



Preface

ModelSkill is a Matlab™ App to load gridded, meshed or timeseries (vector) data and to plot the data on a Taylor diagram. The standard plotting, data manipulation and statistical tools from the `multoolbox` are also included.

Requirements

The model is written in Matlab™ and provided as Open Source code (issued under a GNU General Public License). The App requires the `dstoolbox` and `multoolbox`. The network analysis utility uses the Matlab™ Statistics and Machine Learning Toolbox and Image Processing Toolbox.

Resources

The ModelSkill App can be downloaded from www.coastalsea.uk. The download package includes the install file, the user manual and the software licence.

Cite as:

Townend, I.H., 2021, ModelSkill manual, CoastalSEA, UK, pp41, www.coastalsea.uk.

Bibliography

Bosboom J and Reniers A J H M, 2014, Scale-selective validation of morphodynamic models, 34th International Conference on Coastal Engineering, pp. 1911–1920, Seoul, South-Korea.

Bosboom J, Reniers A J H M and Luijendijk A P, 2014, On the perception of morphodynamic model skill. *Coastal Engineering*, 94, 112-125, doi: <https://doi.org/10.1016/j.coastaleng.2014.08.008>.

Taylor K E, 2001, Summarizing multiple aspects of model performance in a single diagram. *Journal of Geophysical Research - Atmospheres*, 106 (D7), 7183-7192, doi: 10.1029/2000JD900719.

Revision history

Version	Date	Changes
2.1	Oct 2022	Adapted to use Grid tools based on GDinterface
2.0	May 2021	Ported to work as an App using <code>dstoolbox</code> and <code>multoolbox</code>
1.0	Jan.2020	First release



Contents

1	Introduction	1
2	Getting Started.....	1
2.1	Configuration.....	1
2.1.1	Installing the toolboxes	1
2.1.2	Installing the App	1
2.2	Model Set-up	1
2.3	Taylor Diagram	2
2.4	Inlet Tools	2
2.5	User Tools	3
3	Application Menus	4
3.1	File.....	4
3.2	Tools.....	4
3.3	Project.....	4
3.4	Setup.....	5
3.5	Run	8
3.6	Analysis.....	14
3.6.1	Plotting	14
3.6.2	Statistics.....	15
3.7	Help.....	20
3.8	Tabs	20
3.9	UI Data Selection	20
3.10	Form Model Output Table Content	21
3.10.1	Form Table	21
3.10.2	Hypsometry Table	22
3.10.3	SectionProps Table.....	22
3.10.4	Plan Table.....	23
3.10.5	GrossProps Table	23
3.10.6	WaterLevels Table	24
3.11	Accessing data from the Command Window	24
4	Implementation.....	26
4.1	Taylor diagram	26
4.2	Skill score.....	27
4.3	Brier Skill Score relative to initial morphology	28
4.4	Statistical significance	29
4.5	Derive Output.....	29
4.5.1	Calling an external function	30



4.5.2	Input and output format for external functions.....	31
4.5.3	Pre-defined functions	33
5	Input formats	34
5.1	Grid and Mesh data	34
5.2	Timeseries data.....	34
6	Program Structure.....	35
7	Bibliography.....	38



1 Introduction

ModelSkill enables data to be loaded and then compared on a Taylor diagram. This form of plot was originally proposed for the comparison of model timeseries output (Taylor, 2001) and has subsequently been adapted for the comparison of morphological model outputs (Bosboom and Reniers, 2014; Bosboom *et al.*, 2014). In this implementation the Taylor approach is used but modified in line with the method proposed by Bosboom for the analysis of grid and mesh data. The options include the ability to create and add points to a Taylor diagram, output the results to the Clipboard, and estimate global and local skill scores. Some additional tools are included that are specific to inlet/channel form analysis and channel network analysis.

2 Getting Started

2.1 Configuration

ModelSkill 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: `ModelSkill.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:

```
>> ModelSkill;
```

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.

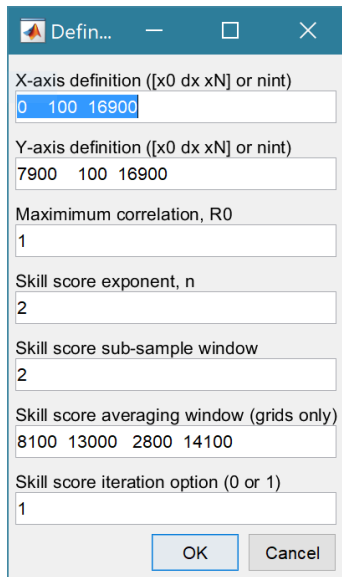
2.2 Model Set-up

File > New to create a new project space.

Setup > Input Data > Gridded Data: sub-menu options to Load, Add, Delete, Quality Control of gridded data sets (see Section 3.4).

Setup > Input Data > Timeseries: timeseries data sets can be loaded containing multiple variables that use a common data and time vector. Different data sets can be loaded with different date and time vectors and/or different variables. For each timeseries, the user is prompted for a data format ‘`m`’ file. This defines the format of the data to be loaded (see Section 5.2).

Setup > Run Parameters: define the dimensions of the grid to use and the parameters to be used to compute the skill score.



The X and Y-axis are defined as ranges and an interval using the format $x_0 \text{ dx } x_N$ (this is used in Matlab to create the range $x_0:dx:x_N$). For re-gridding data, this is the format required to define the new grid. When just plotting mesh data the number of grid intervals can be specified instead (the start and end of the range are obtained from the data set). Note, however, that the values currently set in Run Parameters are the ones used for Re-gridding.

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

Exponent used in computing the skill score (see Section 4.2).

Number of grid cells ($\pm W$) used to define a local window around the i th grid point. If $W=0$ (default) the local skill score is not computed.

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

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

2.3 Taylor Diagram

Run>Taylor Diagram: the workflow is similar for both gridded data and timeseries data. If both types of data have been loaded, the user is prompted to choose between Grid and Timeseries. The next step is to select the Reference dataset (NB: for grids this must be a Cartesian Grid). One is then prompted for the Test dataset (NB: if a grid, this should have the same grid dimensions as the Reference dataset). The Taylor diagram and a plot of local skill score (graph for timeseries and a map for grids) are produced. The user is prompted to Add, or Delete more Test points, or to Exit. Examples of the outputs from the analysis are shown below and more details about the methods used are given in Section 4.1 and 4.2.

A similar tool, using the default Stats UI, is available from the *Analysis> Statistics* menu option. Further details of the Run and Stats options are given in Sections 3.5 and 3.6.2, respectively.

2.4 Inlet Tools

Run>Inlet Tools: calls the function file `getInletTools.m`. This provides a selection of tools to investigate grids which define inlets, estuaries or tidal basins. The functions provided include options to add the gross properties and hypsometry to gridded data sets in the project, as well as tabulate and plot the resulting properties (see Section 3.5 for details).

The *Add Properties* option computes a range of properties for the inlet including the total volumes, surface areas at high and low water, tidal prism and along channel variation of CSA and width.

The *Delete Properties* option removes ALL the property tables from a grid object.

The *Edit Inlet Definition* accesses variables to define channel orientation and distance to the mouth.

The *Plot Case Properties* generates a figure tabulating the gross properties for a selected Case and providing options for a range of property plots.

The *Tabulate Set Properties* tabulate the gross properties for all timesteps in Selected Case.

The *Plot Set Properties* option allows any group of models to be plotted as a set.

The *Plot Hypsometry* option plots the variation of surface area and volume as functions of elevation for a selected model.

The *Plot X-Y Property Sets* option allows a Case+variable to be plotted against another Case+variable. The output shows the evolution of the two variables over time.



The *Plot Thalwegs* option allow the user to define a start and end point for a path and to compute the shortest path between the two points.

The *User Function* option calls the function `ms_userfunction.m`, which can be edited or replaced as required. The example code compares different ways of computing the tidal prism in the basin.

2.5 User Tools

Run>User Tools: calls the function file `getUserTools.m`. This provides a selection of tools to investigate patterns of networks and bed forms.

Network analysis extracts the channel network from a bathymetry and computes the variation of the number of channels and the width of the channels with distance from the inlet mouth. After selecting a case to analyse (bathymetry grid) the user is prompted to define the parameters used to regularize the grid based on diffusion (Passalacqua *et al.*, 2010; Sangireddy *et al.*, 2016).

Rhythmic Forms analyses the rhythmic form spacing and amplitude together with the change in volume over time.

For further details of the input UIs and the types of output generated see Section 3.5.

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 scenarios, 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 a large number of 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 scenario 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). 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

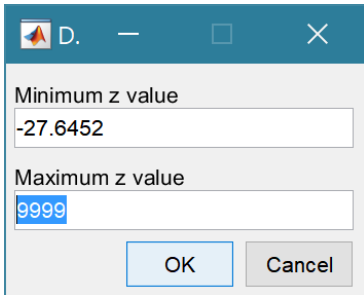
The setup menu provides a series of menus to enable different components of the model to be defined.

Setup> Input Data: options to load gridded and timeseries data. When the data has been loaded, the user is prompted to provide a description of the data set (Case) and this is listed on the *Cases* tab. The source file(s) can be checked by selecting the first column of a Case on the *Cases* tab and then the ‘Source’ button on the Meta-data table, to display a listing of the input files used.

XYZ-gridded data

Setup> Input Data> Gridded Data: load a gridded data set from an ASCII text file containing x, y, z tuples.

Setup>Input data> Gridded Data > Load: prompts for file format to be loaded. The first prompt is for the date-stamp of the file(s) being loaded. For a single file the default of 0 years can be used. For multiple files the default vector is 0 1 2 ...N and should be edited to define the time sequence of the grids (assumed

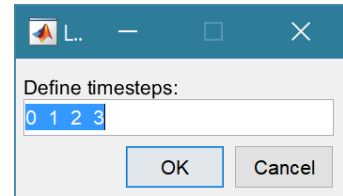


to be in years). The next prompt presents the maximum and minimum Z values in the grid and the option to

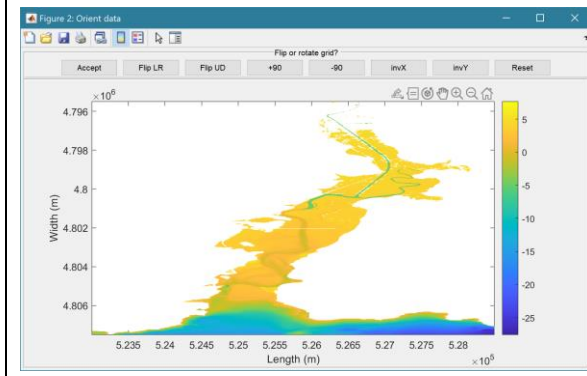
limit the range of data to be loaded (values outside the range are set to NaN). The grid is then plotted on a figure with multiple options to rotate or flip the grid, and invert the direction of the X and Y axes.

This is illustrated below for the case of the Oka estuary in northern Spain. In (b) the orientation of the grid and axes are geographically correct, whereas in (c) the estuary channel has been aligned with the

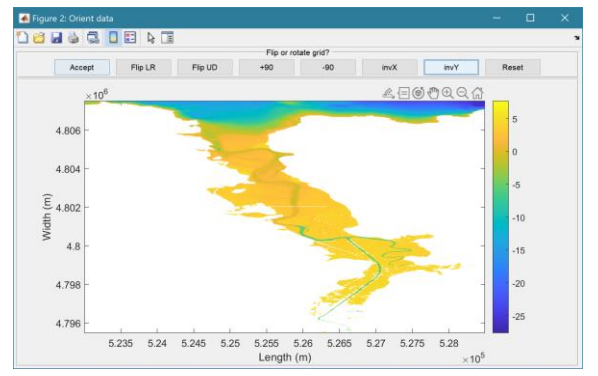
X-axis, whilst keeping the grid coordinates correct relative to the channel (X-axis is descending). This is required if the Inlet Property tools for hypsometry and gross properties are to be used as these assume the channel is aligned to the X-axis.



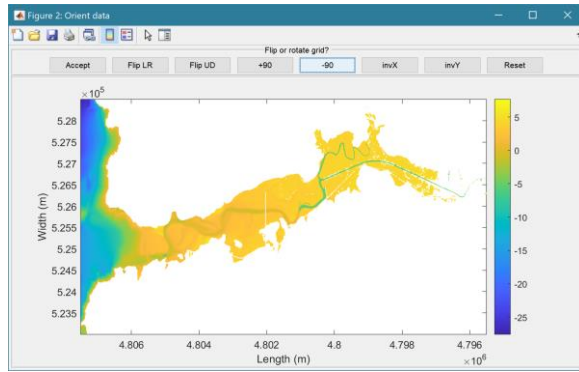
(a) raw data loaded from file



(b) grid flipped UD and Y-axis inverted to give correct geographical orientation



(c) grid flipped UD and rotated -90 to align estuary channel with x-axis (note X-axis is descending but correct geographically)



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

The data is then loaded and the user is prompted for a description (working title) for the data set.

Setup > Input data > Gridded Data > Add: 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). The user assigns timestep, range and adjust orientation as explained for loading grids, above, Only files with the format used to create the data set can be used to Add data to a data record.

Setup > Input data > Gridded Data > Delete: prompts for Case from which some part of the data is to be deleted.

Setup > Input data > Gridded Data > Quality Control: runs a series of checks on the data. This is only available if defined for the specific data format.

Timeseries data

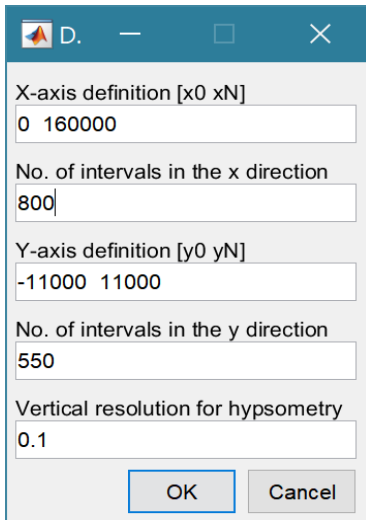
Setup > Input Data > Timeseries: timeseries data sets can be loaded containing multiple variables that use a common data and time vector. Different data sets can be loaded with different date and time vectors and/or different variables. For each timeseries, the user is prompted for a data format ‘m’ file. This defines the format of the data to be loaded (see Section 5.2).

Setup > Input Data > Timeseries > Load: 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.

Setup > Input Data > Timeseries > Add: 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 > Input Data > Timeseries > Delete: prompts for Case from which some part of the data is to be deleted.

Setup > Input Data > Timeseries > Quality Control: runs a series of checks on the data. This is only available if defined for the specific data format.



Setup > Grid Parameters: The grid is defined by x and y ranges and intervals for both axes.

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

The vertical interval to be used when computing the channel hypsometry.

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 grid¹.

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.

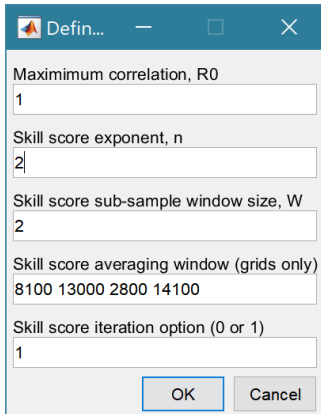
Setup > Grid Tools > Plot Sections: interactively define section on a grid and plot as sections.

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.

¹ Rotation re-orientes 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.

Setup> Run Parameters: define the parameters to be used to compute the skill score.



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

Exponent used in computing the skill score (see Section 4.2).

Number of grid cells (+/-W) used to define a local window around the *i*th grid point. If W=0 (default) the local skill score is not computed.

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

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

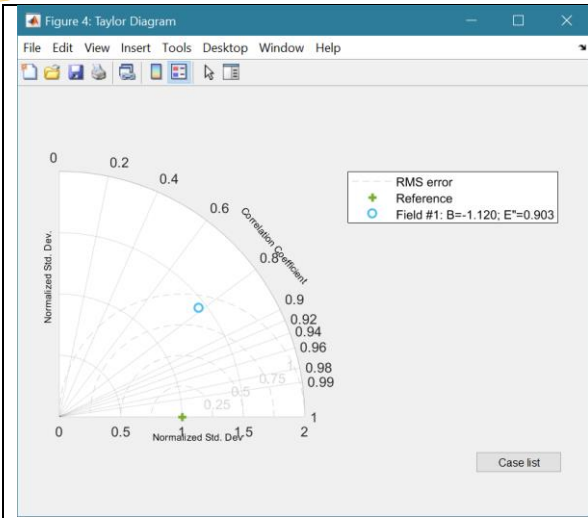
Setup>Input Data>Model Constants: various constants are defined for use in models, such as the acceleration due to gravity, viscosity and density of sea water, and density of sediment. Generally, the default values are appropriate (9.81, 1.36e-6, 1025 , 2650 respectively) but these can be adjusted and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

3.5 Run

Run>Taylor Diagram: the workflow is similar for both gridded data and timeseries data. If both types of data have been loaded, the user is prompted to choose between Grid and Timeseries. The next step is to select the Reference dataset (NB: for grids this must be a Cartesian Grid). One is then prompted for the Test dataset (NB: if a grid, this should have the same grid dimensions as the Reference dataset). The Taylor diagram and a plot of local skill score (graph for timeseries and a map for grids) are produced. The user is prompted to Add, or Delete more Test points, or to Exit. Examples of the outputs from the analysis are shown below and more details about the methods used are given in Section 4.1 and 4.2.

For timeseries data when selecting Reference and Test datasets one may be additionally prompted to select a variable if the timeseries collection contains more than one variable. Additionally, if there are NaNs in either of the variables selected the user is prompted to select which variable to use as the basis of interpolation to create two timeseries of the same length.

When adding points to an existing plot, the user can opt to use the existing Reference dataset or choose a new one. NB: the Reference data sets are all plotted at the same point with $(\sigma, R) = [1, 1]$. This allows composite plots to be generated (e.g. for a bootstrap analysis, using each model output as the Reference case in turn).



Taylor diagram with legend that includes:

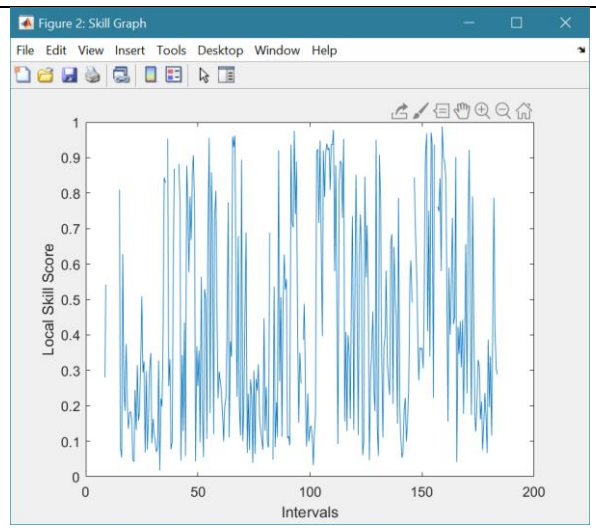
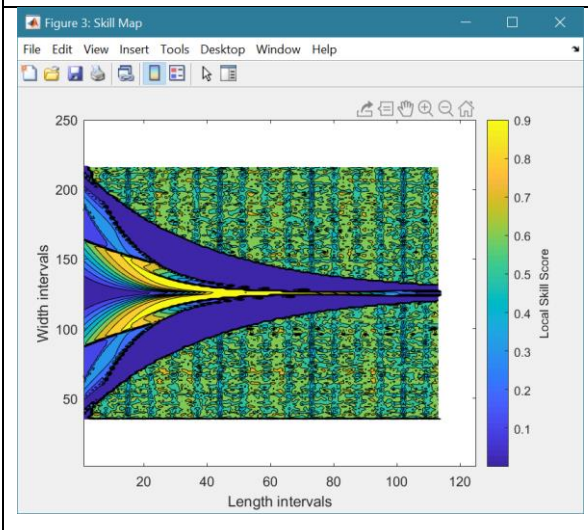
B – bias

E* – normalised RMS difference

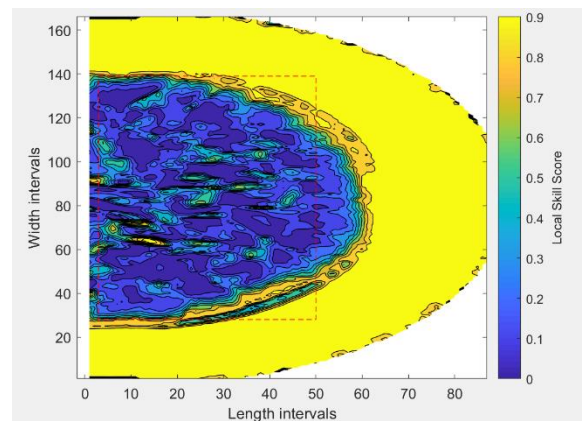
The normalised standard deviation and correlation coefficient are also given in the Case List table, along with the global skill score, Sg, and the average local skill score, Sl.

An additional figure showing the local skill scores as a map for 2D data. or an x-y plot for timeseries data. is also generated.

To suppress the inclusion of local skill scores and associated plots set W=0 in Run Parameters



If $W > 0$ the skill score is computed. This creates an additional plot showing the local skill, computed over the sampling window defined by W . The dashed red line is the skill score averaging window – the average value of the local skill scores is calculated using the values inside this window. The global and local skill scores for each case are listed on the Case List (accessed using the Case List button on the Taylor Diagram figure).



Run>Inlet Tools: calls the function file getInletTools.m. This provides a selection of tools to investigate grids which define inlets, estuaries or tidal basins. The functions provided include options to add the gross properties and hypsometry to gridded data sets in the project, as well as tabulate and plot the resulting properties.

Add Properties: add form properties to a gridded inlet data set. The user is prompted to input levels for HW, MTL and LW, or provide a file² to define along channel water levels. If there are multiple grids loaded for the selected Case, there is the option to use the same water levels for all grids, or specify for each grid. *Note: the grid must have the channel oriented along the x-axis and sections are taken normal to the x-axis. This can produce strange results for the section properties, such as width and cross-sectional area, if the channel meanders.*

Delete Properties: delete ALL property tables associated with a selected gridded data set.

Edit Inlet Definition: provides access to the definition of the position of the head of the inlet, distance from the x-axis origin to the mouth and any definition of the channel centre-line (if used).

Plot Case Properties: plot the properties of a selected case on a figure along with a tabulation of the inlet/channel gross properties.

Tabulate Set Properties: tabulate the gross properties for all timesteps in Selected Case, which can then be copied to the clipboard.

Plot Set Properties: plot selected gross properties for selected Cases as a function of time. The output is a composite timeseries plot of selected Cases+variables. The user is prompted to give the set a title (used in legend) and then prompted to add another data set or exit. For each addition a new set of models and a different variable can be selected providing the freedom to generate a variety of plot outputs.

Plot Hypsometry: plot surface area and volume hypsometry for one or more selected Cases and time intervals (note: the gross properties function must have been run for the selected model for this to work).

Plot X-Y property sets: plot selected gross properties as X and Y to show co-variant change over time. Selected Cases must be the same length and are assumed to have contemporaneous points.

Plot Thalwegs option allow the user to define a start and end point for a path and to compute the shortest path between the two points. The user is prompted for a maximum water level to be accessible to the path and a depth exponent. The cost function in the A* algorithm is defined by the height above the deepest water depth in the grid weighted by the depth exponent (default = 2). The default value follows channels but often short-circuits taking the shallower channel around bends. Increasing the value to 4 or 5 usually corrects this. The first grid selection is used for the background surface but additional paths can be added using different grids.

User Function option calls the function ms_userfunction.m, which can be edited or replaced as required. The example code compares different ways of computing the tidal prism in the basin.

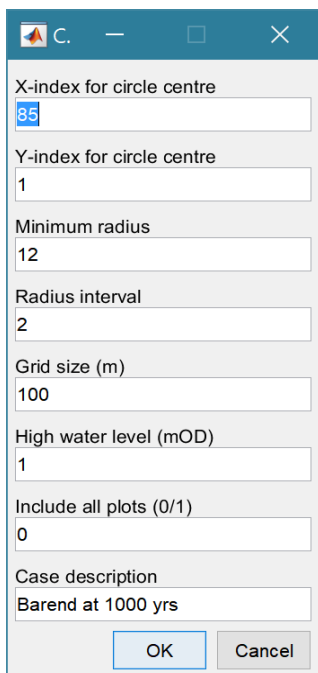
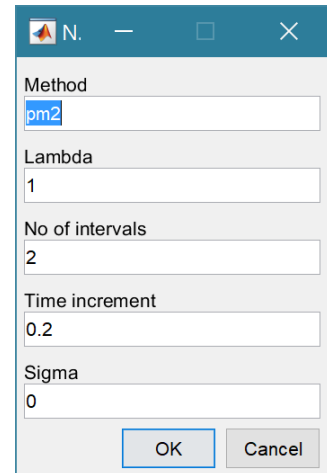
² File format is: single header line followed by 4 columns of numerical values for distance, high water level, mean tide level and low water level. The distance must cover the extent of the grid being used.

Run>User Tools: calls the function file `getUserTools.m`. This provides some sample functions for the analysis of dendritic networks and the rhythmic nature of bed forms. However, this function file can be replaced or overloaded to meet the needs of other applications.

Network Analysis: extracts the channel network from a bathymetry and computes the variation of the number of channels and the width of the channels with distance from the inlet mouth. After selecting a case to analyse (bathymetry grid) the user is prompted to define the parameters used to regularize the grid based on diffusion (Passalacqua *et al.*, 2010). The method is selected from:

- 'lin': Linear diffusion (constant $c=1$).
- 'pm1': perona-malik, $c=\exp(-(|\text{grad}(J)|/\lambda)^2)$
- 'pm2': perona-malik, $c=1/(1+(|\text{grad}(J)|/\lambda)^2)$
- 'tukey': $c=0.5 ((1-(\text{grad}(J)/\lambda)^2)^2) \cdot \text{grad}(J)$
- 'rmp': complex valued - ramp preserving

Lambda is a coefficient used in the Perona-Malik (Perona and Malik, 1990) and tukey formulations, number of intervals, time increment and sigma are used to adjust how the diffusion in the grid is determined.



The second UI defines variables specific to the case being examined. A series of circles are used to extract network properties at increasing distance (radius) from a defined centre. This will depend on the orientation of the grid and has not been extensively tested. There may therefore be a need to rotate the grid before the function will work. A value '0' defines the mid-point in the X or Y direction and allows the user to adjust the point interactively. Other values can be any number of grid cells from the edge, thereby adjusting the area that is searched. Minimum radius defines the smallest radius to be used and the Radius interval is used to increase the radius up to the maximum that will fit within the domain. Both are defined as a number of grid cells.

The Grid size should be based on the grid being analysed.

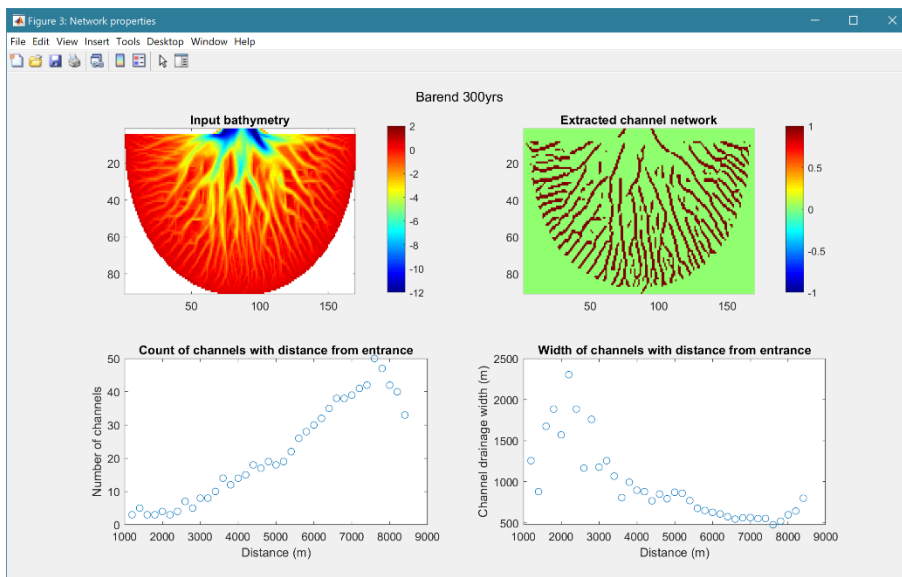
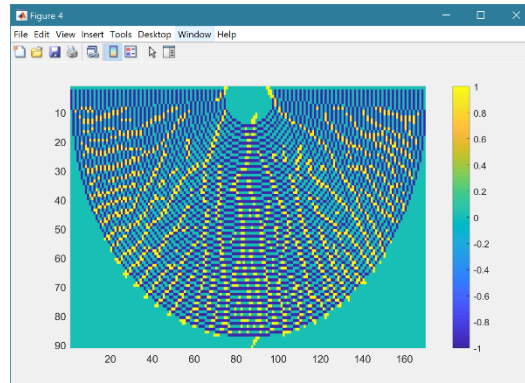
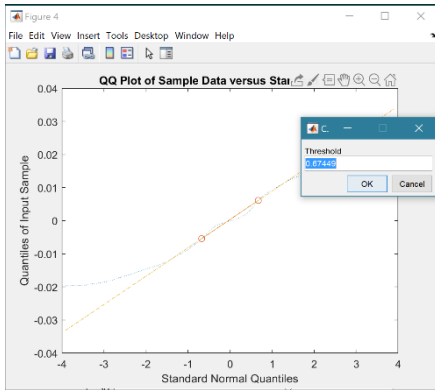
High water provides an upper bound for the analysis.

Logical flag to include or exclude several plots that can be useful for initial checking of parameter selection.

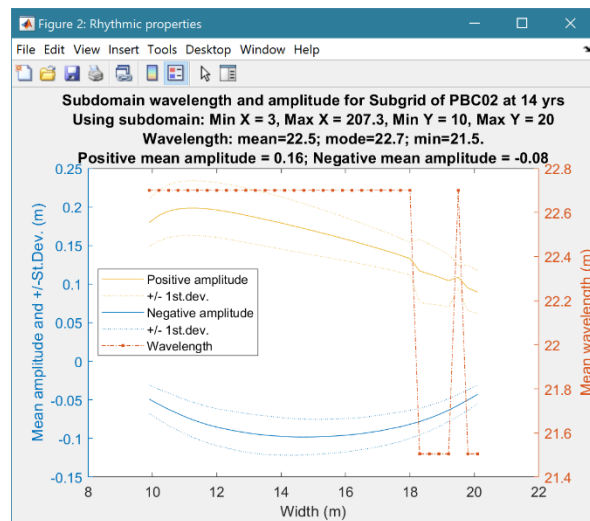
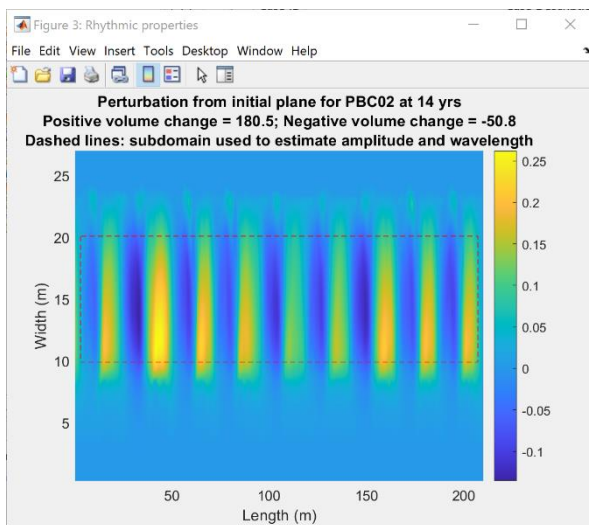
Case description is used to label the plot output.

A plot to help identify the threshold to use as the curvature threshold is then generated and the user is prompted to adjust the default value. The default is based on the 75-percentile value of the Standard Normal Quantile. Larger values result in narrower channels and smaller values in more space filling channels. If the option for All plots is selected, a plot showing the network and the circles used to extract data points is plotted. This is useful for checking that the centre has been correctly defined. If the X and Y-index of the circle centre have not been specified (0 in above UI) the central value is plotted on the extracted network and the user can edit the position interactively to get the desired coverage of the circles used for sampling.

The final plot includes the source bathymetry, the network, a count of channels and drainage distance between channels both plotted against radial distance from the inlet mouth. The former plot includes details of the peak count, and the latter has additional lines for the mean (black dashed), +/- 1 standard deviation (black dots) and the linear slope (red dash-dot) with regression equation and R^2 value.



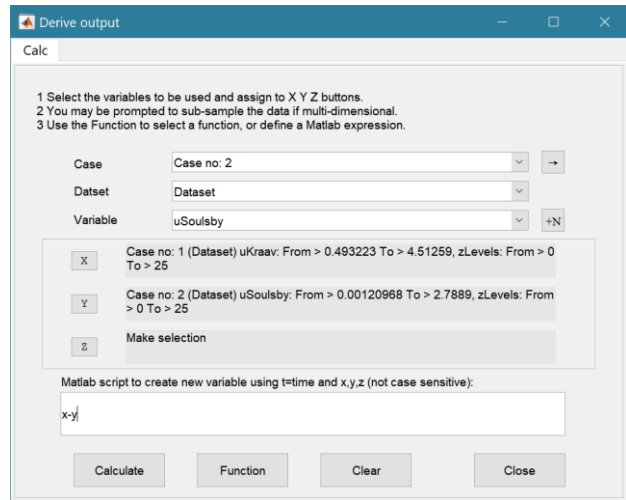
Rhythmic Forms: plot rhythmic forms on a horizontal surface (ie bed perturbation), with estimates of the spacing and volume change relative to a plane surface.



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 variables³. 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™ 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 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 (number 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);
```

³ 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.


```
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.

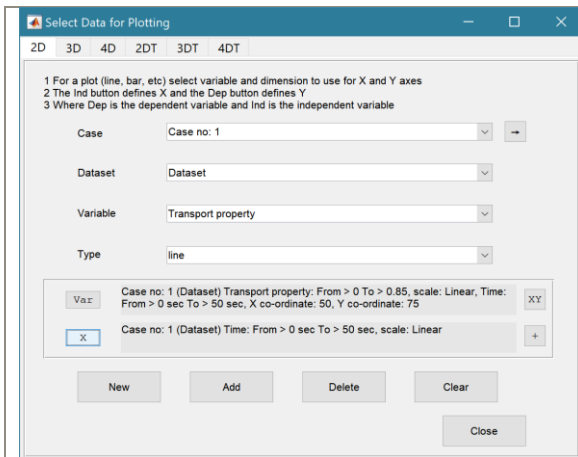
Some further details on using this option and the ‘**Function**’ library available are provided in Section 4.5.

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)

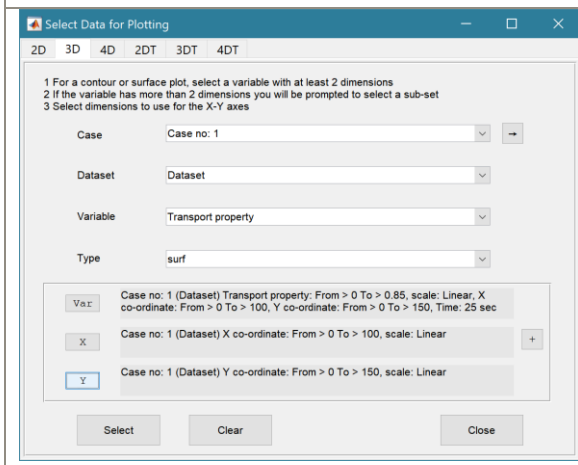
Control Buttons:

→ : 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.

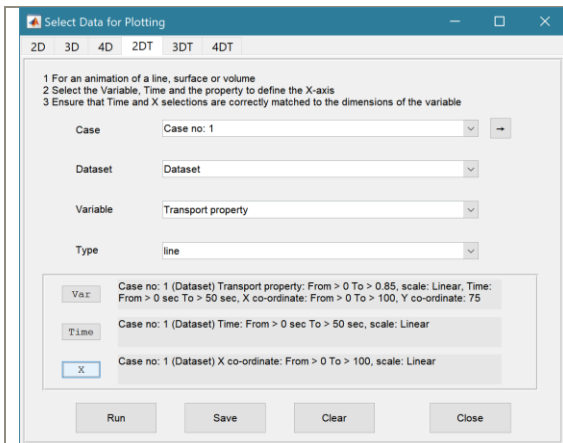
For all plot types, when the data has more dimensions than the plot or animation the user is prompted to sub-select from the data (by selecting sampling values for the dimensions that are not being used).

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.

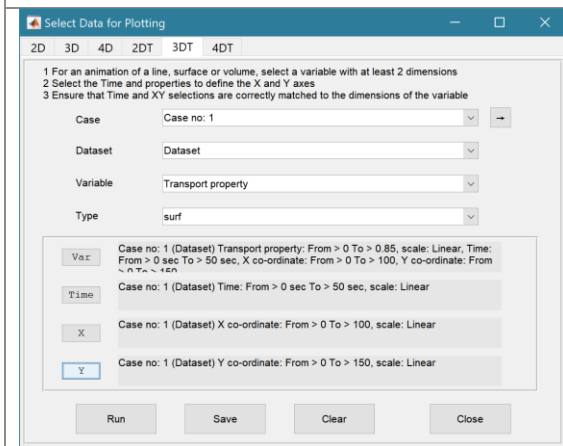
> Select plot type (line, bar, scatter, stem, etc)

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 particular App or the muitoolbox. The sample code can be found in the pfunctions 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.

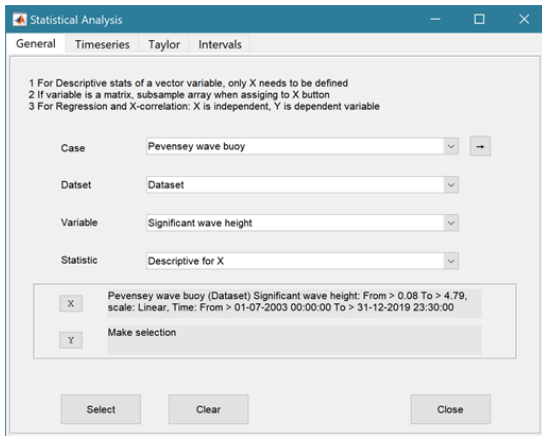
3.6.2 Statistics

Analysis > Statistics: several statistical analysis options have been included within the Statistical Analysis GUI. The tabs are for **General** statistics, **Timeseries** statistics, model comparisons using a **Taylor** Plot, and the generation of a new record based on the statistics over the **Intervals** defined by another timeseries.

General tab

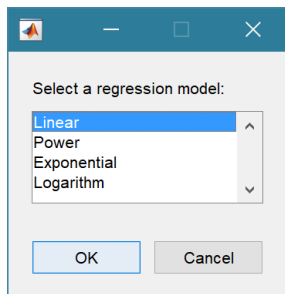
The General tab allows the user to apply the following statistics to data loaded in ModelUI:

- 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.



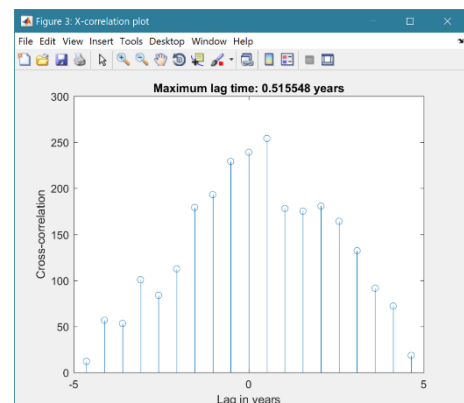
sum and linear regression fit parameters). Only X needs to be defined. The range of the variable can be adjusted when it is assigned to the X button (see Section 3.9). If the variable being used is a multi-dimensional matrix (>2D), the user is prompted to define the range or each additional dimension,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

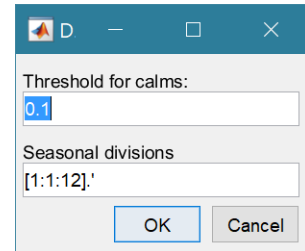
Timeseries tab

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 be 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.

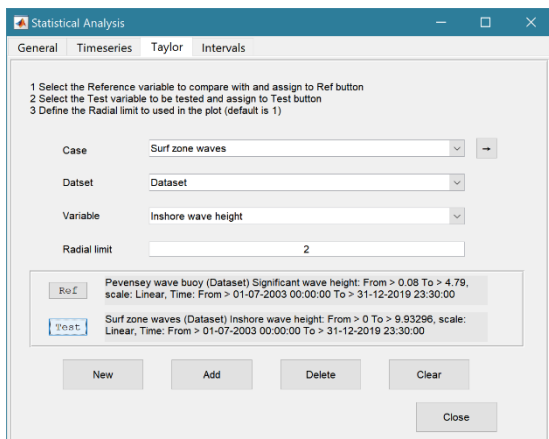
- 5) **Poisson Stats:** user is prompted to select a threshold, method and peak separation (see Peaks above) and the function generates a plot of the peak magnitude, time between peaks (interarrival time) and the duration above the threshold for each peak. The plot shows a histogram of each variable and the exponential pdf derived from the data, along with the μ value for the fit.
- 6) **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 mutoolbox. 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):

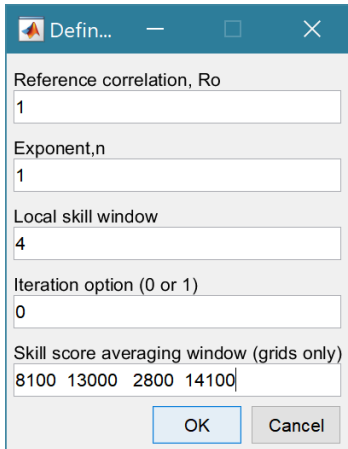
A Reference dataset and a Test dataset are selected. Datasets need to be the same length if 1D, or same size if 2D. If the data are timeseries they are clipped to a time-period that is common to both, or any user defined interval that lies within this clipped period. The statistics (mean, standard deviation, 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).

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 (Dataset) 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).

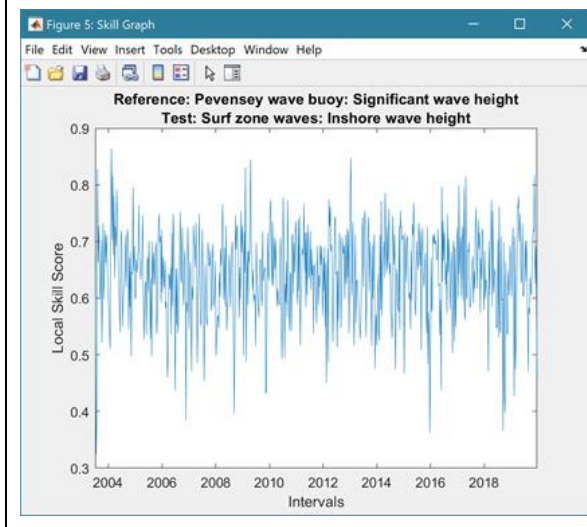
Exponent used in computing the skill score (see ModelSkill manual for details).

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.

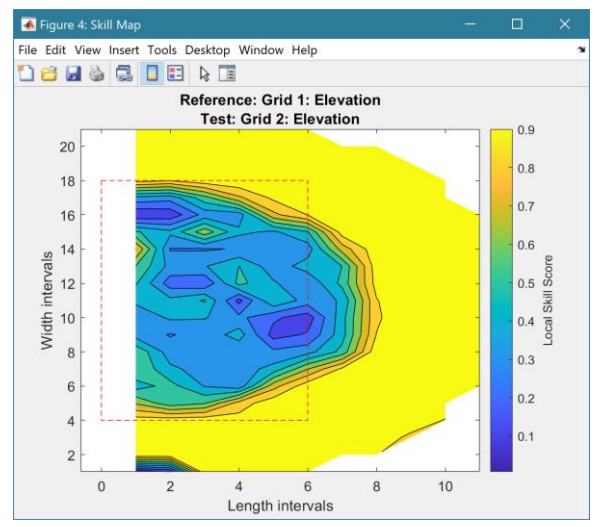
Local skill score is computed for window around every grid cell (=1), or computes score for all non-overlapping windows (=0)

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

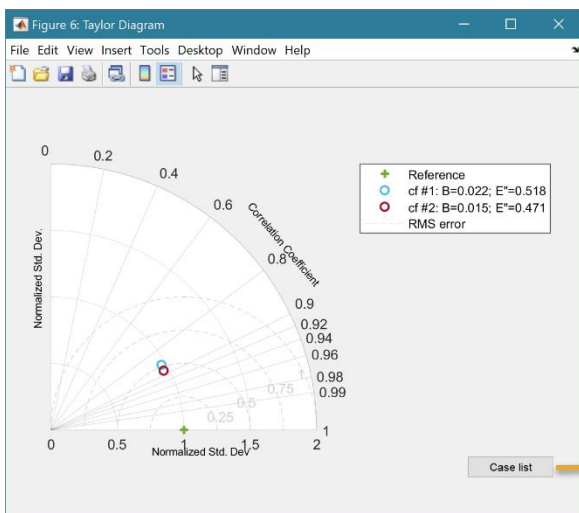
(a) time series skill score plot



(b) grid skill score plot

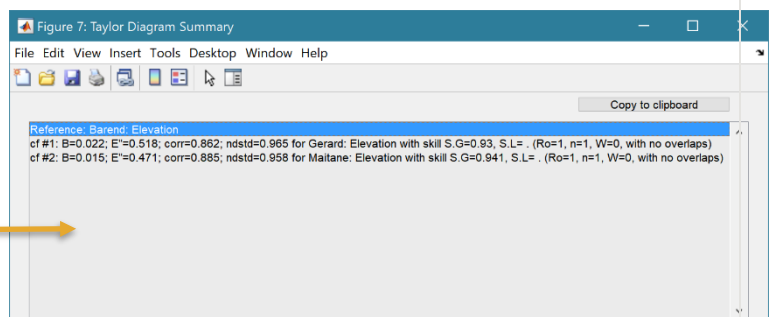


The Taylor Plot shows the Reference point as a green cross and the Test points as coloured circles. The legend details the summary statistics and the Case List button generate a table figure listing all the results. These can be copied to the clipboard.



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, S_g, and the average local skill score, S_l.



3.7 Help

The help menu initialises the App documentation in the Matlab™ Supplemental Software documentation.

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).

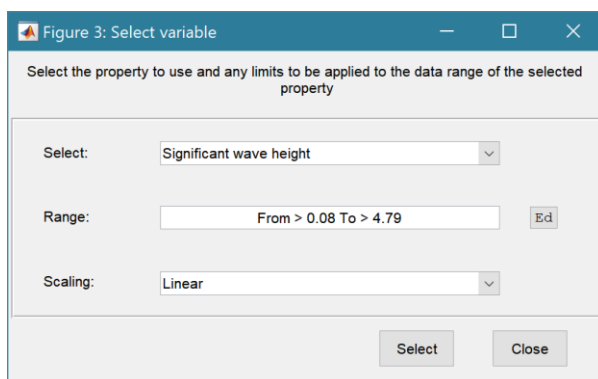
Inputs: tabulates the system properties that have been set (display only).

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:



Step 1.

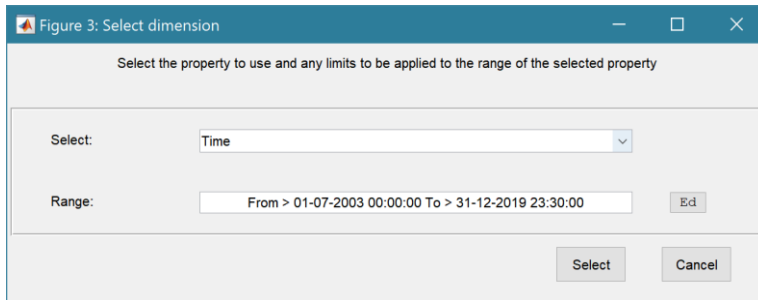
Select the attribute to use. This can be the variable or any of its associated dimensions, which are listed in the drop-down list.

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.

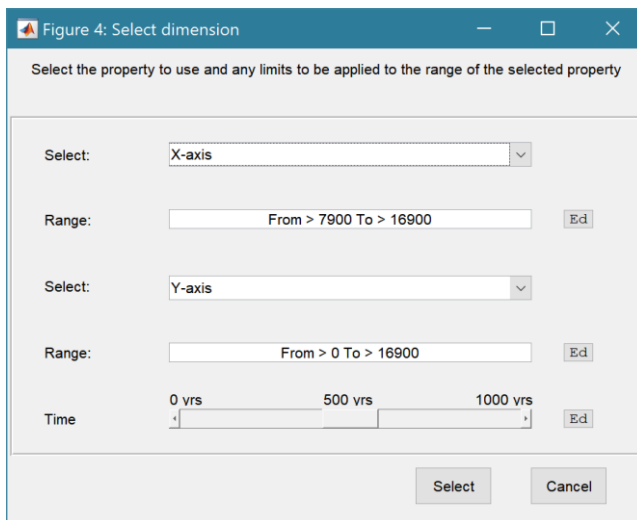
Press Select to go to the next step or Close to quit.

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.



Step 2 - Variable only has dimension of time.

No selection to be made.
Edit range if required.



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

Select the 1st variable to use as a dimension,
Edit range if required.

Select the 2nd 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 Form Model Output Table Content

The following tables are generated when a Form model is created and Properties are added.

3.10.1 Form Table

Elevations of grid along with the x and y dimensions, with the following variables:

- Z - Elevation (mAD)

Dimensions of time (table rows), X and Y.

UserData.ishead - orientation of x-axis relative to mouth (true if minimum x is at the head of the estuary/inlet, default is false).

UserData.xM - distance from grid origin to mouth of estuary/inlet (default is 0).

UserData.cline – struct for x and y coordinates of meander centre-line.

Description – user assigned Case description.

Source - source file or model class.

MetaData - details any manipulation eg type of model or grid rotation.

3.10.2 Hypsometry Table

Variation of surface area and volume as a function of elevation, with the following variables:

- Volume - Volume (m^3)
- SurfaceArea - Surface area (m^2)
- SAfreq - Surface area frequency

Dimensions of time (table rows), and Z.

Description – user assigned Case description.

Source - source file or model class.

MetaData - details any manipulation eg type of model or grid rotation.

3.10.3 SectionProps Table

Variation of the width and cross-sectional area along the x-axis derived from the gridded data, with the following variables:

- Whw - High Water Width (m)
- Wmt - Mean Tide Width (m)
- Wlw - Low Water Width (m)
- CSAhw - High Water Cross-sectional Area (m^2)
- CSAmt - Mean Tide Cross-sectional Area (m^2)
- CSAlw - Low Water Cross-sectional Area (m^2)
- Dhwh - High Water depth (m)
- Dmt - Mean Tide depth (m)
- Dlw - Low Water depth (m)
- PrA - Tidal Prism (using CSA) (m^3)
- Shw - High Water surface area (m^2)
- Smt - Mean Tide surface area (m^2)
- Slw - Low Water surface area (m^2)
- Vhw - High Water volume (m^3)
- Vmt - Mean Tide volume (m^3)
- Vlw - Low Water volume (m^3)
- PrV - Tidal Prism (using hypsometry) (m^3)
- Gamma - Dronkers' Gamma (-)
- Vs - Storage Volume (m^3)
- Vc - Channel Volume (m^3)
- amp - Tidal amplitude at mouth (m)
- hyd - Hydraulic depth of channel (m)

Dimensions of time (table rows), and X.

Description – user assigned Case description.

Source - source file or model class.

MetaData - details any manipulation eg type of model or grid rotation.

3.10.4 Plan Table

Widths along the x-axis, specified in the model, or based on an exponential fit to the gridded data, with the following variables:

- Whw - High Water Width (m)
- Wmt - Mean Tide Width (m)
- Wlw - Low Water Width (m)

Dimensions of time (table rows), and X.

Description – user assigned Case description.

Source - source file or model class.

MetaData - details any manipulation eg type of model or grid rotation.

3.10.5 GrossProps Table

Various summary properties of the form related to length, area, volume, rate of convergence, etc, with the following variables:

- Shw - High Water Surface Area (m²)
- Slw - Low Water Surface Area (m²)
- Vhw - High Water Volume (m³)
- Vlw - Low Water Volume (m³)
- PrA - Tidal Prism using cross-sectional areas (m³)
- PrV - Tidal Prism using hypsometry volumes (m³)
- Gamma - Dronkers' Gamma (-)
- Vs - Storage Volume (m³)
- Vc - Channel Volume (m³)
- Wm - Mean Tide Width at mouth (m)
- Am - Mean Tide Cross-sectional Area at mouth (m²)
- Dm – Depth at mouth to MTL (m)
- amp - Tidal amplitude at mouth (m)
- hyd - Hydraulic depth of channel (m)
- aoh - Amplitude to Depth ratio (-)
- VsoVc - Storage to Channel Volume ratio (-)
- PrvAm - Prism to Cross-sectional Area ratio at mouth (m)
- SflShw - Intertidal to Basin Area ratio
- Lw - Width convergence length

- La - Cross-sectional Area convergence length

Dimensions of time (table rows).

Description – user assigned Case description.

Source - source file or model class.

MetaData - details any manipulation eg type of model or grid rotation.

3.10.6 WaterLevels Table

Variation in water levels at high water, mean tide and low water along the x-axis, with the following variables:

- zhw - High Water level (mAD), where mAD = metres above Datum
- zmt - Mean Tide level (mAD)
- zlw - Low Water level (mAD)

Dimensions of time (table rows), and X.

Description – user assigned Case description.

Source - source file or model class.

MetaData - details any manipulation eg type of model or grid rotation.

3.11 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 *dtables* 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 Implementation

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

4.1 Taylor diagram

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:

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

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

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:

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

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 ϕ 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:
$$\hat{E}' = \frac{E'}{\sigma_{obs}}$$

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

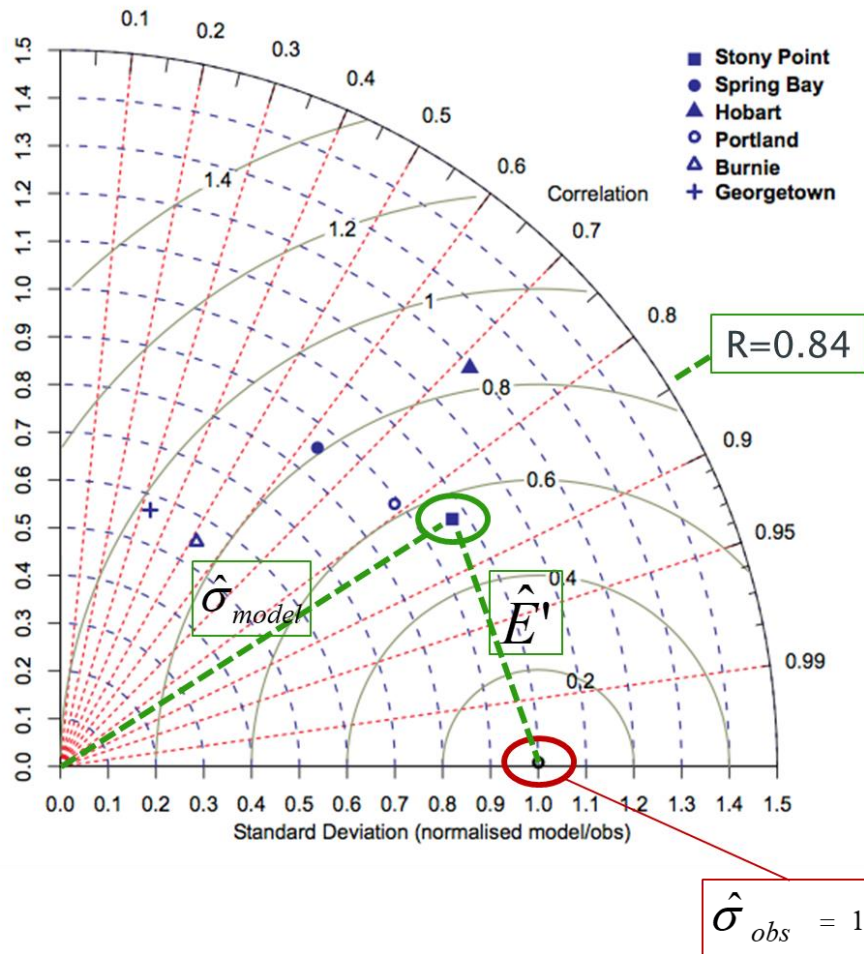
$$\hat{\sigma}_{obs} = 1$$

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

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

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

NB: Normalised RMS difference, \hat{E}' is denoted E'' and bias, \bar{E} , as B in ModelSkill graphical output.



4.2 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 the 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, 'S.G'. Bosboom and Reniers (2014) also make use of spatially local estimates of the skill score, 'S.L', 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. (Note: a similar one-dimensional window can be applied to a vector when comparing timeseries data).

This is the approach adopted in ModelSkill. The code takes grids of size $2W \times 2W$, where W is defined in Run Parameters, such that the grids do not overlap (i.e. a subset of points are used). This can be modified to sample a grid around every internal point by changing the skill score iteration option in the Run Parameter settings. For large grids using non-overlapping grids is faster. The average skill score, $S.L$, 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.

4.3 Brier Skill Score relative to initial morphology

The performance of mathematical models is often judged based using the Brier Skill Score. This measure the change relative to an initial bed of the resultant observed bed and the resultant model bed. A positive score suggests that the modelling is doing better than simply assuming no change and a negative score suggests that assuming no change from the initial bed would be a better option! The Brier Skill Score has the form (Sutherland *et al.*, 2004):

$$\text{BSS} = 1 - \frac{\langle (Y - X)^2 \rangle}{\langle (B - X)^2 \rangle}$$

where B are the initial observed depths, X are the final observed depths and Y are the final predicted depths, compared on a point by point basis

However, in two papers Bosboom et al (2014; 2018) explain why using the initial bathymetry as a reference from which the changes predicted by different model simulations are judged may not be well founded.

"This means that for the zero change model to be an adequate reference model enabling cross-comparison and absolute ranking of predictions, the net bed changes from the start time of the simulations must represent an evaluator's judgments about the difficulty of predictions for different situations and simulation times." ...

"... observed cumulative bed changes are not likely to be a proper indicator of the inherent ease or difficulty of a morphological prediction, since they do not reflect the nature of the morphological development prior to the evaluation time, but only its cumulative effect" (Bosboom et al, 2014).

For these reasons we have elected to only implement the form of Skill Score proposed by Taylor as outlined above. However, a modified form of the Brier Skill can easily be implemented in ModelSkill by:

1. Loading the 3 grids (initial, observed and modelled);
2. Using *Run>Derive Output* UI to obtain new grids for the observed and modelled bed level change;
3. Use the bed level change grids in the Taylor Diagram to obtain an equivalent skill score.

4.4 Statistical significance

Taylor (2001) suggests that for many applications a visual inspection of the Taylor diagram can suffice to distinguish relative performance. Where multiple data points are available a formal test for statistical significance could be performed based on the spread of points. Alternatively, some form of bootstrapping could be undertaken by sub-sampling the observed and modelled timeseries. This is similar to the local skill score that has been implemented. By varying the window width, W , one gets a measure of the influence of scale (in time for timeseries data and in space for gridded data) on the resultant scores.

When examining model stability/predictability, one option would be to compute the skill score at a series of time steps over each model simulation. The skill score vector should then give an indication of whether different model simulations are converging to (or maintaining) a comparable level of skill or diverging. [*This has not yet been implemented*].

4.5 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 4.5.2).

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 4.5.2 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 `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™ expression, or function call, can be used to create a new time series as follows:

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

4.5.1 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 ModelSkill are detailed in Section 4.5.3.

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 4.5.2). 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 `dstable`s, 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 `dstable`s to the function. Using this syntax, the function can return a `dstable` or struct of `dstable`s, or as variables, containing one or more data sets.

4.5.2 Input and output format for external functions

There are several possible use cases:

4.5.2.1 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
```

4.5.2.2 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
qdrift = array1;
qtime = 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
```

4.5.2.3 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
```

4.5.2.4 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 muitoolbox. 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.

4.5.3 Pre-defined functions

The following examples are provided within ModelSkill, 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 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. **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.

3. **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).

5 Input formats

5.1 Grid and Mesh data

Input data for grid and mesh data are defined in an ASCII text file as rows of X, Y, Z values. The input format is defined using Matlab script in header line. The values can use any separator but this should be included in the format definition given in the header.

```
%f, %f, %f
1920, -7200, 3.044
1600, -7200, 2.757
1280, -7200, 2.474
960, -7200, 2.195
640, -7200, 1.921
320, -7200, 1.651
0, -7200, 1.385
```

5.2 Timeseries data

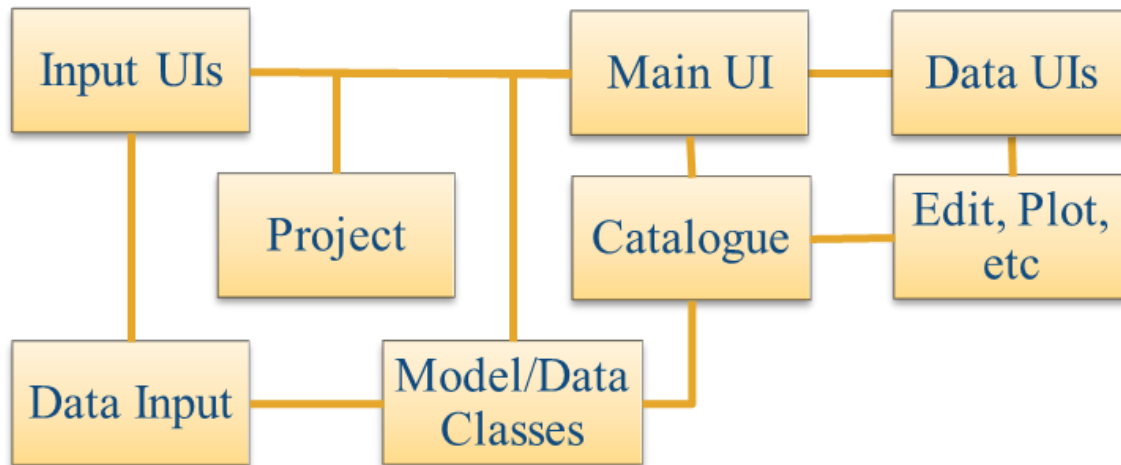
Timeseries data use a generic Date-Record format. The input format is defined using Matlab script in header line. Format shown here is date, time, wind speed (m/s) direction (degTN). This allows different date and record formats to be easily handled.

```
{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
```

6 Program Structure

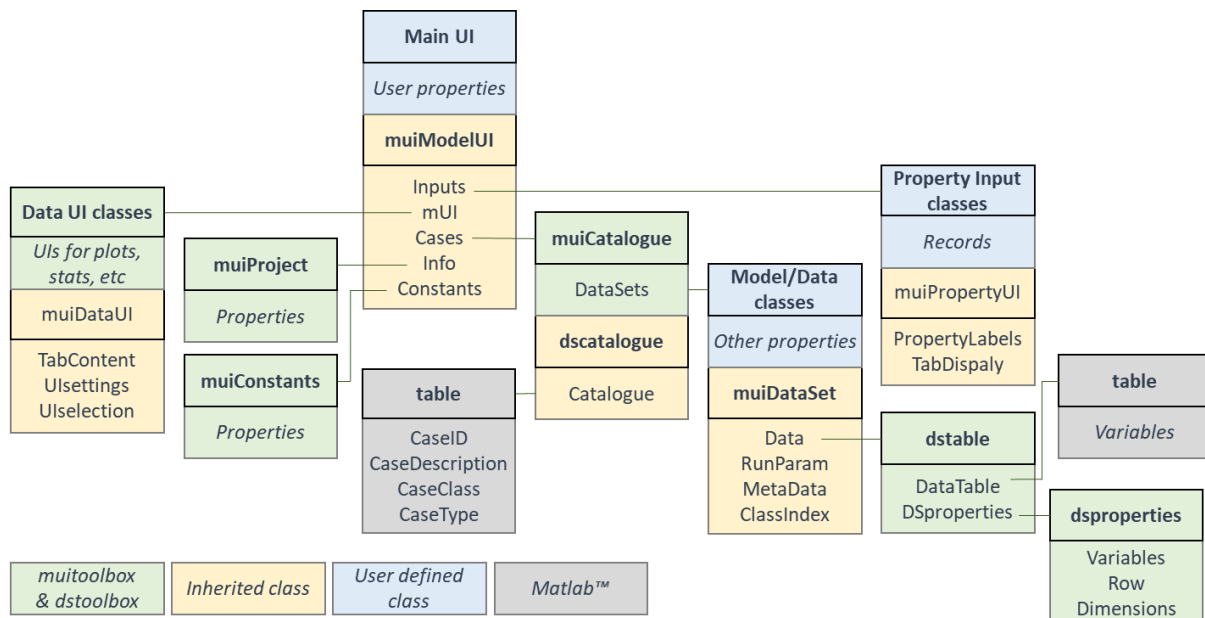
The overall structure of the code is illustrated schematically in Figure 1. This is implemented through several classes that handle the graphical user interface and program workflows (Main UI) and several classes that handle the data manipulation and plotting (Input UIs and Data UIs).

Figure 1 – High level schematic of program structure



The interfaces and default functionality are implemented in the ModelSkill 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 ModelSkill App uses the following classes and functions:

ModelSkill – bespoke UI for the model

muiUserData – class in the mui toolbox used to hold timeseries data of user specified format. A format files such as the *demo_format.m* included in the examples folder is needed to load a specific data format.

MS_RunParams – defines dimensions for re-gridding and skill score parameters

Model functions

Functions used in ModelSkill:

example/demo_format – used to set up import format and metadata description of timeseries variables

getInletTools - tools to extract and plot morphological properties of inlets and channels

getTaylorPlot – selects data and calls the *taylor_plot* function om the *mui*toolbox.

getUserTools – user functions to do additional analysis on data loaded in ModelSkill

network_count – extract channels from a bathymetry and perform some network analysis (function derived from GeoNet toolbox by Barend van Maanen (Apr 2020) – see Sangireddy *et al* (2016) and <https://sites.google.com/site/geonethome/source-code>)

Grid classes

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.

Grid functions

Functions used to manipulate cartesian grids can be found in the *muiAppGridFcns* folder and includes 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 *mui*toolbox 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 Property 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.

*getconvergence*length - 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 Property 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.

*getconvergence*length - 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.

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>.⁴

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

⁴ 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

- Bosboom J and Reniers A, 2018, The Deceptive Simplicity of the Brier Skill Score, In: Handbook of Coastal and Ocean Engineering, Series, pp. 1639-1663.
- Bosboom J and Reniers A J H M, 2014, Scale-selective validation of morphodynamic models, 34th International Conference on Coastal Engineering, pp. 1911–1920, Seoul, South-Korea.
- Bosboom J, Reniers A J H M and 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>.
- Coles S, 2001, An Introduction to Statistical Modeling of Extreme Values, Springer-Verlag, London.
- Passalacqua P, Do Trung T, Fofoula-Georgiou E, Sapiro G and Dietrich W E, 2010, A geometric framework for channel network extraction from lidar: Nonlinear diffusion and geodesic paths. Journal of Geophysical Research, 115 (F1), 10.1029/2009jf001254.
- Perona P and Malik J, 1990, Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12 (7), 629-639, 10.1109/34.56205.
- Sangireddy H, Stark C P, Kladzyk A and Passalacqua P, 2016, GeoNet: An open source software for the automatic and objective extraction of channel heads, channel network, and channel morphology from high resolution topography data. Environmental Modelling & Software, 83, 58-73, 10.1016/j.envsoft.2016.04.026.
- Sutherland J, Peet A H and Soulsby R, 2004, Evaluating the performance of morphodynamic modelling. Coastal Engineering, 51 (8-9), 917-939.
- 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.