the metadata for the test point in the Taylor diagram and can be viewed by selecting the 'Case List' button on the Taylor Diagram figure

## 4.7 Derive Output

The *Run> Derive Output* option allows the user to make use of the data held within App to derive other outputs or, pass selected data to an external function (see Section 3.5). The equation box can accept  $t$ ,  $x$ ,  $y$ ,  $z$  in upper or lower case. Time can be assigned to  $X$ ,  $Y$ , or  $Z$  buttons, or simply included in the equation as  $t$  (as long as the data being used in one of the variables includes a time dimension). Each data set is sampled for the defined data range. If the data set being sampled includes NaNs, the default is for these to be included (button to right of Variable is set to '+N'). To exclude NaNs press the button so that it displays '-N'. The selection is based on the variable limits defined whenever a variable is assigned to  $X$ ,  $Y$  or  $Z$  using the  $X$ ,  $Y$ ,  $Z$  buttons.

The equation string entered in the UI is used to construct an anonymous function as follows:

```
heq = str2func(['@(t,x,y,z,mobj) ',inp.eqn]); %handle to anonymous function
```

```
[varout{:}] = heq(t,x,y,z,mobj);
```

or when using dstables:

```
heq = str2func(['@(dst,mobj) ',inp.eqn]); %handle to anonymous function
```

```
[varout{:}] = heq(dst,mobj);
```



This function is then evaluated with the defined variables for  $t$ ,  $x$ ,  $y$ , and  $z$  and optionally  $mobj$ , where  $mobj$  passes the handle for the main UI to the function. Some functions may alter the length of the input variables ( $x$ ,  $y$ ,  $z$ ,  $t$ ), or return more than one variable. In addition, the variables selected can be sub-sampled when each variable is assigned to the X, Y, or Z buttons. The dimensions of the vector or array with these adjustments applied need to be dimensionally correct for the function being called. This may influence how the output can be saved (see Section 4.7.2).

If the function returns a single valued answer, this is displayed in a message box, otherwise it is saved, either by adding to an existing dataset, or creating a new one (see Section 4.7.2 and 3.5).

*NB1: functions are forced to lower case (to be consistent with all Matlab functions), so any external user defined function call must be named in lower case.*

Equations can use functions such as  $\text{diff}(x)$  - difference between adjacent values - but the result is  $n-1$  in length and may need to be padded, if it is to be added to an existing data set. This can be done by adding a NaN at the beginning or the end:

e.g.: `[NaN;diff(x)]`

NB: the separator needs to be a semi-colon to ensure the correct vector concatenation. Putting the NaN before the equation means that the difference over the first interval is assigned to a record at the end of the interval. If the NaN is put after the function, then the assignment would be to the records at the start of each interval.

Another useful built-in function allows arrays to be sub-sampled. This requires the array,  $z$ , to be multiplied by an array of the same size. By including the dimensions in a unitary matrix, the range of each variable can be defined. For a 2D array that varies in time one way of doing this is:

```
>> [z.*repmat(1, length(t), length(x), length(y))]
```

*NB2: the order of the dimensions  $t$ ,  $x$ ,  $y$  must match the dimensions of the array,  $z$ .*

*NB3: When using Matlab compound expressions, such as the above sub-sampling expression, the expression must be enclosed in square brackets to distinguish it from a function call.*

Adding the comment `%time` or `%rows`, allows the the row dimension to be added to the new dataset. For example if  $x$  and  $y$  data sets are timeseries, then a Matlab™ expresion, or function call, can be used to create a new time series as follows:

```
x^2+y %time
```

#### 4.7.1 Calling an external function

The Derive Output UI can also be used as an interface to user functions that are available on the Matlab search path. Simply type the function call with the appropriate variable assignment and the new variable is created. (NB: the UI adopts the Matlab convention that all functions are lower case). Some examples of functions provided in WaveRayModel are detailed in Section 4.7.3.

The input variables for the function must match the syntax used for the call from the Derive Output UI, as explained above. In addition, functions can return a single value, one or more vectors or arrays, or a dstable (see Section 4.7.2). If the variables have a dimension (e.g., *time*) then this should be the first variable, with other variables following. If there is a need to handle additional dimensions then use the option to return a dstable.

If there is no output to be passed back, the function should return a variable containing the string 'no output' to suppress the message box, which is used for single value outputs (numerical or text).



An alternative when calling external functions is to pass the selected variables as dstables, thereby also passing all the associated metadata and RowNames for each dataset selected. For this option up to 3 variables can be selected and assigned to the X, Y, Z buttons but they are defined in the call using *dst*, for example:

```
[time,varout] = myfunction(dst, 'usertext', mobj);

dst = myfunction(dst, 'usertext', mobj);
```

where 'usertext' and mobj are call strings and a handle to the model, respectively.

This passes the selected variables as a struct array of dstables to the function. Using this syntax, the function can return a dstable or struct of dstables, or as variables, containing one or more data sets.

## 4.7.2 Input and output format for external functions

There are several possible use cases:

### 4.7.2.1 Null return

When using a function that generates a table, plots a figure, or some other stand alone operation, where the function does not return data to the main UI, the function should have a single output variable. The output variable can be assigned a text string, or 'no output', if no user message is required, e.g.:

```
function res = phaseplot(x,y,t,labels)
...
res = {'Plot completed'}; %or res = {'no output'}; for silent mode
...
end
```

### 4.7.2.2 Single value output

For a function that may in some instances return a single value this should be the first variable being returned and can be numeric or text, e.g.:

```
function [qtime,qdrift] = littoraldriftstats(qs,tdt,varargin)
...
%Case 1 - return time and drift
qdttime = array1;
qdrift = array2;
%Case 2 - return summary value
qtime = mean(array2); %return single value
%Case 3 - return summary text
qtime = sprintf('Mean drift = %.1f',mean(array2)); %return test string
...
end
```

### 4.7.2.3 Using variables

If only one variable is returned (length>1), or the first variable is empty and is followed by one or more variables, the user is prompted add the variables to:

- i) Input Cases – one of the datasets used in the function call;
- ii) New Case – use output to define a new dataset;
- iii) Existing Case – add the output to an existing dataset (data sets for the selected existing case and the data being added must have the same number of rows.

In each case the user is prompted to define the properties for each of the variables.

**Note** that variable names and descriptions must be unique within any one dataset.

```
function y = moving(x,m,fun)
    %a single variable is returned with no rows
    y is a vector or array
    ...
end

or

function [x,y,z] = afunction(x,m,fun)
    %multiple variables returned but the first variable is empty
    x = [];
    y and z are a vectors or arrays
    ...
end
```

When the first variable defines the rows of a table and subsequent variables the table entries, all variables must be the same length for the first dimension. This is treated as a new Case and the user is prompted to define the properties for each of the variables.

```
function [trange,range,hwl,lwl] = tidalrange(wl,t,issave,isplot)
    %first variable is row dimension followed by additional variables
    trange,range,hwl,lwl are vectors or arrays
    ...
end
```

#### 4.7.2.4 Using dstables

When the output has multiple variables of a defined type it can be more convenient to define the dsproperties within the function and return the data in a dstable. This avoids the need for the user to manually input the meta-data properties. In addition, if the function generates multiple dstables, these can be returned as a struct, where the struct fieldnames define the Dataset name.

```
function dst = tidalrange(wl,t,issave,isplot)
    %dst is a dstable with variables, dimensions and dsproprties assigned
    %as required, or a struct of dstables with the struct fieldnames defining
    %each Dataset.
    dst = ...
    ...
end
```

Similarly if the input is also using dstables, the syntax is as follows:

```
function dst_out = myfunction3(dst_in,'usertext',mobj)
    %dst_in is one or more input dstables, 'usertext' is some additional
    %instruction to the function and mobj is a handle to the model
    %allowing access to other datasets. dst_out is either a dstable, or a
    %struct of dstables with the struct fieldnames defining each Dataset.
    dst = ...
    ...
end
```

### Adding functions to the Function library

To simplify accessing and using a range of functions that are commonly used in an application, the function syntax can be predefined in the file `functionlibrarylist.m` which can be found in the `utils` folder of the `multoolbox`. This defines a struct for library entries that contain:

- `fname` - cell array of function call syntax;
- `fvars` - cell array describing the input variables for each function;
- `fdesc` - cell array with a short description of each function.

New functions can be added by simply editing the struct in `functionlibrarylist.m`, noting that the cell array of each field in the struct must contain an entry for the function being added. In addition, a sub-selection of the list can be associated with a given App based on the class name of the main UI. To amend the selection included with an App or to add a selection for a new App edit the '`switch classname`' statement towards the end of the function.

The Function button on the Derive Output UI is used to access the list, select a function and add the syntax to the function input box, where it can be edited to suit the variable assignment to the XYZ buttons.

#### 4.7.3 Pre-defined functions

The following examples are provided within WaveRayModel, where the entry in the UI text box is given in Courier font and X, Y, Z, refer to the button assignments.

Some useful examples primarily for timeseries data include:

1. **Moving Average.** There are several moving average functions available from the Matlab Exchange Forum, such as `moving.m`. The call to this function is:

`moving(X, n, 'func')`, where `x` is the variable to be used, `n` specifies the number of points to average over and '`func`' is the statistical function to use (e.g. mean, std, etc). If omitted the *mean* is used. Add `%time` to the call, to include time in the output dataset.

2. **Moving average (or similar) of timeseries**, over defined duration, advancing at defined interval

`movingtime(x, t, tdur, tstep, 'func')`, where `x` is the variable to be used and `t` the associated datetimes (defined by variable selection), `tdur` is the duration over which to apply the statistic, `tstep` is the interval to advance the start time for the averaging period and '`func`' is the statistical function to use (e.g. mean, std, etc). If omitted the *mean* is used. `tdur` and `tstep` are both duration character strings of form '`2.5 d`'. Any of the following duration intervals can be used: y, d, h, m, or s. Returns a time series based on the defined `tstep`, where the time used is for the beginning of each stepping interval, i.e. every `tstep` from the start of the record to the nearest interval that is less than `tdur` from the end of the record.

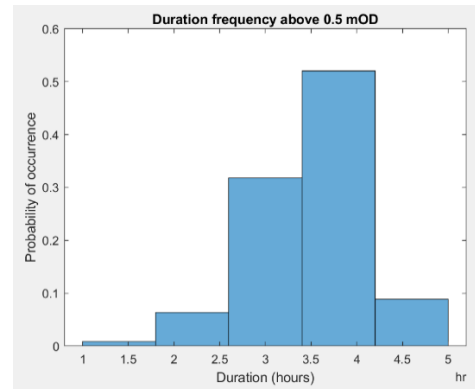
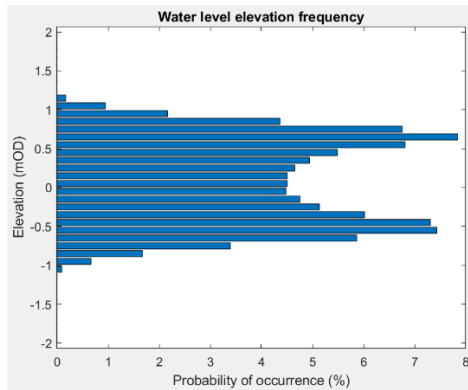
3. **Down-sampling a time series.** This allows a timeseries to be resampled at a different interval (that must be less than the source timeseries). The call to this function is:

`downsample(x, t, 'period', 'method')`, where `x` is the variable to be resampled, `time` is the associated time for that variable, `period` can be 'year', 'month', 'day', 'hour', 'minute', 'second', and `method` can be any valid function call such as 'mean', 'std', etc. The 'period' is required but the 'method' is optional and if omitted the mean is used.

For timeseries with gaps the 'nanmean' function is particularly useful but requires the Statistics toolbox.

4. **Interpolate and add noise.** To infill a record with additional points and, if required, add some random noise to the interpolated values. This is called using:  
`interpwithnoise(x, t, npad, scale, method, ispos)` , where X is the variable, t is time, npad is the number of points to add between the existing data points, scale determines the magnitude of the random noise (a value of 0 results in an interpolated record with no noise), method is the Matlab algorithm used for the interpolation (the default is linear) and ispos is a true/false flag which sets negative values to zero if true.
5. **Subsample one record at the time intervals of another record** (e.g. subsample water levels to be at the same intervals as the wave data). Function is:  
`subsample_ts(X, t, mobj)`, where X and t are the variable to be subsampled and *mobj* is the UI handle (must be *mobj*). The user is prompted to select the dataset to be used to define the time intervals. A time series is returned and added as a Derived data set. The user is prompted to define the metadata for the new data set.
6. **Subsample one record based on a threshold defined for another record** (e.g. subsample waves based on a threshold water level). Function is:  
`subsample(X, t, thr, mobj)`, where X and t are the variable to be subsampled, *thr* is the threshold value and *mobj* is the UI handle (must be *mobj*). The user is prompted to select the dataset and variable to be used to define the condition and a condition operator ( $\leq$ ,  $=$ , etc). A time series is returned and added as a Derived data set. The user is prompted to define the metadata for the new data set.
7. **Phase plot.** This function is similar to the recursive plot function but generates a plot based on two variables that can, optionally, be functions of time. The call to this function is:  
`phaseplot(X, Y, t)`, where X and Y are the variables assigned to the respective buttons and t is time (this does not need to be assigned to a button and t can be omitted if a time stamp for the datapoints is not required).
8. **Recursive plot.** Generates a plot of a variable plotted against itself with an offset (e.g.  $x(i)$  versus  $x(i+1)$ ). This is called from the Derive Output GUI using:  
`recursive_plot(x, 'varname', nint)`, where x is the variable, 'varname' is a text string in single quotes and *nint* is an integer value that defines the size of the offset.
9. **Add sea level rise to tidal water levels** (ie typically predictions rather than measured water levels). Based on exponential growth from 1900 and zeroed to a defined year using:  
`addslrtotides(X, t, delta, exprate, pivotyear)`, where X and t relate to the water level variable to be adjusted, delta is a rate for the year 1900 (e.g. 0.001 m/yr), *exprate* is the rate of exponential growth (e.g 0.011 for a fit to observations to-date) *pivotyear* is the year to use for zero sea level rise (e.g. 1900 adds slr based on change since 1900, whereas 2000 assumes that the tidal predictions are correct to the datum for the year 2000 and adjusts the record based on the slr function relative to that year).
10. **Tidal range time series from a water level series.** The call to the function is:  
`tidalrange(X, t)` where X is the water level and t is the times of the water level values. Assumes that there are multiple water level values per tide. Also outputs mean water level and tidal range values as a table.
11. **Selection of plots for water level frequency and duration** using the function:  
`waterlevelfreqplots (X, t)` where X is the variable and t is time. Plot options include Water level elevation frequency, Water level spectrum, Elevations above a threshold, Duration of

threshold exceedance, Elevation frequency above threshold. Designed to analyse water levels but could easily be adapted for other variables.



## 12. Selection of frequency analysis plots of timeseries data using the function:

`frequencyanalysis(X, t, 'vardesc')` where  $X$  is the variable,  $t$  is time and *vardesc* is the description of the variable to be used in the plots (optional – defaults to 'Variable'). Plot options include Time series plot of variable, Time series plot of variable above threshold, 'Plot variable frequency, Plot variable frequency above threshold, Spectral analysis plot, Duration of threshold exceedance, Rolling mean duration above a threshold.

## 13. Depth dependent wave steepness using the function:

`wave_steepness(X, Y, Z, t)` where  $X$  is the wave height,  $Y$  the wave period and  $Z$  the water depth, with  $t$  passing the time associated with the timeseries variables. Note that the water depth data should be a times series of the same length and at the same time intervals as the wave data, or specified as a single constant value (i.e.: `wavesteepness(X, Y, 3.5, t)`).

## 14. Wave height-period scatter plots using the function:

`wave_scatter(dst)` where *dst* invokes the option to pass the data as dstables. This requires that the wave height, wave period and water depth time series to be used are assigned to the  $x$ ,  $y$  and  $x$  buttons respectively.

## 15. Ratio of alongshore to cross-shore transport using the function:

`beachtransportratio(x, theta)` where  $X$  is the wave direction and  $\theta$  is the beach or shoreline angle (both in degTN).

## 16. Littoral drift statistics. Plots the annual and monthly volumes of drift along with details of gaps and calms. The call to this function is:

`littoraldriftstats(X, t, 'period')`, where  $X$  is the rate of drift, time is the associated time for that variable and *period* can be 'year', or 'month'.

If no *period* is specified, the default is month. The *period* selection does not alter the plot (which shows both) but if the results are saved as a timeseries, *period* determines the



timeseries interval. In the lower plot, the diamonds denote the start and end of the timeseries.

17. **Ratio of alongshore to cross-shore transport.** The CERC formula for littoral transport is based on the energy flux (P) in the direction of wave advance per unit length of beach. ie:  $F = P \cdot \cos(\alpha)$ , where  $\alpha$  is the angle between wave crest and bed contour. The longshore component of energy flux is  $P \cdot \cos(\alpha) \cdot \sin(\alpha)$ , which leads to the main terms in the CERC formula. It follows that the cross-shore component is  $P \cdot \cos^2(\alpha)$ . The ratio of longshore/cross-shore energy flux (or transport potential) =  $\tan(\alpha)$ . The call to the function is:  
`beachtransportratio(X, theta)` where X is a timeseries of inshore wave directions and 'theta' is the angle of the shoreline to True North.

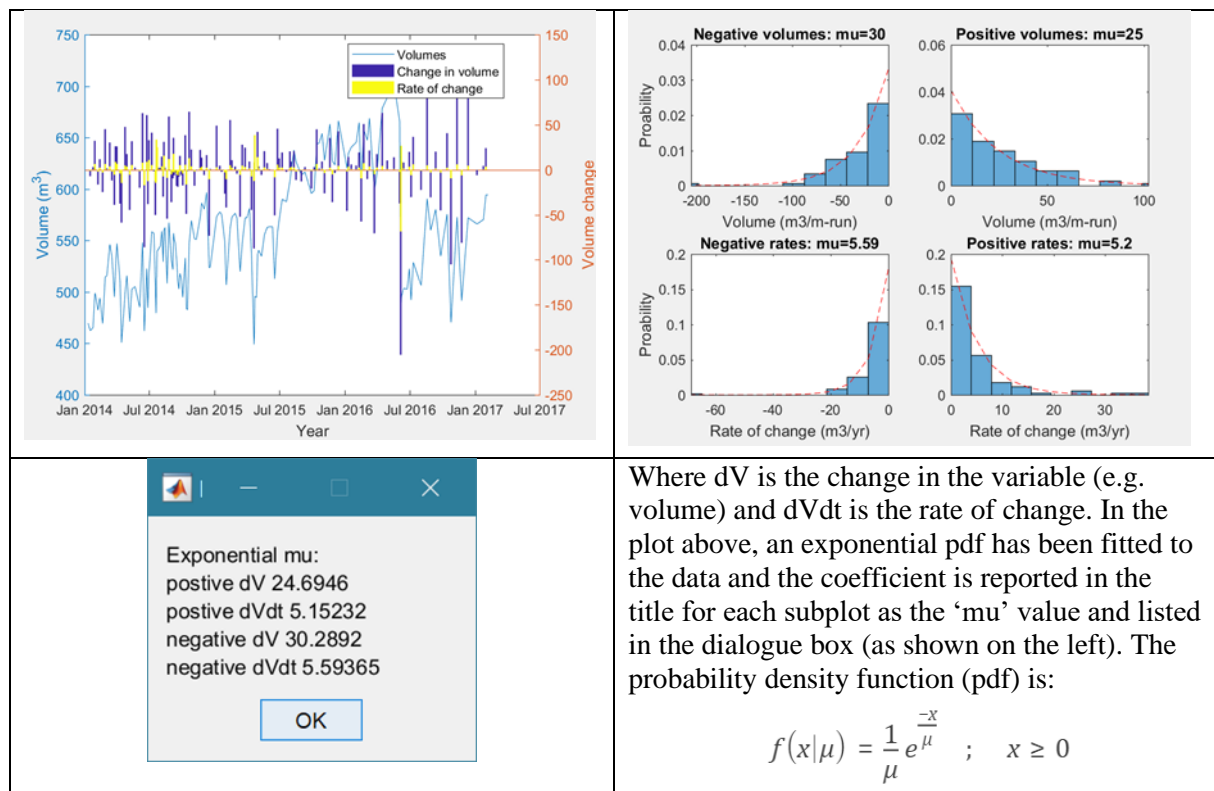
18. **Examining the rate of beach profile change (accretion and erosion).** The function computes the change in the variable over each time interval and the rate of change (assuming a linear rate of change between surveys), subdivides the population into positive and negative change values (typically this represents accretion and erosion when using volumes, or shoreline position, data) and presents the histogram and exponential fit for each data set.

**Warning:** the results are entirely dependent on the adequacy of the source data to represent change (e.g. volumes or shoreline position). If the survey frequency is not regular the results are unlikely to be reliable.

Select a variable X. The variable can be any metric such as beach volume or shoreline position. Use the function:

`posneg_v_stats(x, t, 'VariableName')`, where x is a time series variable, such as beach volume of shoreline position, with associated time, t (defined by variable selection). The *VariableName* is optional but if used should be between single quote marks. The *VariableName* is used to label the plot axes.

The output comprises the following plots and dialogue box:



N.B. `posneg_dv_stats.m` requires the statistics and machine learning toolbox.

#### 4.7.4 Adding variables to peak and cluster time series

Peak and cluster time series are a subset of the source data set, saved as an independent record. For some workflows, there may be a need to add other variables with the same date-time as the subseries (e.g. to produce a polar plot of the cluster events, direction must be added to the timeseries, or to run a model (such as overtopping) period and direction may need to be added). This can be done using the Derive Output GUI, by selecting the subseries variable (e.g. wave energy clusters) as X (it must be X) and the source data set variable (e.g. wave period or direction) as the Y variable.

Then for the equation string, use a function of the form  $Y \cdot (X/X)$

This simply adds the variable defined by Y at the intervals defined by X (assuming X is the subseries). The new variable is then named as required. However, if the new variable is *direction* and it is to be used in the plotting routines (e.g. wave roses) the Variable Name MUST begin with 'Dir' but this can be followed by any additional characters.



## 5 User functions

To allow the user to add other data sets, models or statistical routines, without disrupting the underlying model structure, selected functions have been made external to the class definitions. Copies of the sample files can be found in the `..\muitoolbox\muitemplates` folder. For User Data and User Model these can be copied to a working folder and renamed to suit the application. For User Statistics and User Plots the files in the templates folder can be used to replace the version in the `..\muitoolbox\psfunctions` folder. The `user_plots` and `user_stats` functions are called from the respective UIs, hence the names should not be changed (unless the code in calling class is also edited).

### 5.1 User Statistics

On the General and Timeseries tabs of the Statistics UI there is a Statistic list dialogue box. The User option in this list calls the `user_stats.m` function, which can be found in the `..\muitoolbox\psfunctions` folder. This allows the user to define their own workflow, accessing data and functions already provided by the WaveRayModel App. The sample code illustrates the workflow for timeseries data to produce a clusters plot. If called from the General statistics tab the code simply returns a warning message. The code could be added to provide some alternative function when called from the General stats.

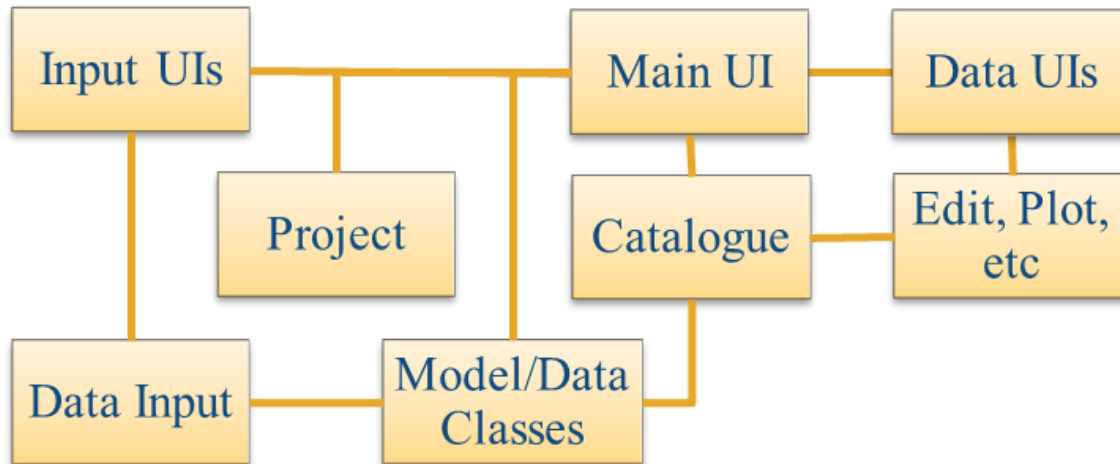
### 5.2 User Plots

On the Plots UI there is a Plot type list dialogue box. Select User and assign the required variables to the buttons for the selected tab. This passes the variables selected to the `user_plot.m` function, which can be found in the `..\muitoolbox\psfunctions` folder. This allows the user to set up their own plotting function. The demonstration function provided plots a line or surface plot depending on whether a variable has been assigned to the z-dimension.

## 6 Program Structure

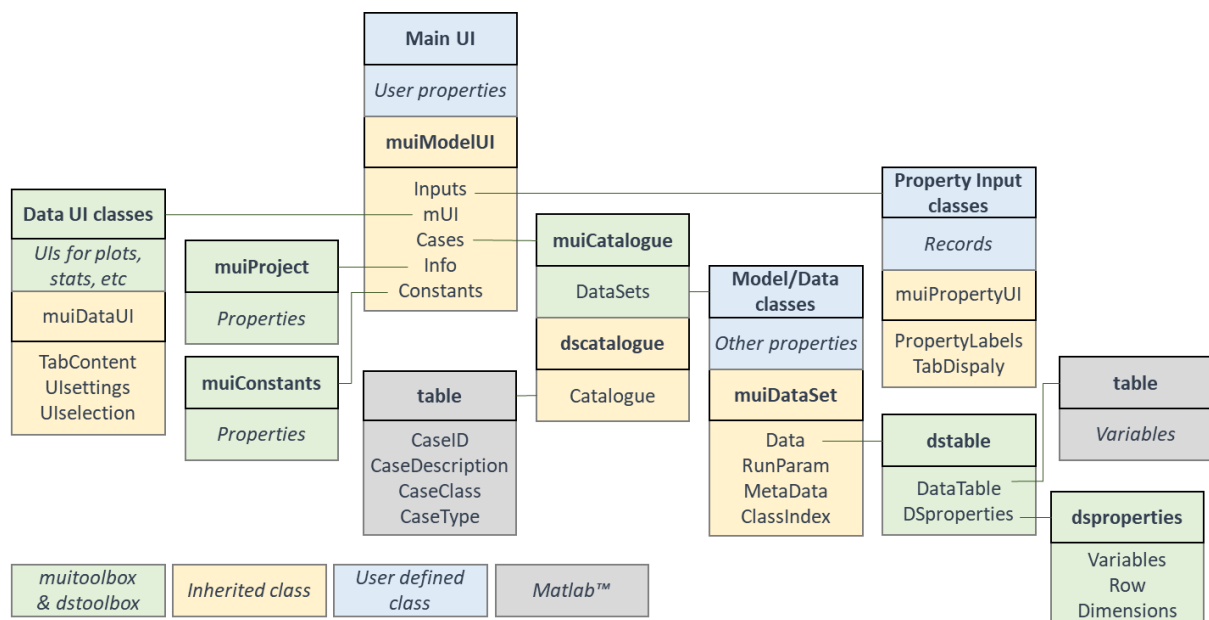
The overall structure of the code is illustrated schematically in Figure 1. This is implemented through several classes that handle the graphical user interface and program workflows (Main UI) and several classes that handle the data manipulation and plotting (Input UIs and Data UIs).

Figure 1 – High level schematic of program structure



The interfaces and default functionality are implemented in the WaveRayModel App using the following mui toolbox classes depicted in Figure 2, which shows a more detailed schematic of the program structure. See the mui toolbox and dstoolbox documentation for more details.

Figure 2 – schematic of program structure showing how the main classes from mui toolbox and dstoolbox are used



In addition, the WaveRayModel App uses the following classes and functions:

**In .../WaveRayModel/**

**WaveRayModel** – main UI

**Ray** – creates individual ray objects. Called by RayTracks and uses arc\_ray function.

**RayTracks** – constructing array of wave ray tracks as a function of wave direction, wave period and water level, working in forward or backward ray tracking mode.

**SpectralTransfer** – builds the offshore and inshore Transfer Tables from a backward ray tracking data set (class RayTracks) for use in WRM\_WaveModel. Also has a method to create plots of the transfer coefficients for a unit wave height.

**WRM\_Bathy** – generate idealised bathymetries using a linear slope or Dean's profile on a linear or crenulate shoreline. Also includes an option for a mound on a linear slope.

**WRM\_BT\_Params** – parameters define initial positions and direction of backward tracking rays.

**WRM\_FT\_Params** – parameters define initial positions and direction of forward tracking rays.

**WRM\_Mesh** - generate triangular mesh of nearshore area from a Cartesian grid.

**WRM\_RunParams** – run parameters for the WaveRayModel.

**WRM\_WaveModel** – Class for wave refraction using backward ray transfer function. Constructs inshore time series from an offshore timeseries. Also includes methods to plot the offshore and inshore spectra.

### In .../WaveRayModel/wrm\_functions

*arc\_ray* – compute the exit position and direction of a ray entering a triangular element at the position and direction defined by the incoming ray.

*celerity\_grid* – calculate the celerity and group celerity over a bathymetry grid for a range of wave periods and a range of water levels.

*compass2trig* – convert compass directions to trigonometric angles, or vice versa.

*curvspace* - evenly spaced points along an existing curve in 2D or 3D., Author: Yo Fukushima, 2005, <https://www.mathworks.com/matlabcentral/fileexchange/7233-curvspace>

*directional\_spreading* – sample a directional spreading function at selected direction intervals.

*get\_element* – define triangular polyshape based on quadrant being entered by ray.

*get\_inshore\_spectrum* - construct the offshore and inshore spectra for given wave conditions or wave buoy spectral data.

*get\_inshore\_wave* - integrate the 2-D spectra to obtain wave parameters and transfer coefficients.

*get\_intersection* – find intersection of a triangle element and a line segment that can be a straight line or an arc segment.

*get\_quadrant* – find the quadrant that the start point lies in or on. Once first intersection has been found subsequent quadrants are defined in next\_element, which calls get\_quadrant if ray direction is aligned to axis.

*interpolate\_cbrewer* – interpolate a colorbrewer map to ncolors levels, Charles Robert, 2011, part of cbrewer from Matlab™ Exchange Forum.

*is\_axis\_point* – find whether point lies on an axis and is travelling in the direction of that axis.

*isangletol* – boolean check of whether an angle lies between upper and lower bounds defined as specific angles, or a tolerance.

*isclosetol* – boolean check of whether an x,y point lies close to another x,y point

*mesh\_arc\_ray* – compute the exit position and direction of a ray entering a triangular mesh element at the position and direction defined by the incoming.

*mesh2d* – generate triangular mesh, Author: Darren Engwirda, 2017, <https://github.com/dengwirda/mesh2d>. NB: the mesh2d folder in the App is empty and this set of

functions needs to be downloaded from the source and installed in the designated sub-folder of the WaveRayModel App.

*PoolWaitbar\_* - initialises and updates a waitbar when running a loop using *parfor*, Matlab™ Forum: Author: Edric Ellis, 2019.

*trigradient* – compute the gradients in the x and y directions using triangular mesh as input, Author: Mick Warehime, 2013, <https://www.mathworks.com/matlabcentral/fileexchange/36837-trigradient-m> .

*wave\_spectrum* – calculate the spectral energy at a number of frequencies using a selection of spectrum definitions (Bretschneider open ocean, Pierson-Moskowitz fully developed, JONSWAP fetch limited, and TMA shallow water).

*wrm\_animation* - animation of model spectra timeseries.

*wrm\_runmovie\_* - callback function for animation figure buttons and slider modified from *muiPlots* to handle two subplots.

#### *Functions:*

##### **In .../muiAppCoastalClasses/**

**ctWaterLevelData** – import water level data

**ctWaveData** – import wave data

**ctWindData** – import wind data

##### **In .../muiAppCoastalClasses/FormatFiles/**

Folder contains a range of format files for loading a number of different data formats and some generic QC functions.

*SFcodes* – beach profile survey feature codes

##### **In .../muiAppCoastalFcns/**

*celerity* - calculate the wave celerity using Hunt's equation (Hunt, 1979).

*crenulate\_bay* - generate the shoreline for an equilibrium crenulate bay using the method of Hsu and Evans, 1989 (Hsu and Evans, 1989).

*deanbeachprofile* - find the bed slope across the surf zone the profile is based on a user defined slope between Hw and SWL (0mOD) and a Dean profile below this level.

Other functions from the folder may be used in the Derive Outputs utility (see Section 4.7 and CoastalTools manual for a full listing).

##### **In .../muiAppGridFcns/**

**GD\_GridProps** - class inherits *muiPropertyUI* abstract class, providing an interface to define the extent and intervals of a cartesian grid.

**GDinterface** - an abstract class to support classes that need additional functionality to handle grids. The class inherits *muiDataSet* abstract class and together they provide an extensive set of methods to handle datasets of various types (eg from models or imported files).

**GD\_ImportData** - class inherits *GDinterface* abstract class (see above) to load xyz data from a file.

Functions used to manipulate cartesian grids can also be found in the *muiAppGridFcns* folder and include the following:

*gd\_ax\_dir* - check direction of grid axes and reverse if descending, OR find grid orientation using ishead and direction of x-axis, OR check a grid axis direction by prompting user.

*gd\_basin\_hypsometry* - compute area and volume hypsometry from gridded elevation data.

*gd\_basin\_indices* - get the indices of the grid x-axis that fall within the basin or channel, when the mouth is offset from the grid origin. (NB: assumes basin/channel is aligned iwth the x-axis). Also returns the index of mouth position on the x-axis.

*gd\_basin\_properties* - use the basin hypsometry from *gd\_basin\_hypsometry* to compute several along-channel/x-axis morphological properties.

*gd\_colormap* - check if Mapping toolbox is installed to use land/sea colormap, or call *cmap\_selection*. the *cmap\_selection* function in the *mutoolbox* if not available.

*gd\_digitisepoints* - creates figure to interactively digitise points on a grid and add elevations if required.

*gd\_dimensions* - get the grid dimsnions for a grid struct (as used in GDinterface).

*gd\_grid\_line* - create a grid from scattered data points input as xyz tuples.

*gd\_gross\_properties* - compute the gross properties of a gridded bathymetry.

*gd\_plan\_form* - compute planform variation along the x-axis at specified planar levels.

*gd\_plotgrid* - create pcolor plot of gridded surface.

*gd\_property\_plots* - plots displayed on Proprety tab or stand-alone figure in Apps that use GDinterface, such as ChannelForm and ModelSkill.

*gd\_section\_properties* - compute the width, cross-sectional area and prism along channel.

*getconvergelength* - least squares fit using fminsearch to find the convergence length of a channel from a distance-width (xy) data set.

*gd\_selectpoints* - accept figure to interactively select one or more points on a grid.

*gd\_setpoint* - interactively select a point on a plot and return the point coordinates. Includes an option to enter an additional value at the selected point (e.g. for elevation).

*gd\_startendpoints* - accept figure to interactively select start and end points on a grid.

*gd\_subdomain* - figure to interactively select a subdomain of a grid.

*gd\_property\_plots* - plots displayed on Proprety tab in ChannelForm model and figure in ModelSkill.

*gd\_xy2sn* - map grid from cartesian to curvilinear coordinates with option to return the elevations on the source cartesian grid, or as a curvilinear grid.

*gd\_sn2xy* - map grid from curvilinear to cartesian coordinates.

*getconvergelength* - least squares fit using fminsearch to find the convergence length of a channel from a distance-width xy data set.

*getsubgrid* - extract a subdomain from a grid and return the extracted grid and the source grid indices of the bounding rectangle.

*a\_star* - implements the A\* search algorithm to find the shortest path given constraints (inaccessible cells) and a cost function (e.g. water depths). Author: Alex Ranaldi, 2022, [https://github.com/alexranaldi/A\\_STAR](https://github.com/alexranaldi/A_STAR).

*InterX* - intersection of two curves. MATLAB Central File Exchange, Author: NS, 2010, <https://www.mathworks.com/matlabcentral/fileexchange/22441-curve-intersections>.



`xy2sn` and `sn2xy` - Cartesian to Curvilinear coordinate forward and backward transformation.  
MATLAB Central File Exchange, Author: Bart Vermeulen, 2022,  
<https://www.mathworks.com/matlabcentral/fileexchange/55039-cartesian-to-curvilinear-coordinate-forward-and-backward-transformation>.<sup>3</sup>

Further details can be found by using the `>>help <function name>` command in the Command Window.

---

<sup>3</sup> The functions by Dugge are equivalent with the order of centreline and co-ordinates reversed in the calls from `gd_xy2sn` and `gd_sn2xy`. See Juernjakob Dugge, 2015, `jdugge/xy2sn`, <https://github.com/jdugge/xy2sn>, which requires `arclength` - John D'Errico, 2012, <https://www.mathworks.com/matlabcentral/fileexchange/34871-arclength>. `distance2curve` - John D'Errico, 2013, <http://www.mathworks.de/matlabcentral/fileexchange/34869-distance2curve>. `interparc` - John D'Errico, 2012, <https://www.mathworks.com/matlabcentral/fileexchange/34874-interparc>.

## 7 Bibliography

- Abernethy, C.L., Gilbert, G., 1975. Refraction of wave spectra. INT 117, Wallingford, UK.
- Abramov, V., Khan, M.K., 2017. A Practical Guide to Market Risk Model Validations (Part II - VaR Estimation). 70. [<http://dx.doi.org/10.2139/ssrn.3080557>]
- Antoniades, I.P., Brandi, G., Magafas, L., Di Matteo, T., 2021. The use of scaling properties to detect relevant changes in financial time series: A new visual warning tool. *Physica A: Statistical Mechanics and its Applications*, 565. [10.1016/j.physa.2020.125561]
- Benoit, M., Marcos, F., Becq, F., 1997. Development of a Third Generation Shallow-Water Wave Model with Unstructured Spatial Meshing, *Coastal Engineering* 1996, pp. 465-478.
- Booij, N., Ris, R.C., Holthuijsen, L.H., 1999. A third-generation wave model for coastal regions: 1. Model description and validation. *Journal of Geophysical Research: Oceans*, 104(C4), 7649-7666. [10.1029/98jc02622]
- Bosboom, J., Reniers, A.J.H.M., 2014. Scale-selective validation of morphodynamic models, 34th International Conference on Coastal Engineering, Seoul, South-Korea, pp. 1911–1920.
- Bosboom, J., Reniers, A.J.H.M., Luijendijk, A.P., 2014. On the perception of morphodynamic model skill. *Coastal Engineering*, 94, 112-125. [<https://doi.org/10.1016/j.coastaleng.2014.08.008>]
- Bouws, E., Günther, H., Rosenthal, W., Vincent, C.L., 1985. Similarity of the wind wave spectra in finite depth water: Part 1 - spectral form. *Journal of Geophysical Research*, 90(C1), 975-986.
- Brandi, G., Di Matteo, T., 2021. On the statistics of scaling exponents and the multiscaling value at risk. *The European Journal of Finance*, 1-22. [10.1080/1351847x.2021.1908391]
- Carter, D.J., 1982. Estimation of wave spectra from wave height and period. 4, Godalming.
- Coles, S., 2001. An Introduction to Statistical Modeling of Extreme Values. Springer Series in Statistics. Springer-Verlag, London.
- Datawell, 2023. Datawell Waves5 Reference Manual, Datawell BV oceanographic instruments, Haarlem, The Netherlands.
- Di Matteo, T., Aste, T., Dacorogna, M.M., 2003. Scaling behaviors in differently developed markets. *Physica A: Statistical Mechanics and its Applications*, 324(1-2), 183-188. [10.1016/s0378-4371(02)01996-9]
- Donelan, M.A., Hamilton, J., Hui, W.H., 1985. Directional Spectra of wind-generated waves. *Phil.Trans.R.Soc.Lond.A.*, 315(1534), 509-562.
- Hsu, J.R.C., Evans, C., 1989. Parabolic bay shapes and applications. *Proceedings - Institution of Civil Engineers.Part 2.Research and theory*, 87, 557-570.
- Hunt, J.N., 1979. Direct solution of wave dispersion equation. *ASCE, Journal of the Waterway, Port, Coastal and Ocean Division*, 105(4), 457-459.
- Ihlen, E.A., 2012. Introduction to multifractal detrended fluctuation analysis in matlab. *Front Physiol*, 3, 141. [10.3389/fphys.2012.00141]



- Kitaigorodskii, S.A., Krasitskii, V.P., Zaslavskii, M.M., 1975. On Phillips theory of equilibrium range in the spectra of wind-generated gravity waves. *Journal of Physical Oceanography*. *Journal of Physical Oceanography*, 5(3), 410-420.
- Kuik, A.J., Van Vledder, G.P., Holthuijsen, L.H., 1988. A Method for the Routine Analysis of Pitch-and-Roll Buoy Wave Data. *Journal of Physical Oceanography*, 18(7), 1020-1034. [10.1175/1520-0485(1988)018<1020:amftra>2.0.co;2]
- Montano, J., Coco, G., Antolinez, J.A.A., Beuzen, T., Bryan, K.R., Cagigal, L., Castelle, B., Davidson, M.A., Goldstein, E.B., Ibaceta, R., Idier, D., Ludka, B.C., Masoud-Ansari, S., Mendez, F.J., Murray, A.B., Plant, N.G., Ratliff, K.M., Robinet, A., Rueda, A., Senechal, N., Simmons, J.A., Splinter, K.D., Stephens, S., Townend, I., Vitousek, S., Vos, K., 2020. Blind testing of shoreline evolution models. *Sci Rep*, 10(1), 2137. [<https://doi.org/10.1038/s41598-020-59018-y>]
- Morales, R., Di Matteo, T., Gramatica, R., Aste, T., 2012. Dynamical generalized Hurst exponent as a tool to monitor unstable periods in financial time series. *Physica A: Statistical Mechanics and its Applications*, 391(11), 3180-3189. [<https://doi.org/10.1016/j.physa.2012.01.004>]
- Pacheco, J.C.R., Román, D.T., Vargas, L.E., 2008. R/S Statistic: Accuracy and Implementations, 18th International Conference on Electronics, Communications and Computers (conielecomp 2008). IEEE. [10.1109/conielecomp.2008.14]
- Pierson Jr., W.J., Moskowitz, L., 1964. A proposed spectral form for fully developed wind seas based on the similarity theory of S. A. Kitaigorodskii. *Journal of Geophysical Research* (1896-1977), 69(24), 5181-5190. [<https://doi.org/10.1029/JZ069i024p05181>]
- Southgate, H., 1981. Ray methods for combined refraction and diffractions problems. IT 214, Hyrdraulic Research Station, Wallingford, UK.
- Sutcliffe, J., Hurst, S., Awadallah, A.G., Brown, E., Hamed, K., 2016. Harold Edwin Hurst: the Nile and Egypt, past and future. *Hydrological Sciences Journal*, 61(9), 1557-1570. [10.1080/02626667.2015.1019508]
- Taylor, K.E., 2001. Summarizing multiple aspects of model performance in a single diagram. *Journal of Geophysical Research - Atmospheres*, 106(D7), 7183-7192. [10.1029/2000JD900719]
- Townend, I.H., Savell, I.A., 1985a. The application of ray methods to wave refraction studies. In: P.P.G. Dyke, Moscardini, A.O., Robson, E.H. (Ed.), *Offshore and Coastal Modelling*. Lecture Notes on Coastal and Estuarine Studies No12. Springer-Verlag, pp. 137–164.
- Townend, I.H., Savell, I.A., 1985b. The use of modelling techniques for a harbour wave study, Halcrow. Unpublished paper.
- USACE, 1984. Shore Protection Manual, I & II. US Army Corps, Vicksburg.

## Appendix A – Input Data File Formats

Existing file formats are detailed in this section. For details of how to add a file format to an existing data class, or add a new data class see Appendix B.

### Bathymetry

Bathymetry is loaded as X, Y, Z tuples for data on a Cartesian grid with a one line header defining the input format followed by rows of data.

1	%f %f %f
2	45000 12456 4.2
3	45120 12455 4.4
4	45122 12458 4.3
5	...

### Waves

#### 1) Channel Coastal Observatory (CCO) format: <http://www.channelcoast.org/>

Date/Time (GMT)	Latitude	Longitude	Flag	Hs (m)	Hm0 (m)	Hmax (m)	Tp (s)	Tz (s)	Dirp (degrees)	Spread (deg)	SST (deg C)
01-Jan-2004 00:00:00	50.73422	-0.49434	3	1.510	2.600	4.8	3.9	183	28	9999.0	
01-Jan-2004 00:30:00	50.73412	-0.49400	3	1.630	2.580	5.0	4.0	176	27	9999.0	
01-Jan-2004 01:00:00	50.73412	-0.49400	3	1.940	3.300	5.3	4.3	176	19	9999.0	
01-Jan-2004 01:30:00	50.73409	-0.49406	3	2.030	3.900	5.6	4.3	177	23	9999.0	
01-Jan-2004 02:00:00	50.73410	-0.49402	3	2.270	3.420	5.6	4.5	183	18	9999.0	
01-Jan-2004 02:30:00	50.73410	-0.49404	3	2.360	3.330	5.9	4.6	176	21	9999.0	
01-Jan-2004 03:00:00	50.73410	-0.49402	3	2.410	3.740	6.3	4.8	184	15	9999.0	
01-Jan-2004 03:30:00	50.73410	-0.49400	3	2.340	3.780	6.3	4.9	186	18	9999.0	
01-Jan-2004 04:00:00	50.73417	-0.49402	3	2.600	3.830	6.7	4.9	191	15	9999.0	
01-Jan-2004 04:30:00	50.73418	-0.49406	3	2.560	3.740	6.3	5.0	194	17	9999.0	

#### 2) Simple wave record

Format is defined using Matlab script in the header line. The format shown here is date, time (hours and minutes only), Hs, Tp, Dir all as real numbers. This allows different date formats to be easily handled. The utility add\_file\_header.m (located in MUIfunctions folder) can be used to add a file header to multiple files.

1	%{dd/MM/yyyy}D %{HH:mm}D %f %f %f
2	01/01/1980 00:00 0.274 9.35 217
3	01/01/1980 03:00 0.256 9.26 211.5
4	01/01/1980 06:00 0.252 9.26 211
5	01/01/1980 09:00 0.25 9.17 210.6
6	01/01/1980 12:00 0.248 9.17 209.9
7	01/01/1980 15:00 0.248 9.09 208.9
8	01/01/1980 18:00 0.246 14.08 207.7
9	01/01/1980 21:00 0.278 13.89 203.3
10	02/01/1980 00:00 0.378 13.7 179.3
11	02/01/1980 03:00 0.402 13.7 258.9
12	02/01/1980 06:00 0.414 13.51 251.3
13	02/01/1980 09:00 0.418 13.33 224
14	02/01/1980 12:00 0.43 13.16 211.9
15	02/01/1980 15:00 0.468 12.99 217.6
16	02/01/1980 18:00 0.436 12.99 203.4
17	02/01/1980 21:00 0.442 13.16 200.6

#### 3) ShoreCast file format (date and time are in vector format followed by variables)

Year	Month	Day	Hour[NZST]	Min	Sec	Hs[m]	Tp[s]	Tm01[s]	Tm02[s]	Dir[°]
1979	1	1	12	0	0	1.860000	12.093000	11.615000	11.303000	53.447000
1979	1	1	15	0	0	1.860000	12.097000	11.625000	11.338000	53.866000
1979	1	1	18	0	0	1.859000	12.108000	11.636000	11.365000	54.076000
1979	1	1	21	0	0	1.851000	12.091000	11.626000	11.365000	54.219000
1979	1	2	0	0	0	1.823000	12.027000	11.598000	11.345000	54.491000
1979	1	2	3	0	0	1.771000	11.900000	11.505000	11.186000	54.856000
1979	1	2	6	0	0	1.690000	11.758000	11.301000	10.744000	55.202000
1979	1	2	9	0	0	1.593000	11.617000	11.158000	10.719000	55.567000
1979	1	2	12	0	0	1.499000	11.475000	11.001000	10.672000	55.958000
1979	1	2	15	0	0	1.420000	11.316000	10.826000	10.563000	56.397000

## Water Levels

### 1) Date-Level format

Format is defined using Matlab script in header line. Format shown here is record number (not used), date, time, elevation (mOD). This allows different date formats to be easily handled. The utility `add_file_header.m` (located in MUIfunctions folder) can be used to add a file header to multiple files.

```
%*s % {yyyy/MM/dd}D % {HH:mm:ss}D %q
262921) 2000/01/01 00:00:00 1.599
262922) 2000/01/01 00:15:00 1.604
262923) 2000/01/01 00:30:00 1.633
262924) 2000/01/01 00:45:00 1.669
262925) 2000/01/01 01:00:00 1.704
262926) 2000/01/01 01:15:00 1.750
262927) 2000/01/01 01:30:00 1.804
262928) 2000/01/01 01:45:00 1.868
262929) 2000/01/01 02:00:00 1.926
262930) 2000/01/01 02:15:00 1.986
262931) 2000/01/01 02:30:00 2.038
```

### 2) Channel Coastal Observatory (CCO) format: <http://www.channelcoast.org/>

Date/Time (GMT)	Tide_OD	Tide_CD	Residual (m)	Flag
04-Jul-2008 14:50:00	-0.262	2.388	0.328	1
04-Jul-2008 15:00:00	-0.190	2.460	0.307	1
04-Jul-2008 15:10:00	-0.098	2.552	0.309	1
04-Jul-2008 15:20:00	-0.006	2.644	0.311	1
04-Jul-2008 15:30:00	0.095	2.745	0.320	1
04-Jul-2008 15:40:00	0.168	2.818	0.299	1
04-Jul-2008 15:50:00	0.298	2.948	0.330	1
04-Jul-2008 16:00:00	0.429	3.079	0.355	1
04-Jul-2008 16:10:00	0.498	3.148	0.310	1
04-Jul-2008 16:20:00	0.620	3.270	0.311	1
04-Jul-2008 16:30:00	0.711	3.361	0.271	1
04-Jul-2008 16:40:00	0.858	3.508	0.279	1
04-Jul-2008 16:50:00	0.975	3.625	0.250	1
04-Jul-2008 17:00:00	1.124	3.774	0.247	1
04-Jul-2008 17:10:00	1.297	3.947	0.264	1
04-Jul-2008 17:20:00	1.470	4.120	0.279	1
04-Jul-2008 17:30:00	1.631	4.281	0.283	1
04-Jul-2008 17:40:00	1.784	4.434	0.283	1
04-Jul-2008 17:50:00	1.920	4.570	0.273	1

### 3) British Oceanographic Data Centre format: <http://www.bodc.ac.uk/>

#### 3(a) BODC NTSLF format

```
Port: P008
Site: Portsmouth
Latitude: 50.80256
Longitude: -1.11175
Start Date: 01JAN1991-00.00.00
End Date: 31DEC1991-23.00.00
Contributor: National Oceanography Centre, Liverpool
Datum information: The data refer to Admiralty Chart Datum (ACD)
Parameter code: ASLVBG02 = Surface elevation (unspecified datum) of the water body by bubbler tide gauge
```

Cycle	Date	Time	ASLVBG02	Residual				
Number	YYYY	MM	DD	HH	MI	SS	f	f
689)	1991/01/29	16:00:00	0.618	-0.227				
690)	1991/01/29	17:00:00	1.076	-0.155				
691)	1991/01/29	18:00:00	1.634	-0.119				
692)	1991/01/29	19:00:00	1.946	-0.138				
693)	1991/01/29	20:00:00	2.508	-0.133				
694)	1991/01/29	21:00:00	3.436	-0.117				
695)	1991/01/29	22:00:00	4.291	-0.079				
696)	1991/01/29	23:00:00	4.592	-0.046				
697)	1991/01/30	00:00:00	4.455	-0.046				
698)	1991/01/30	01:00:00	4.151	-0.050				
699)	1991/01/30	02:00:00	3.214	-0.049				
700)	1991/01/30	03:00:00	1.734	-0.157				

### 3(b) BODC ODV format

```
//Data documentation at https://www.bodc.ac.uk/data/documents/series/537651/
//If you find any issues with this data please report it to enquiries@bodc.ac.uk
//SDN_parameter_mapping
//<subject>SDN:LOCAL:Chronological Julian Date</subject><object>SDN:P01::CJY1101</object><units>SDN:P06::UTAA</units>
//<subject>SDN:LOCAL:SeaLvl_bubbler</subject><object>SDN:P01::ASLVBG01</object><units>SDN:P06::ULAA</units>
//<subject>SDN:LOCAL:SeaLvl_bubbler2</subject><object>SDN:P01::ASLVBG02</object><units>SDN:P06::ULAA</units>
//<subject>SDN:LOCAL:HTSeaLvl</subject><object>SDN:P01::ASLVBH01</object><units>SDN:P06::ULAA</units>
//
Cruise Station Type WWW-mm-ddThh:mm:ss Longitude [degrees_east] Latitude [degrees_north] LOCAL_CDI_ID EDMO_code Bot. Depth [m]
Chronological Julian Date [days] QV:SEADATANET SeaLvl_bubbler [m] QV:SEADATANET SeaLvl_bubbler2 [m] QV:SEADATANET HTSeaLvl [m] QV:SEADATANET
unspecified 988/1996/172 * 1996-06-20T10:00:17.000 -1.8749 50.7143 537651 43 0 2450255.416858 1 1.862 1 1.862 1 1.872 1
2450255.427179 1 1.877 8 1.869 8 1.880 8
2450255.437500 1 1.901 1 1.892 1 1.901 1
2450255.447917 1 1.920 1 1.913 1 1.920 1
2450255.458333 1 1.920 1 1.913 1 1.920 1
2450255.468750 1 1.904 1 1.895 1 1.901 1
2450255.479167 1 1.898 1 1.890 1 1.896 1
2450255.489583 1 1.884 1 1.882 1 1.888 1
2450255.500000 1 1.860 1 1.859 1 1.864 1
2450255.510417 1 1.831 1 1.828 1 1.834 1
2450255.520833 1 1.799 1 1.796 1 1.802 1
2450255.531250 1 1.765 1 1.763 1 1.765 1
2450255.541667 1 1.731 1 1.726 1 1.736 1
2450255.552083 1 1.713 1 1.707 1 1.718 1
2450255.562500 1 1.705 1 1.698 1 1.705 1
```

### 4) ShoreCast file format (date and time are in vector format followed by variables)

```
Year Month Day Hour[NZST] Min Sec tide[m]
1968 1 1 2 0 0 -0.796000
1968 1 1 3 0 0 -0.462000
1968 1 1 4 0 0 -0.009000
1968 1 1 5 0 0 0.454000
1968 1 1 6 0 0 0.820000
1968 1 1 7 0 0 1.003000
1968 1 1 8 0 0 0.963000
1968 1 1 9 0 0 0.717000
1968 1 1 10 0 0 0.328000
```

## Winds

Three formats are currently implemented.

#### 1) Date-Record format

The format read statement is given as the header of the data. The default definition assumes date, time, wind speed (m/s) direction (degTN). A typical header would then be:

```
%{yyyy-MM-dd}D %{HH:mm:ss}D %f%f
```

This is similar to the wave and water level date-record format files.

Note: the order of the year-month-date format needs to match that used in the file. This allows different date and record formats to be handled. The utility `add_file_header.m` (located in MUIfunctions folder) can be used to add a file header to multiple files.

```
%{dd/MM/yyyy}D %{HH:mm}D %f %f
01/01/1980 00:00 7.1 80
01/01/1980 03:00 7.2 75
01/01/1980 06:00 7.5 80
01/01/1980 09:00 7.1 80
01/01/1980 12:00 7 80
01/01/1980 15:00 5.9 70
01/01/1980 18:00 5.1 80
01/01/1980 21:00 5.3 80
02/01/1980 00:00 6.2 90
02/01/1980 03:00 999 80
02/01/1980 06:00 4.8 80
02/01/1980 09:00 6.44 80
02/01/1980 12:00 6.6 80
02/01/1980 15:00 7 80
```

Note that if the variables in the record change, then a new file format will be needed to define the variables (see Appendix B).

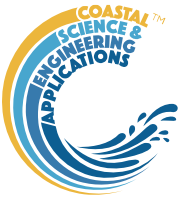
#### 2) CEDA MIDAS format

This data is available from:

<https://catalogue.ceda.ac.uk/data.ceda.ac.uk/badc/ukmo-midas-open/data/uk-mean-wind-obs/dataset-version-201908/>

The data is delivered as a csv file that can be opened in Excel and loaded directly into CoastalTools. The files have a very large header, with a lot of useful meta-data, so an image is not reproduced here.

#### 3) Hong Kong data format



You may need to edit the wind file to replace the header as shown below:

1	Date	hour	Dir10min	Speed10min	Dir1hr	Speed1hr
2	20180913	01	80	7.1	80	7.2
3	20180913	02	80	7.1	80	6.7
4	20180913	03	80	7.1	80	7.5
5	20180913	04	80	7.7	80	7.1
6	20180913	05	80	6.3	80	7.0
7	20180913	06	80	5.4	80	5.9
8	20180913	07	70	6.0	80	5.1
9	20180913	08	80	5.8	80	5.3
10	20180913	09	80	6.3	80	6.2
11	20180913	10	90	2.9	80	5.0
12	20180913	11	80	6.1	80	4.8
13	20180913	12	80	7.4	80	6.4
14	20180913	13	80	6.6	80	6.6
15	20180913	14	80	6.9	80	7.0
16	20180913	15	80	6.5	80	7.0
17	20180913	16	90	5.7	80	6.7
18	20180913	17	90	5.9	90	6.2
19	20180913	18	90	5.7	90	5.7
20	20180913	19	90	5.5	90	5.4

## Wind-wave Hindcast Fetches

For each location on the coast you will need a fetch file. The file Test\_Fetch\_Lengths.txt is an example file. This defines the fetch lengths at the site in 10-degree intervals measured from true north (defined from 0 to 350 degrees even if zero).

1	Dir	Fetch
2	0	0
3	10	0
4	20	0
5	30	1500
6	40	2300
7	50	16000
8	60	17500
9	70	23000
10	80	22000

## Beach Profiles

1) Channel Coastal Observatory (CCO) format: <http://www.channelcoast.org/>

Easting	Northing	Elevation_OD	Chainage	FC	Profile	Reg_ID
488072.856	95988.852	6.364	130.396 G	112	4d01403	
488075.556	95986.645	6.381	133.882 G	112	4d01403	
488078.392	95984.552	6.446	137.406 G	112	4d01403	
488081.390	95982.129	6.501	141.260 G	112	4d01403	
488084.150	95979.926	6.573	144.791 G	112	4d01403	
488087.123	95977.767	6.619	148.463 G	112	4d01403	
488089.877	95975.476	6.691	152.044 G	112	4d01403	
488092.331	95973.597	6.659	155.134 G	112	4d01403	
488094.608	95971.872	4.963	157.991 G	112	4d01403	
488095.918	95970.882	4.810	159.632 G	112	4d01403	
488096.593	95970.270	4.808	160.541 G	112	4d01403	
488097.447	95969.618	5.052	161.615 G	112	4d01403	
488098.132	95969.069	5.117	162.493 G	112	4d01403	

## 2) Chainage profiles listed as one, or more, profile with one, or more, surveys (dates) per file

```

1 Site ProfileID Date Chainage Elevation Flag
2 NARRA PF6 1976-04-27 0 6.29 EMERY
3 NARRA PF6 1976-04-27 10 1.68 EMERY
4 NARRA PF6 1976-04-27 20 0.1 EMERY
5 NARRA PF6 1976-04-27 30 -0.95 EMERY
6 NARRA PF8 1976-04-27 0 3.42 EMERY
7 NARRA PF8 1976-04-27 10 2.65 EMERY
8 NARRA PF8 1976-04-27 20 2.24 EMERY
9 NARRA PF8 1976-04-27 30 1.9 EMERY
10 NARRA PF8 1976-04-27 40 1.42 EMERY
11 NARRA PF8 1976-04-27 50 1.18 EMERY
12 NARRA PF8 1976-04-27 60 0.88 EMERY
13 NARRA PF8 1976-04-27 70 0.36 EMERY
14 NARRA PF8 1976-04-27 80 -0.19 EMERY
15 NARRA PF8 1976-04-27 90 -0.51 EMERY
16 NARRA PF2 1976-04-27 0 9.25 EMERY
17 NARRA PF2 1976-04-27 10 7.92 EMERY
18 NARRA PF2 1976-04-27 20 6.75 EMERY
19 NARRA PF2 1976-04-27 30 3.72 EMERY
20 NARRA PF2 1976-04-27 40 2.54 EMERY
21 NARRA PF2 1976-04-27 50 2.15 EMERY
22 NARRA PF2 1976-04-27 60 0.66 EMERY
23 NARRA PF2 1976-04-27 70 0.18 EMERY
24 NARRA PF2 1976-04-27 80 -0.09 EMERY

```

## 3) Chainage profiles listed as one survey (date) with multiple profiles per file

```

1 Site ProfileID Date Chainage Elevation Flag
2 DUCK P-100 1998-01-12 0.00 6.30 GPS
3 DUCK P-100 1998-01-12 12.00 5.85 GPS
4 DUCK P-100 1998-01-12 24.00 3.62 GPS
5 DUCK P-100 1998-01-12 36.00 2.25 GPS
6 DUCK P-100 1998-01-12 48.00 1.13 GPS
7 DUCK P-100 1998-01-12 60.00 -0.01 GPS
8 DUCK P-100 1998-01-12 72.00 -1.19 GPS
9 DUCK P-100 1998-01-12 84.00 -1.86 GPS
10 DUCK P-100 1998-01-12 96.00 -2.14 GPS
11 DUCK P-100 1998-01-12 108.00 -2.29 GPS
12 DUCK P-100 1998-01-12 120.00 -2.26 GPS
13 DUCK P-100 1998-01-12 132.00 -2.17 GPS
14 DUCK P-100 1998-01-12 144.00 -2.00 GPS
15 DUCK P-100 1998-01-12 156.00 -1.88 GPS
16 DUCK P-100 1998-01-12 168.00 -2.07 GPS

```

## Shoreline Position

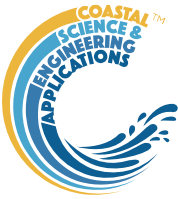
Format is as used for the 2018 ShoreCast exercise (Montano et al., 2020)

Header defines columns (NB date and time are 6 columns unlike most other formats provided for).

```

1 Year Month Day Hour[NZST] Min Sec Average[n
2 1999 1 2 9 0 0 56.399675
3 1999 1 3 7 30 0 57.672548
4 1999 1 4 6 0 0 58.945422
5 1999 1 4 11 0 0 58.861482
6 1999 1 4 18 0 0 57.973684
7 1999 1 5 7 0 0 56.324917
8 1999 1 5 11 0 0 56.583697
9 1999 1 6 7 0 0 61.708058
10 1999 1 6 12 0 0 62.101505
11 1999 1 7 10 30 0 61.209084
12 1999 1 8 9 0 0 60.316662
13 1999 1 8 14 0 0 60.458741
14 1999 1 9 10 0 0 60.972464
15 1999 1 10 2 40 0 60.536215
16 1999 1 10 19 20 0 60.099966
17 1999 1 11 12 0 0 59.678588
18 1999 1 11 16 0 0 59.593717
19 1999 1 12 12 40 0 59.155216
20 1999 1 13 9 20 0 58.716715

```



## BlueKenue

See Blue Kenue manual for details of format:

[http://www.nrc-cnrc.gc.ca/eng/solutions/advisory/blue\\_kenue\\_index.html](http://www.nrc-cnrc.gc.ca/eng/solutions/advisory/blue_kenue_index.html)

### 1) Time series at a point \*.ts1

```
#####
:FileType ts1  ASCII  EnSim 1.0
# Canadian Hydraulics Centre/National Research Council (c) 1998-2012
# DataType          Type 1 Time Series
#
:Application        BlueKenue
:Version            3.3.4
:WrittenBy          Ian
:CreationDate       Fri, Feb 17, 2017 03:04 PM
#
#-----
:Name  Resampled FREE SURFACE (X 618062.01 Y 5633731.79)
#
:AttributeUnits 1 M
#
:LocationX      618062.009194
:LocationY      5633731.789242
#
:StartTime      2010/01/01 12:00:00.000
:DeltaT         0:15:00.000
#
:EndHeader
1.500000
1.500000
1.500000
1.500000
1.500539
```

### 2) Time series at a point \*.ts2

Can be velocity and direction (MAGDIR), as shown here, or two velocity components (UV):

```
#####
:FileType ts2  ASCII  EnSim 1.0
# Canadian Hydraulics Centre/National Research Council (c) 1998-2012
# DataType          Type 2 Time Series
#
:Application        BlueKenue
:Version            3.3.4
:WrittenBy          Ian
:CreationDate       Fri, Feb 17, 2017 04:43 PM
#
#-----
:SourceFile  Solent1.xss
:ExtractPointSet  Sample point
#
:Name  VELOCITY UV (X 618062.01 Y 5633731.79)
:Title VELOCITY UV: Solent1
#
:AttributeUnits 1 M/S
#
:LocationX      618062.009194
:LocationY      5633731.789242
#
:DataDefinition  MAGDIR
#
:StartTime      2010/01/01 12:00:00.000
:DeltaT         0:15:00.000
#
:EndHeader
0.000000 0.000000
0.000000 248.689590
0.000000 104.149414
0.000000 345.505493
0.000574 333.952698
0.053124 332.008178
```



### 3) Time series at a point \*.ts3

```
#####
:FileType ts3  ASCII  EnSim 1.0
# Canadian Hydraulics Centre/National Research Council (c) 1998-2012
# DataType      Type 3 Time Series
#
:Application      BlueKenue
:Version          3.3.4
:WrittenBy        Ian
:CreationDate      Fri, Feb 17, 2017 03:17 PM
#
#-----
:Name  Value(1) (X 621304.24 Y 5630374.26)
#
:AttributeUnits 1 M
#
:LocationX      0.000000
:LocationY      0.000000
#
#-----
:EndHeader
2010/01/01 12:00:00.000 1.500000
2010/01/01 12:15:00.000 1.500000
2010/01/01 12:30:00.000 1.500000
2010/01/01 12:45:00.000 1.500013
2010/01/01 13:00:00.000 1.509651
2010/01/01 13:15:00.000 1.686642
2010/01/01 13:30:00.000 1.480736
2010/01/01 13:45:00.000 0.595087
2010/01/01 14:00:00.000 -0.250650
2010/01/01 14:15:00.000 -0.681176
```

### 4) Time series at a point \*.ts4

Can be velocity and direction (MAGDIR), or two velocity components (UV) as shown here:

```
#####
:FileType ts4  ASCII  EnSim 1.0
# Canadian Hydraulics Centre/National Research Council (c) 1998-2012
# DataType      Type 4 Time Series
#
:Application      BlueKenue
:Version          3.3.4
:WrittenBy        Ian
:CreationDate      Sun, Feb 19, 2017 11:03 AM
#
#-----
:Name  VELOCITY UV (X 618291.73 Y 5633143.44)
:Title VELOCITY UV: Solent1
#
:AttributeUnits 1 M/S
#
:LocationX      618291.732987
:LocationY      5633143.444948
#
:DataDefinition  UV
#
:EndHeader
2010/01/01 12:00:00.000 0.000000 0.000000
2010/01/01 12:15:00.000 -0.000000 -0.000000
2010/01/01 12:30:00.000 -0.000000 0.000000
2010/01/01 12:45:00.000 -0.000000 0.000000
2010/01/01 13:00:00.000 -0.000466 0.000647
2010/01/01 13:15:00.000 -0.034492 0.048785
2010/01/01 13:30:00.000 -0.113091 0.158124
2010/01/01 13:45:00.000 0.256879 -0.387983
2010/01/01 14:00:00.000 0.847288 -1.242919
```

## Appendix B – Data set properties (DSproperties)

Data are stored in a *dstable*, which extends a Matlab *table* to hold more comprehensive metadata for multi-dimensional data sets. This makes use of a *dsproperties* class object to hold the metadata. The *dstable* and *dsproperties* classes are part of the *dstoolbox*. When loading data or saving model results the DSproperties can be defined, loaded and saved when creating a new Case within the application. These data are used in the application to provide descriptions of variables and dimensions in UIs, define units and formats and generic labels which are used for plots and analysis outputs. Further details of the *dstable* and *dsproperties* classes can be found in the Matlab Supplemental Software Documentation for the *dstoolbox*. An example of the code to load DSproperties for a time series and a variable with 2 spatial dimensions are shown below.

DSproperties for a set of timeseries variables.

```
dsp = struct('Variables', [], 'Row', [], 'Dimensions', []);

dsp.Variables = struct(...
    'Name', {'AvSpeed', 'MaxSpeed', 'Dir'}, ...
    'Description', {'Mean wind speed', 'Maximum wind speed', 'Mean wind direction'}, ...
    'Unit', {'m/s', 'm/s', 'deg'}, ...
    'Label', {'Wind speed (m/s)', 'Wind speed (m/s)', 'Wind direction (deg)'}, ...
    'QCflag', repmat({'raw'}, 1, 3));

dsp.Row = struct(...
    'Name', {'Time'}, ...
    'Description', {'Time'}, ...
    'Unit', {'h'}, ...
    'Label', {'Time'}, ...
    'Format', {'dd-MM-yyyy HH:mm:ss'});

dsp.Dimensions = struct(...
    'Name', {''}, ...
    'Description', {''}, ...
    'Unit', {''}, ...
    'Label', {''}, ...
    'Format', {''});
```

DSproperties for a 2D variable with 2 spatial dimensions.

```
dsp = struct('Variables', [], 'Row', [], 'Dimensions', []);
|
dsp.Variables = struct(...    %cell arrays can be column or row vectors
    'Name', {'u'}, ...
    'Description', {'Transport property'}, ...
    'Unit', {'m/s'}, ...
    'Label', {'Transport property'}, ...
    'QCflag', {'model'});

dsp.Row = struct(...
    'Name', {'Time'}, ...
    'Description', {'Time'}, ...
    'Unit', {'s'}, ...
    'Label', {'Time (s)'}, ...
    'Format', {'s'});

dsp.Dimensions = struct(...
    'Name', {'X', 'Y', 'Z'}, ...
    'Description', {'X co-ordinate', 'Y co-ordinate', 'Z co-ordinate'}, ...
    'Unit', {'m', 'm', 'm'}, ...
    'Label', {'X co-ordinate (m)', 'Y co-ordinate (m)', 'Z co-ordinate (m)'}, ...
    'Format', {'-', '-', '-'});
```


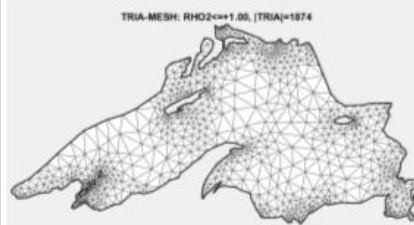
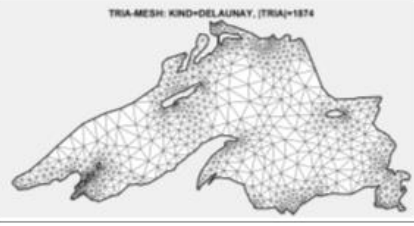
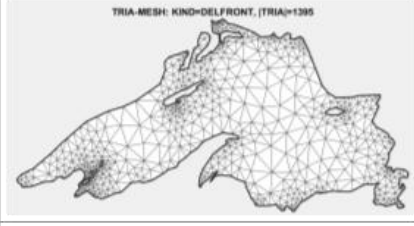

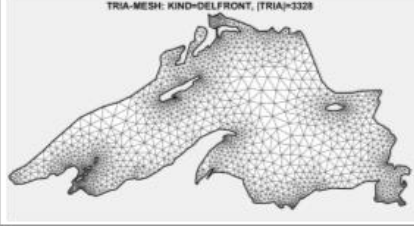
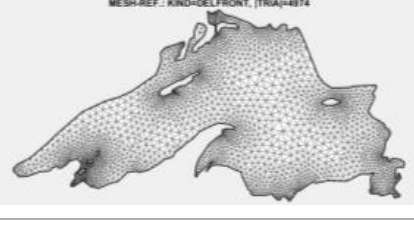
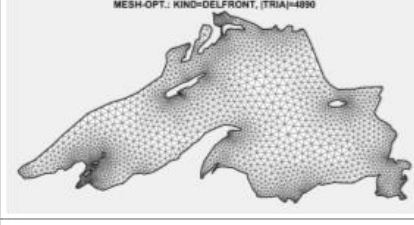
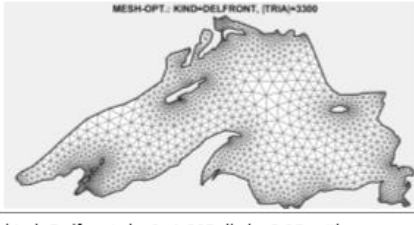
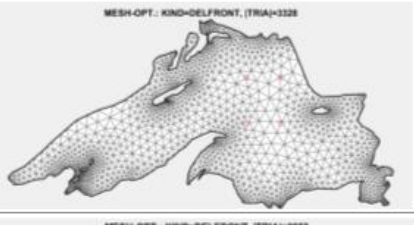
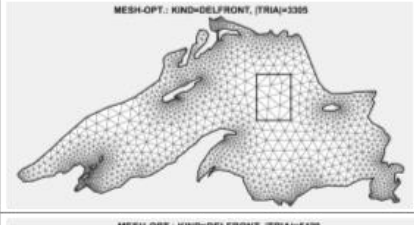
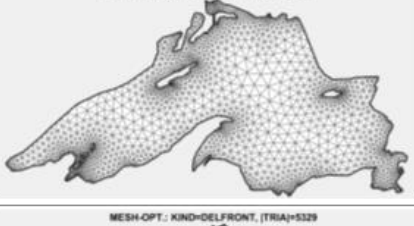
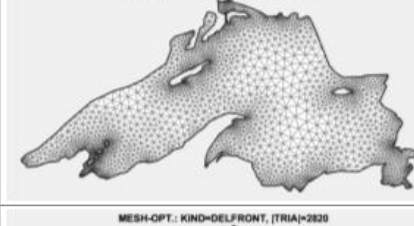
## Appendix C - Date and Time Locale

The system was developed to provide rapid access to data from the Channel Coastal Observatory. It was therefore developed to use British date and time formats. On some computers, with a difference system language, the date and time formats will be different and this can result in some problems with the “Locale” settings. One way to get around this issue is to change the date and time settings in the “Time & Language” section of Settings, or in the “Clock and Region” section in the Control Panel. Change the “Country or region” to United Kingdom. And then click “Additional date, time & regional settings” under “Related settings”; in the “Clock and Region” dialog box select “Change date, time or number formats”, and set the format to “English (United Kingdom)” and other date/time formats are as shown in the picture below.

### [Language preferences](#)

Date and time formats	
Short date:	dd/MM/yyyy
Long date:	dd MMMM yyyy
Short time:	HH:mm
Long time:	HH:mm:ss
First day of week:	Monday

## Appendix D – Mesh parameters

<p>Demo 1</p> <p>Variation of rho2 which influences mesh density/quality</p> <p>kind=Delaunay in both cases</p>		
<p>Demo 2</p> <p>Variation of kind for Delaunay and Delfront</p> <p>rho2=1 in both cases</p>		
<p>Demo3</p> <p>Defaults for rho2=1.025 and kind=Delfront</p> <p>Additionally calls lfshfn2</p> <p>Mesh size constraint</p> <p>dhd x=0.15 and 0.25</p>		
<p>As demo3 with the addition of mesh optimization using smooth2</p>		
<p>kind=Delfront, rho2=1.025, dhd x=0.25, with optimization.</p>	<p>Four nodes added internally. Note that before smoothing points are captured in raw mesh but a small offset is introduced due to smoothing</p>	<p>Four nodes now with defined edges, which are captured in the raw and smoothed mesh</p>
		
<p>kind=Delfront, rho2=1.025, dhd x=0.25, with optimization.</p> <p>Above with default settings for siz1=1.333 and siz2=1.3000</p> <p>Left siz1=1.0 and siz2=1.300 (boundary refined)</p> <p>Right siz1=1.333 and siz2=1.0 (domain refined)</p>		
<p>Left siz1=1.0 and siz2=1.0</p> <p>Right siz1=1.5 and siz2=1.5</p> <p><u>SIZ1</u>: Each exterior edge is refined until <math>LL/HH &lt; SIZ1</math>, where LL is the edge-length, HH is the edge-centred mesh-size value.</p> <p><u>SIZ2</u>: Each interior tria is refined until <math>RE/HH &lt; SIZ2</math>, where RE is an effective tria length, based on the circumradius, HH is the tria-centred mesh-size value.</p>	