



## Preface

EstuaryDB is a Matlab<sup>TM</sup> App that enables tables of estuary data (multi-estuary) to be loaded. In addition, estuary specific datasets that are vector or matrix can also be loaded, such as along-channel data, bathymetry and images. Several be-spoke plotting functions are included to examine the variation amongst estuaries, including cross-plots to examine variations about central values (e.g., HW, MT, LW).

## Requirements

The model is written in Matlab<sup>TM</sup> and provided as Open Source code (issued under a GNU General Public License) and runs under v2016b or later. EstuaryDB uses the *multitoolbox* and *dstoolbox*, both available from <https://github.com/CoastalSEA>.

## Resources

The EstuaryDB App and two toolboxes (*multitoolbox* and *dstoolbox*) can be downloaded from <https://github.com/CoastalSEA>.

*Cite as:*

Townend, I.H., 2021, EstuaryDB manual, CoastalSEA, UK, pp50, [www.coastalsea.uk](http://www.coastalsea.uk).

## Bibliography

Manning A J, 2012, Enhanced UK estuaries database: explanatory notes and metadata, Report No: TR167, pp. 1-20, Wallingford. <https://www.estuary-guide.net/pdfs/TR167.pdf>

Whitehouse R J S, 2006, Review and formalisation of geomorphological concepts and approaches for estuaries, Joint Defra/EA Flood and Coastal Erosion Risk Management R&D Programme, Report No: FD2116/TR2 prepared by HR Wallingford, ABPmer and Prof Pethick, pp. 1-335, [www.estuary-guide.net/pdfs/FD2116\\_TR2.pdf](http://www.estuary-guide.net/pdfs/FD2116_TR2.pdf), [www.estuary-guide.net/pdfs/FD2116\\_TR2.pdf](http://www.estuary-guide.net/pdfs/FD2116_TR2.pdf)

Townend I H, 2005, An examination of empirical stability relationships for UK estuaries. Journal of Coastal Research, 21 (5), 1042-1053. <https://www.jstor.org/stable/4299504>

Townend I H, 2007, The Estuary-Guide, Website prepared for the joint Defra/EA Flood and Coastal Erosion Risk Management R&D Programme, [www.estuary-guide.net](http://www.estuary-guide.net).

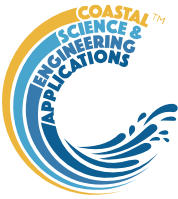
## Acknowledgements

Staff at ABPmer, HR Wallingford, University of Southampton and the Environment Agency have all contributed to the development of the Estuary Guide and the UK Estuary Database [www.estuary-guide.net](http://www.estuary-guide.net).



## Revision history

Version	Date	Changes
2.0	Mar 2025	Added functionality to load grids and extract surface area and width hypsometry. Load tables for tidal levels and river discharge and compute gross morphological properties and derived properties.
1.2	Dec 2024	Updated to use TableViewer table import of flat tables and bespoke options for import of vector, matrix and image data.
1.1	May 2024	Ported to use muitoolbox
1.0	Aug 2019	First release via <a href="http://www.coastalsea.uk">www.coastalsea.uk</a>



## Contents

1	Introduction .....	1
2	Getting started .....	2
2.1	Configuration .....	2
2.1.1	Installing the toolboxes.....	2
2.1.2	Installing the App .....	2
2.2	Quick Guide .....	2
2.3	Typical Workflows .....	3
2.3.1	Tabular Properties.....	3
2.3.2	Surface Area Hypsometry .....	3
2.3.3	Width Hypsometry .....	3
2.3.4	Gross Properties.....	4
2.3.5	Illustrated Workflow.....	4
3	Application Menus .....	10
3.1	File .....	10
3.2	Clear.....	10
3.3	Project .....	10
3.4	Setup .....	11
3.4.1	Table Data .....	11
3.4.2	Spatial Data .....	13
3.4.3	Hydraulic Properties .....	14
3.4.4	Grid Tools.....	15
3.4.5	Section Tools .....	16
3.4.6	Other Setup options .....	18
3.5	Tools .....	18
3.5.1	Hypsometry and Gross Properties .....	18
3.5.2	Derived Output .....	19
3.5.3	User Tools .....	20
3.6	Analysis .....	20
3.6.1	Plotting .....	20
3.6.2	Statistics.....	21
3.6.3	Bespoke Plots .....	22
3.7	Help.....	22
3.8	Tabs.....	23
3.9	UI Data Selection .....	23
3.10	Accessing the tables .....	24
3.11	Accessing data from the Command Window .....	25



---

4	Supporting Information .....	27
4.1	Loading spatial data .....	27
4.2	Derive Output.....	27
4.2.1	Calling an external function .....	28
4.2.2	Input and output format for external functions .....	28
4.2.3	Pre-defined functions.....	31
5	Program Structure.....	33
6	Bibliography.....	36
	Appendix A - Import file formats.....	38
	Appendix B - Data set properties (DSproperties).....	39
	Appendix C – Sample DSproperties for data import.....	40
	Appendix D – Bespoke data import format file .....	42
	Appendix E – Estuary Property Example Formats.....	46

## 1 Introduction

EstuaryDB is a Matlab™ App that enables tables of estuary data (multi-estuary) to be loaded. In addition, estuary specific datasets that are vector or matrix can also be loaded, such as along-channel data, bathymetry and images. Several be-spoke plotting functions are included to examine the variation amongst estuaries, including cross-plots to examine variations about central values (e.g., HW, MT, LW).

The original work on UK estuaries was undertaken by Davidson et al (Davidson *et al.*, 1991). Similar work on classification has been undertaken in other countries such as New Zealand (Hume and Herdendorf, 1988; Hume *et al.*, 2007) and the United States of America (Hansen and Rattray, 1966; Pritchard, 1989; Geyer and MacCready, 2014). In the UK, the Estuary Research Programme that ran from 1998-2007. An early version was a byproduct of the EMPHASYS project and led to two papers that provided data and guidance (Townend, 2004; 2005). These concepts were then encapsulated in a portal published by [Associated British Ports \(ABP\)](#) and known as the Estuary Guide ([www.estuary-guide.net](http://www.estuary-guide.net)). Subsequent development of the portal was funded by the [Department of Food, Environment and Rural Affairs \(Defra\)](#), the [Environment Agency \(EA\)](#) and [Natural England \(NE\)](#). The portal has undergone updates over time funded through joint Defra / Environment Agency Flood and Coastal Erosion R&D Programme.

There have been many contributors to this work, drawing on a wide range of experience. These include: ABPmer and HR Wallingford are grateful to our many partners in various aspects of this work, including: Black & Veatch, British Geological Survey, CEFAS, Centre for Ecology and Hydrology, Halcrow, Plymouth Marine Laboratory, Posford Duvivier, Proudman Oceanographic Laboratory, Royal Holloway University of London, University College London, University of Cardiff, University of Durham, University of Liverpool, University of Newcastle, University of Nottingham, University of Plymouth, University of Southampton and WL|Delft Hydraulics Laboratory.

One of the key limitations of this work is the quality of the data available in the early 2000's to create detailed digital terrain models of the estuaries. Whilst a consistent approach was used to determine properties such as volumes and surface area, there were known errors based on comparisons with more detailed data sets (see Townend, 2005). With the advent of more detailed information for both estuary form but also tidal levels and river discharges, there is now an opportunity to revisit the large-scale, or gross, properties of estuarine systems. The tools provided in the EstuaryDB aim to provide a consistent approach that can be adopted to investigate the morphology of estuary systems.



## 2 Getting started

### 2.1 Configuration

EstuaryDB is installed as an App and requires `multoolbox` 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`

`multoolbox`: `multoolbox.mltbx`

The App file: `EstuaryDB.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:

```
>> EstuaryDB;
```

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 Quick Guide

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

*Setup>Import Table Data>Load*

Load some data from an array in a text file, a worksheet of a spreadsheet, or an existing table or dstable from a `.mat` file (see Section 3.4 and Appendix A - Import file formats). The user is prompted for a file and then for a description of the data source (text string) and a name for the dataset (must be a valid Matlab™ variable name, or the code will try to convert the text entered to a valid name).

When loaded into a dstable the data are assigned dsproperties (i.e., name, description, unit, label and qcflag or format – see Appendix B - Data set properties (DSproperties)). By default, these are assigned the variable name and qcflag is set to none. The user is then prompted to load a **File** of definitions (which can be an `.m` file, text file, or Excel file), or use a **UI** to input the definitions manually, or **Skip** this step and use the defaults. For details of how to set the DSproperties and import file formats see Section 3.4 and Appendix C – Sample DSproperties for data import).

*Setup>Import Table Data > Add to Table>Dataset*: 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). For other options see Section 3.4.

Individual tables can be viewed on the *Table* tab and simple default plots are accessed using the *Q-Plot* tab.

*Analysis>Plots*

The imported data can be selected and plotted. By using the **Add** button, additional Cases, Datasets or Variables can be used to define a plot variable, allowing rapid intercomparison of datasets.

## 2.3 Typical Workflows

Several different types of data can be loaded and analysed in the EstuaryDB App. This involves different workflows depending on the type of data being loaded. This may be loading an existing table of morphological properties for several estuaries (scalar data table), or loading or generating hypsometry data for surface area,  $S(z)$ , or width,  $W(x,z)$ . Details of how to access the various types of data are given in Section 3.10.

### 2.3.1 Tabular Properties

A simple table holds scalar values. For example this may be the properties for a number of estuaries and can be in Excel, ASCII text or .mat matlab format. Some example files of this type of data are provided in the .../example folder. When loading from an Excel spreadsheet, the user is prompted to confirm the worksheet, rows and columns to load. For a text file the header is used to define the inputs (see Appendix A - Import file formats). The user is then prompted to load, define or skip the definition of the DSproperties. For a quick look at the data this can be skipped but for more detailed exploration the DSproperties are used extensively in EstuaryDB to provide meta-data about selections. If loading from a file, this can again be in Excel or ASCII text format and once loaded there is the opportunity to check and edit the meta-data for each variable.

To complete the process the user is prompted for a Case description. The record can then be viewed on the **Data** tab and the checked on the **Q-Plot** and **Table > Dataset** tabs.

### 2.3.2 Surface Area Hypsometry

The surface area data can be loaded directly from a spreadsheet or text file (see 3.4.2, 4.1, and Appendix D – Bespoke data import format file). Alternatively, there are two options to create the data set within the App. For both cases load a bathymetry grid (see Section 3.4.2) and make any adjustments using the Grid tools (see Section 3.4.4). The either:

- (i) Load a shape file to define the enclosing waterbody boundary (see Section 3.4.5); or
- (ii) Use the **Setup > Sections > Waterbody** tool to define a suitable polygon boundary (see Section 3.4.5).

Use the **Tools>Hypsometry >Surface area** to generate the new data set and add it to the estuary Case record (see Section 3.5.1). Upon execution a plot of the grid with the enclosing waterbody polygon are shown alongside the surface area and volume hypsometry. The volume hypsometry is not stored in the dataset but calculated when needed. The new dataset can then be viewed on the **Q-Plot** and **Table > Dataset** tabs.

### 2.3.3 Width Hypsometry

The along-channel width hypsometry for a single reach can be loaded directly from a spreadsheet or text file (see 3.4.2, 4.1, and Appendix D – Bespoke data import format file). Alternatively, the tools provided in the App can be used to define the sections and compute the hypsometry for estuaries with multiple reaches. Start by loading a bathymetry grid (see Section 3.4.2) and make any adjustments using the Grid tools (see Section 3.4.4). Then work through the Section tools to create the linework needed to extract cross-sections. The steps are as follows:

- (i) Create a boundary to the estuary using **Setup > Sections > Boundary** (see Section 3.4.5). This sets the extent of the cross-sections. The boundary should be set at an elevation that allows the hypsometry to be computed over the desired range of elevations. However, it does not need to be accurate or detailed because the boundary is only used to adjust the length of the sections. The interface allows the generated contour to be smoothed or resampled. The latter should be done to minimise the size of the line and make editing simpler.

- (ii) Create a centre-line for the channel(s) using *Setup>Sections>Channel Network* (see Section 3.4.5). If there are multiple channels there may be a need to create several lines. Again, precision is not required, as long as the lines follow the channel. The interface allows the generated centre-lines to be smoothed or resampled. The latter is done as the line is created, based on the interval defined when the option is opened but can be adjusted within the interface. The points in the centre-line are used to position the section lines. Consequently, the sampling interval determines the section line interval. However, there is also a need to avoid sections crossing one another. In particular, no section should cross more than one centre-line. This can be adjusted in step (iii) but is helped by starting the centre-line for channel branches within the branch rather than close to the channel that it joins.
- (iii) Once the Boundary and Centre-line have been defined the section lines can be created using *Setup>Sections>Section Lines* (see Section 3.4.5). The section length determines the distance of the section line end points either side of the centre-line and sections are generated at every point on the centre-line. The interface provides options to clip the section lines to the boundary of the channels and edit the lines. At this stage it is important to ensure that there is a centre-line point on every section line, that crossing section lines are edited or deleted and to **make sure that no section line crosses more than one centre-line**.
- (iv) The completed section lines can then be used to generate the cross-section *Setup>Sections>Section* (see Section 3.4.5). A composite plot of all the sections is generated upon completion and the y, z data saved.
- (v) The final preparatory step is to define the channel connectivity using *Setup>Sections>Channel Links* (see Section 3.4.5). Follow the in-figure instructions to select branches the channel point that they connect to. The interface will allow selection of branches and nodes to continue until all branches have been linked into the network. A summary graph of the network topology is plotted and this shows the index of the connecting points the line numbers and the distances from the first point (usually the mouth of the estuary).

The along-channel width hypsometry can now be added using *Tools>Hypsometry >Width* to generate the new data set and add it to the estuary Case record (see Section 3.5.1). Upon execution a plot of the width hypsometry for all reaches combined and a composite plot showing the contribution of individual reaches are generated. The new dataset can then be viewed on the *Q-Plot* and *Table > Dataset* tabs.

### 2.3.4 Gross Properties

Once a hypsometry has been created using either the surface area or the width options, the gross properties can be computed and added to the Gross Properties table (see Section 3.4.3). The results can be viewed on the *Table > Morphology* tab.

### 2.3.5 Illustrated Workflow

The above steps are illustrated with a case study based on the Colne Estuary in the UK.

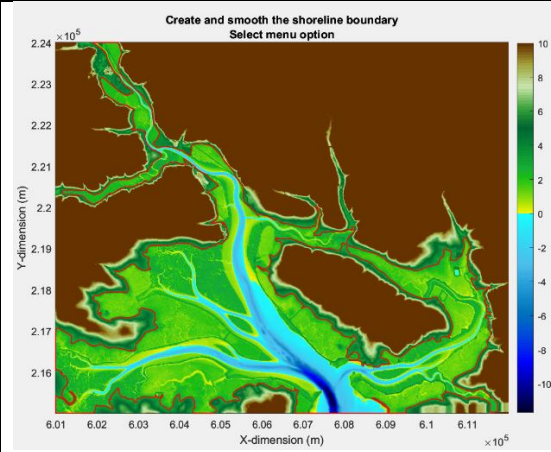
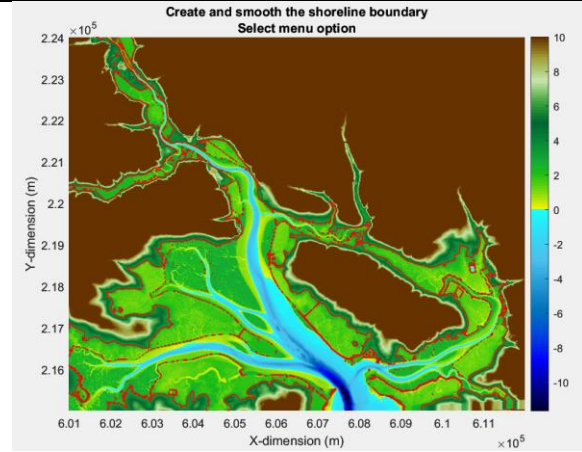


(i) Surface Area Hypsometry

*Setup > Waterbody > Generate*  
contour at 3 mAD (red lines)

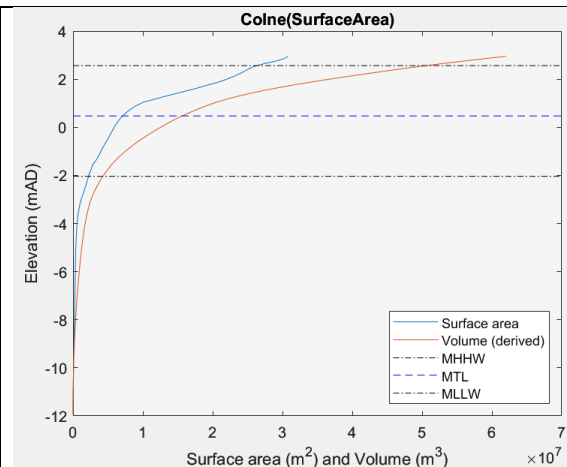
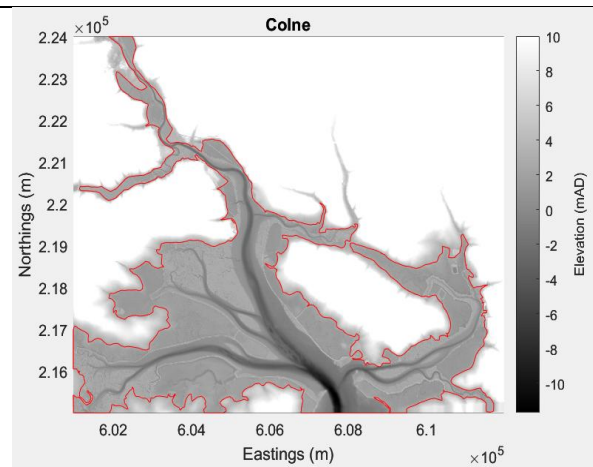
Extract

Resampled at 100m, superfluous lines deleted  
and boundaries closed to form a polygon

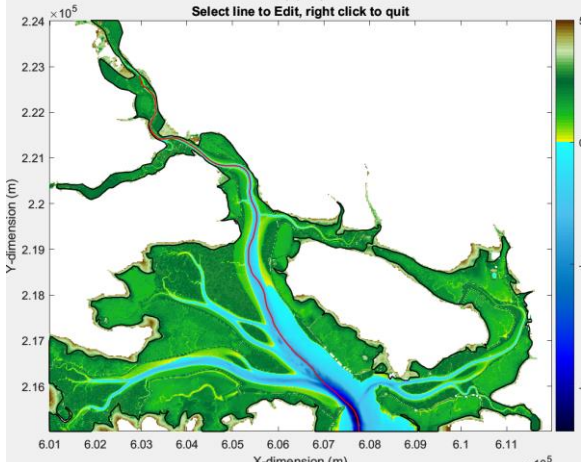
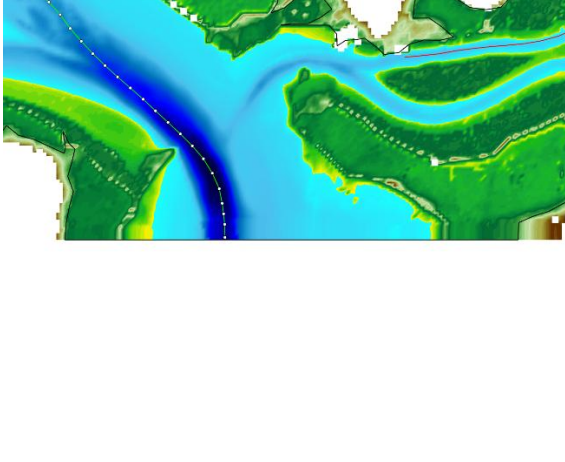
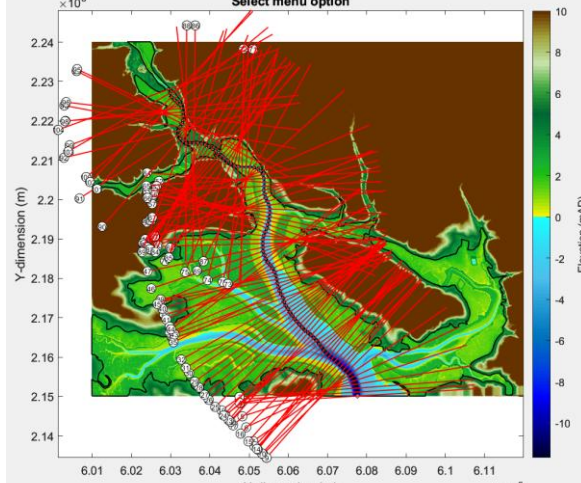
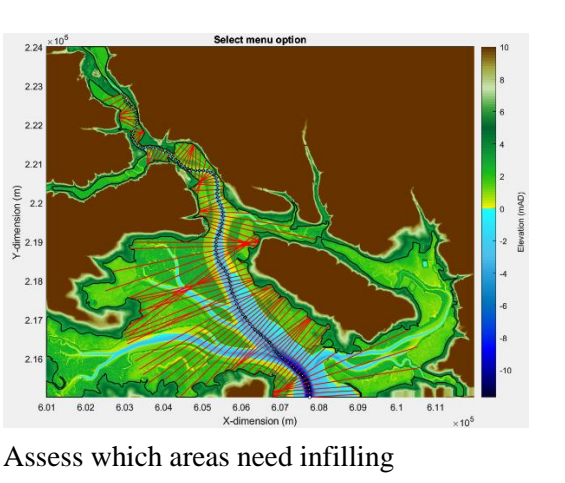


To create the table of surface area hypsometry use: *Tools > Hypsometry > Surface Area*. The results can be viewed on the **Q-Plot** tab.

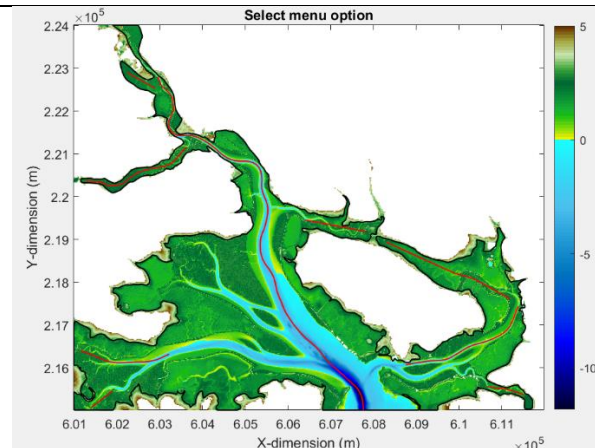
To add the Gross Properties to the Morphology table, first add the tidal levels using *Setup > Estuary Properties > Tidal Levels* and then add the gross properties using *Setup > Estuary Properties > Gross Properties*. The results can be viewed on the **Table > Morphology** tab.



## (ii) Width Hypsometry

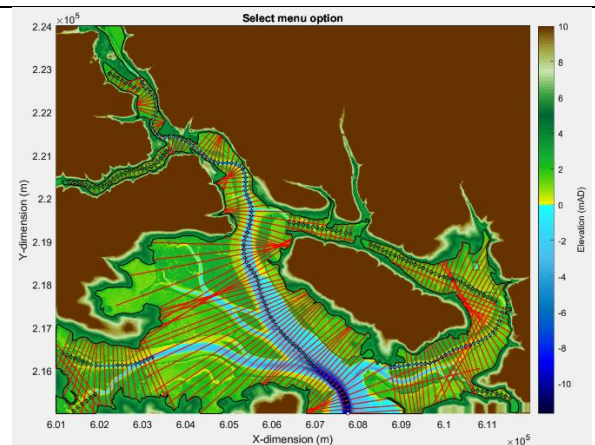
<p><i>Setup&gt;Sections&gt;Boundary&gt;Generate</i></p> <p><i>Setup&gt;Sections&gt;Channel Network&gt;Generate</i></p> <p>Set first line for main channel. Smooth the line (default settings used) – aim to minimise the curvature, as this will reduce the number of intersecting cross-sections. Edit any kinks at the end of lines.</p>	<p>Copy line work for Waterbody</p> <p>Try to get the first 2-3 points of the first line at right angles and approximately straight to the section that will define the mouth section (e.g. Extend line with a new start point and then Resample).</p>
	
<p><i>Setup&gt;Sections&gt;Section Lines&gt;Generate</i></p> <p>Set the initial cross-sections for the main channel using a length of 3000 m.</p>	<p>Cip the lines to limit their extent where they cross the boundary.</p>
	 <p>Assess which areas need infilling</p>

**Setup>Sections>Channel Network>Generate**  
Modify the existing linework. Add additional centre-lines to provide additional lines. This can be a mix of Set lines and Add lines (2 or more points to create additional lines). Smooth and Resample to create set of points that will complete the section coverage of the estuary basin .

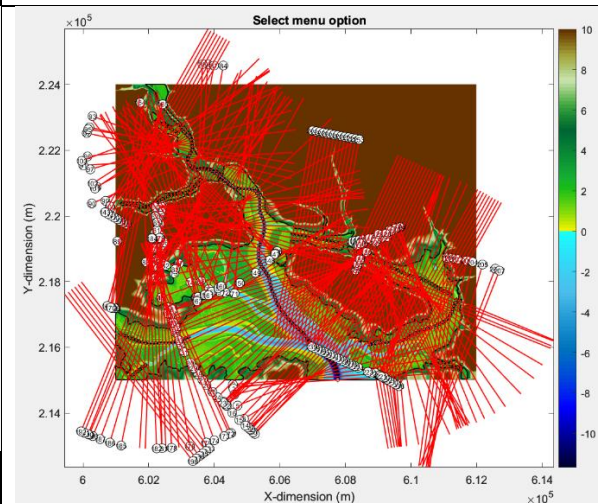


Note that the branch centre-lines are to represent the basin not the meandering channels and do not need to extend to the main channel.

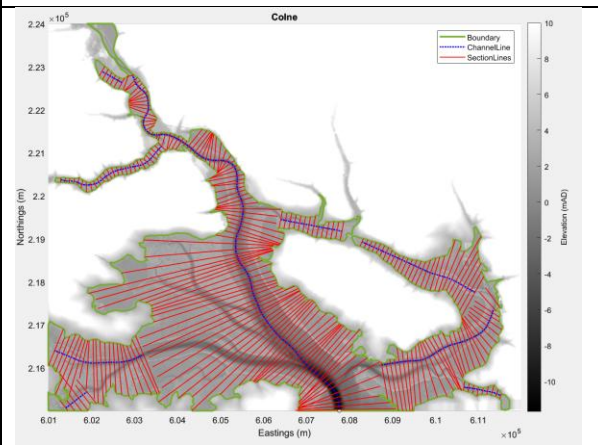
Once the sections lines have been Clipped to the boundary the coverage is clearer. The process of defining centreline points and determining the section coverage that they provide is often an iterative process, especially if there are a number of branches and large intertidal areas.



**Setup>Sections>Section Lines>Generate**  
Reset to remove any existing linework and then Set the sections (a length of 3200 m used to ensure coverage of the tidal flats).



Once a reasonable coverage has been obtained without too many section line crossings, the linework can be edited to give the final section line definition. A composite plot of boundary, centre-lines and sections lines can be viewed using **Setup>Sections>View Sections>Layout**.



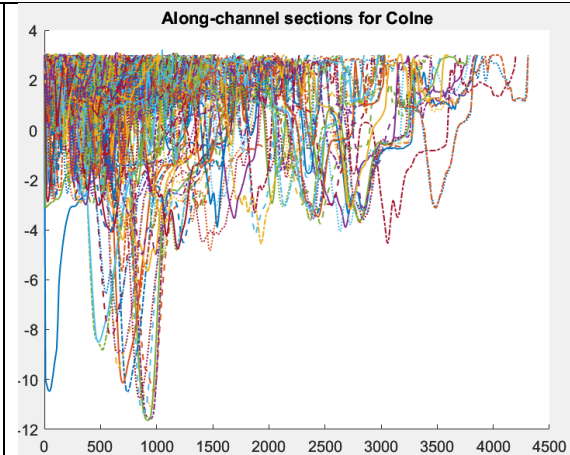
**NB** Adding section lines forces a Reset of the section lines. It is therefore best to make any additions and deletions before undertaking any detailed editing of overlapping sections.



### Setup>Sections >Sections

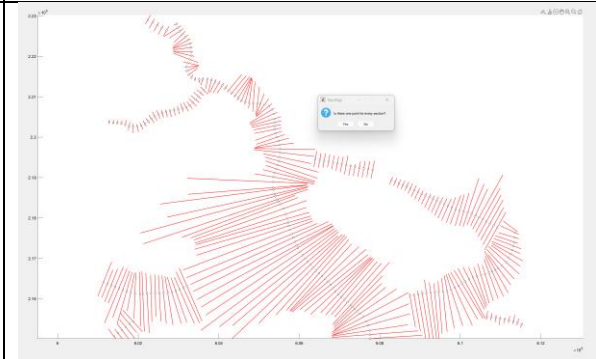
Click on any line to get the section number. Use the figure editing tools to remove unwanted sections.

This graph can be recreated using  
Setup>Sections>View Sections>Sections



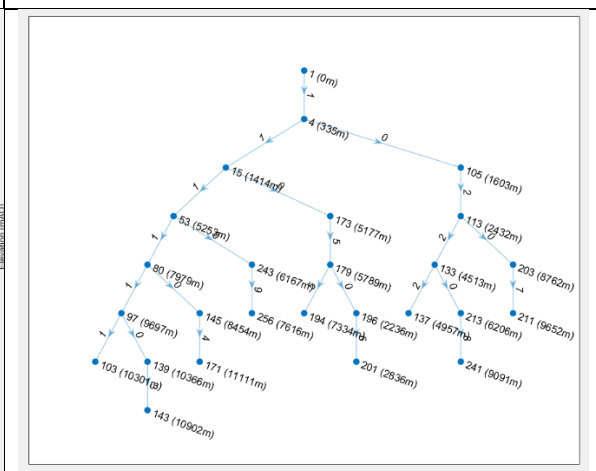
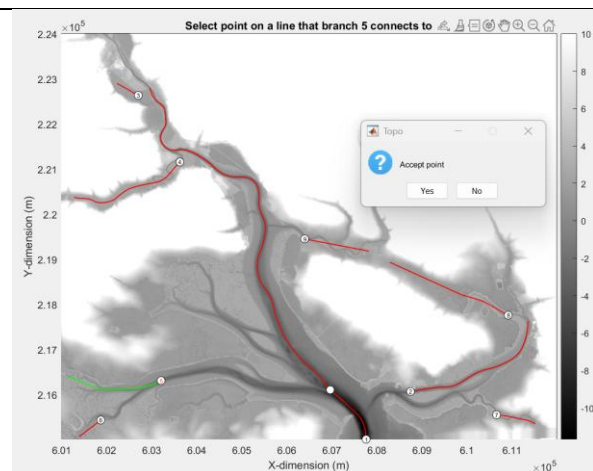
### Setup>Sections >Channel Links

Check that there is only one point (circle marker) on every section (red line).



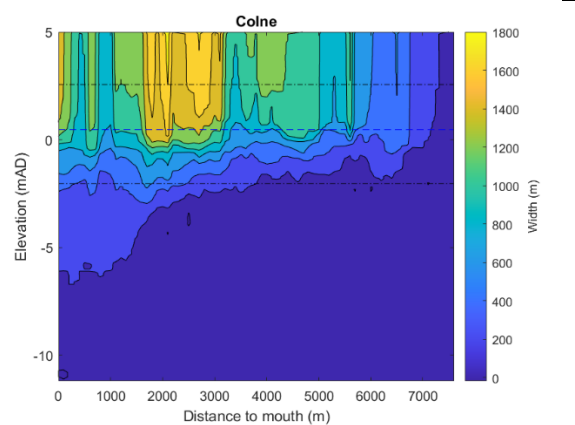
Select branch lines (in any order) and then define the point that the branch connects to. The line numbers turn orange once assigned and the window closes once all connections have been defined.

A graph plot is generated which shows the centreline points at which there are joins and the distance from the first point in paranthesis. The centreline line number is shown on each edge. A value of 0 indicates a join. This graph can be recreated using Setup>Sections>View Sections>Network

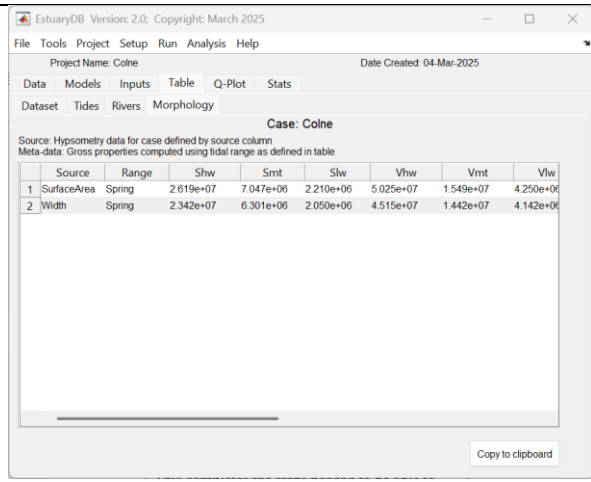


This completes the steps needed to be able to create a table of width hypsometry.

To create the table of width hypsometry use: *Tools > Hypsometry > Width*. The results can be viewed on the **Q-Plot** tab.



To add the Gross Properties to the Morphology table, first check that the tidal levels have been loaded on the **Table > Tides** tab, or add them using *Setup > Estuary Properties > Tidal Levels*. The gross properties can then be added using *Setup > Estuary Properties > Gross Properties* and selecting the Width data set. The results can be viewed on the **Table > Morphology** tab.



### 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 Clear

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

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

*Clear>Clear all>Figures*: deletes all results plot figures (useful if a large number of plots have been produced).

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

#### 3.3 Project

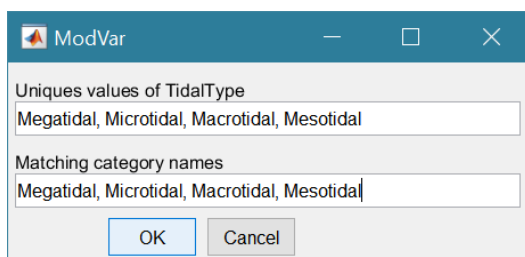
*Project>Project Info*: edit the Project name and Date.

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

*Project>Cases>Edit DS properties*: edit the properties that define the meta-data for a dataset.

*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>Modify Variable Type*: select a variable and modify the data type of that variable. Used mainly to make data categorical or ordinal. First select a variable then the data type required. For text data the next UI lists the categories found for the selected data set. The first input can be modified to the required order. The second input can either match the first, or define alternative category names in the order defined for the first input.



ModVar

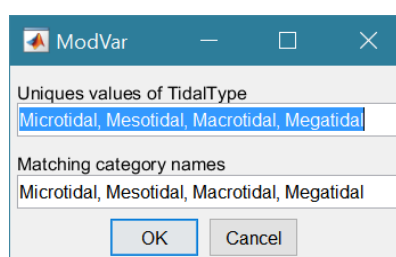
Uniques values of TidalType

Megatidal, Microtidal, Macrotidal, Mesotidal

Matching category names

Megatidal, Microtidal, Macrotidal, Mesotidal

OK Cancel



ModVar

Uniques values of TidalType

Microtidal, Mesotidal, Macrotidal, Megatidal

Matching category names

Microtidal, Mesotidal, Macrotidal, Megatidal

OK Cancel

If the data are already ordinal, selecting cardinal will remove the ordinal setting. Similarly, if already cardinal, selecting ordinal will set this property. If cardinal or ordinal and the ‘type’ selection matches the current type there are two options: (i) if the top box contains the word ‘order’, the definitions in the ‘Matching category names’ entry box are used to reorder the categories, otherwise (ii) the definitions in the ‘Matching category names’ entry box are used to rename the categories.

**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 Case:** select a previous model run and reload the input as the current input settings.

**Project>Cases>View Case 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 select 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.

### 3.4.1 Table Data

**Setup>Import Table Data:** dialog with sub-menu options to Load, Add, and Delete tabular data. This uses the same functionality as the TableView App.

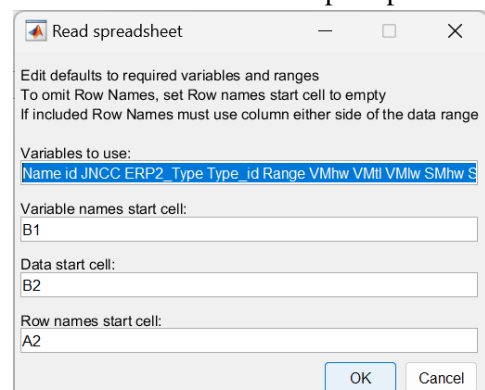
Select a file to load. When the data has been loaded, the user is prompted to provide a description of the data set (a case) and the case is listed on the *Cases* tab.

**Setup>Import Table Data > Load:** the user is prompted for a file. Accepted formats are:

1. A table or dstable created in Matlab and saved as a mat file  
e.g., `>> save('my_table.mat', mytable')`, where the table was assigned to the variable ‘mytable’;
2. As option 1, containing a struct of tables, or dstables, where the fieldnames of the struct define the dataset name assigned to each table.
3. A worksheet in an Excel spreadsheet. If there is more than one worksheet the user is prompted to select one. The Matlab™ *readtable* function is used in `muiTableImport.loadFile` to read the file with the option to `ReadRowNames` set to true. This assumes that the first column of the spreadsheet contains unique names for the data. The variable names must be valid variable names (a valid variable name starts with a letter, followed by letters, digits, or underscores).

The variable names can be edited (number of variables should not change)

Start cell for row of variables



Start cell for array of data

Start cell for column of row names

4. An ASCII text file using the Matlab™ *readtable* function. The header contains the variable names with tab or comma separation (a valid variable name starts with a letter, followed by letters, digits, or underscores). The first column contains the row name (if used) and subsequent columns contain the data with the same separator as the header.

Examples of the spreadsheet and text file formats are provided in Appendix A - Import file formats. The user is then prompted for a description of the data source (text string) and a name for the dataset (must be a valid Matlab™ variable name or code will try to convert the text entered to a valid name).

When data are loaded into a *dstable* the data are assigned *dsproperties* (i.e., name, description, unit, label and *qcflag* or format – see Appendix B - Data set properties (DSproperties)). By default, these are assigned the variable name and *qcflag* is set to none.

The user is then prompted to select an option from File, UI and Skip:

File - loads a file of definitions (.m, text or Excel file)<sup>1</sup>, or

UI - uses a UI to input the definitions,

Skip - skips this step and uses the defaults.

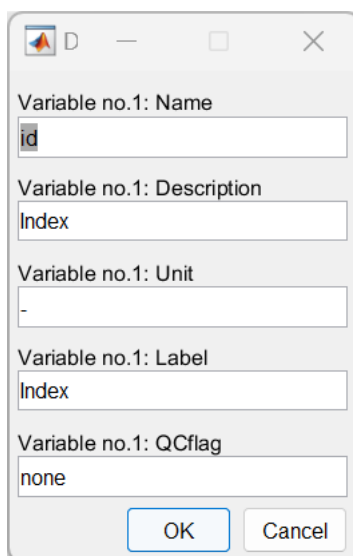
[NB: when loading a struct of tables from a .mat file it is assumed that these already have DSproperties and the UI is used to step through the definitions to provide the opportunity to edit, or modify the definitions.]

### Load DSproperties from file:

A Matlab™ function file that defines a variable that holds a struct of the DSproperties, as explained in Appendix B - Data set properties (DSproperties) and illustrated in Appendix C – Sample DSproperties for data import.

A text file lists the definitions of the variables only as illustrated in Appendix C – Sample DSproperties for data import. These are then scrolled through using the UI (see below) and allows the Description of the rownames to be edited to suit the dataset.

### Load DSproperties using UI:



Data in the imported table is used, where possible, to define the meta-data for each variable. The user is then prompted to either accept the definition to be assigned for each variable in turn. As well as the variable name, this includes a description, units, the label to use when plotting this and similar variables, and the quality control flag for variables, or format for the rowname. The format property defines the date format, when used, using the Matlab identifiers (e.g., 'dd-MM-yyy HH:mm:ss' for 26-Oct\_2024 13:22:00 date style).

NB : the UI prompts for Dimensions with a value of 0. For scalar table data, only rows are used to hold a unique identifier (i.e., a dimension). Accept the default value to continue.

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. All input data sets can be viewed (just one variable at a time) on the *Q-Plot* tab.

<sup>1</sup> Sample files are provided in the /example folder.



Individual tables can be viewed on the *Table* tab and default plots are accessed using the *Q-Plot* tab.

*Setup>Import Table Data >Add to Table*: There are three options for adding data:

*Setup> Import Table Data >Add to Table >Rows*: add rows of data to an existing dataset. Data can be loaded from a text file, table or spreadsheet but the number of variables should be the same as the existing dataset<sup>2</sup>.

*Setup> Import Table Data >Add to Table >Variables*: add variables to an existing dataset. Data can be loaded from a text file, table or spreadsheet but the number of rows should be the same as the existing dataset<sup>3</sup>.

*Setup> Import Table Data >Add to Table >Dataset*: this links another dataset to a selected Case record. The dataset is loaded using the same process as used to *Load data* (see above).

The user is prompted to choose the existing case and/or dataset that the data is to be added to.

The next prompt is to define any likely non-standard missing value indicators. The incoming table/spreadsheet is then concatenated to the existing table. The data added can be as additional rows, in which case the first column of the incoming table is used to define row names and these must be unique for the combined table and the number of variables must match the existing table. Or the incoming table/spreadsheet can contain additional variables, in which case the incoming table must have the same number of rows as the existing table. When variables are added, the user is prompted to accept or update the meta-data for each new variable.

*Setup>Import Table Data >Delete from Table*: There are three options for adding data:

*Setup> Import Table Data >Delete from Table >Rows*: select rows to be deleted.

*Setup> Import Table Data >Delete from Table >Variables*: select variables to be deleted.

*Setup> Import Table Data >Delete from Table >Dataset*: select dataset to be deleted. If there is only one dataset linked to the case you will be prompted to use *Project>Cases>Delete*.

### 3.4.2 Spatial Data

*Setup>Import Spatial Data*:

These menu options are to load data that is not tabular and requires a bespoke format file to load the data. The data can be loaded can be in any form that can be held in a dstable and typically will be vector or matrix data, or images (e.g. jpeg). The load process is described below and the procedure to define a bespoke data format is set out in Section 4.1.

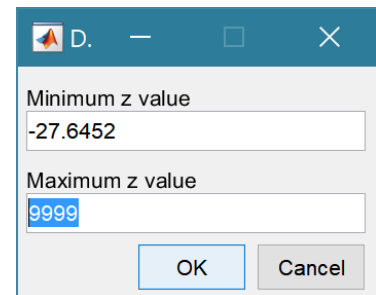
*Setup>Import Spatial Data >Load or Add dataset*: The first prompt is for the type of data to import. Currently the options available include a Bathymetry, Surface Area, Width, Image and GeoImage. These are explained in more detail in Section 4.1. The file name of the dataset define the 'location' used to define the Case and should be unique. When adding datasets with a file that has the name of a location that already exists it will be added to the case as a new dataset, if it is a different type of data, whereas if it is the same as an existing dataset the user is given the option to overwrite the existing dataset, or provide an alternative name for the dataset.

*Setup>Import Spatial Data >Delete dataset*: select dataset to be deleted. If there is only one dataset linked to the case you will be prompted to use *Project>Cases>Delete*.

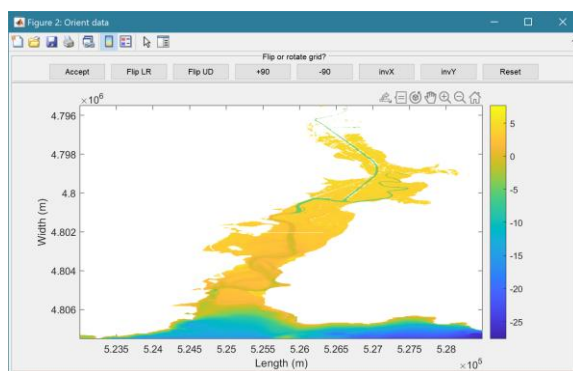
---

<sup>2</sup> Formats for adding data are the same as for loading data, as detailed in the Appendices with sample files provided in the /example folder.

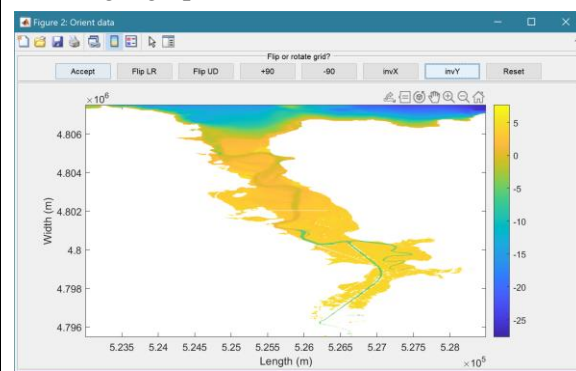
When loading bathymetry, the first prompt presents the maximum and minimum Z values in the grid and the option to limit the range of data to be loaded (values outside the range are set to NaN). The grid is then plotted on a figure with multiple options to rotate or flip the grid, and invert the direction 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.



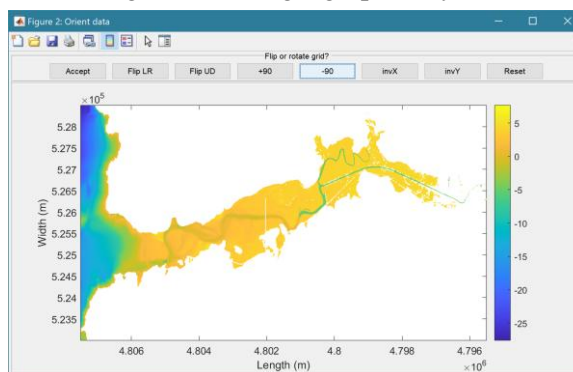
(a) raw data loaded from file



(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.

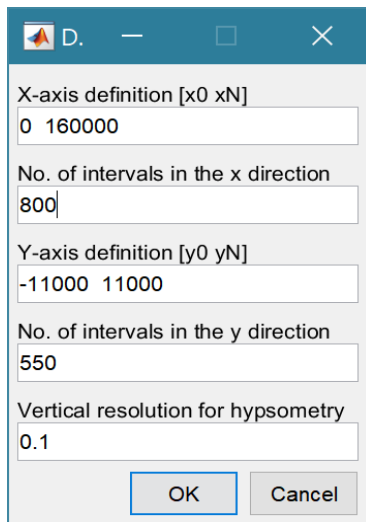
### 3.4.3 Hydraulic Properties

*Setup>Hydraulic Properties*: property tables for tides and river discharge can be loaded, edited and deleted using (*Load, Edit, Delete* submenus for each option):

*Setup> Hydraulic Properties>Tidal Levels*: a table of the elevations that define, for example, spring and neap tides.

*Setup> Hydraulic Properties>River Discharge*: a table of river discharges that define, for example, the annual and seasonal discharges into the estuary.

### 3.4.4 Grid Tools



*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 grid<sup>3</sup>.

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

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

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

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

*Setup>Grid Tools>Infill Surface:* spatial interpolation to infill areas with no data<sup>4</sup>.

*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>Grid Image:* save grid as an image struct with fields 'XData', 'YData', 'CData', 'CMap', 'CLim' (includes option to resize large grids; > 1Mb).

*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.

<sup>3</sup> 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.

<sup>4</sup> Uses inpaint\_nans by John Enrico, which solves approximation to one of several pdes to interpolate and extrapolate holes in an array (see [https://www.mathworks.com/matlabcentral/fileexchange/4551-inpaint\\_nans](https://www.mathworks.com/matlabcentral/fileexchange/4551-inpaint_nans)).

### 3.4.5 Section Tools

*Setup>Sections* A set of tools to create sections along one or more estuary reaches, define the connectivity, extract the sections for use in computing the width hypsometry (*Setup > Add Hypsometry from Grid > Width*). The steps for defining the boundary, centre-lines and section lines have sub-menus as follows:

*Setup>Sections> 'line type' >Generate* - use the built in tools to aid the creation of the lines. This option can be used to modify existing linework, or create new linework. The *Sections > Boundary* tool also has an option to copy the Waterbody linework as an input.

*Setup>Sections> 'line type' >Load* - load a shapefile with the lines to be used.

*Setup>Sections> 'line type' >Edit* - interactively digitise or edit the linework.

*Setup>Sections> 'line type' >Delete* - deletes the saved lines for the selected Case.

The Point and Line tools have a menu that is accessed by right clicking the mouse in the figure but outside the plot axes. This provides access to a set of context sensitive menus. They all include the Figure option, which provides access to the following options:

*Figure > Redraw*: update the line work of points and lines in the plot;

*Figure > Undo*: revert to the previously save linework. If Save has not been used in the session, this will revert to any imported line work;

*Figure > Distance*: display the distance between two points;

*Figure > Save*: save the current points and lines;

*Figure > Save & Exit*: save the current data and exit;

*Figure > Quit*: exit without saving. If new points and lines have been created the user is prompted to confirm that these do no need to be saved.

When using the Edit option there are menus options for points and lines as follows:

*Point > Add*: add one or more points;

*Point > Edit*: edit the position of any existing points;

*Point > Delete*: delete existing points;

*Line > Add*: add one or more lines (each line is terminated with a NaN point);

*Line > Edit*: edit the points in a line;

*Line > Extend*: add points to the end of a line;

*Line > Insert*: insert points within a line;

*Line > Join*: join two lines together;

*Line > Split*: divide a line into two;

*Line > Delete > Points*: delete one or more points in a line.

*Line > Delete > Lines*: delete one or more lines.

For details of the conventions used for points and lines see the Point & Line documentation.

Bespoke tools are provided to create the different line types as follows:

*Setup > Sections > Boundary* - Load or create a polygon, or set of lines, that define the estuary shoreline. This is used to determine the extent of the section lines. Consequently, it does not have to be an exact, or detailed, representation of the actual shoreline but should be positioned at an elevation that captures the desired range for the hypsometry. For example, to examine future change this might be

close to the contour for Highest Tide + 1m to account for surge or sea level rise. Bespoke menus for the Boundary tool include the Line and Figure menus detailed above and:

*Boundary > Resample*: define the sampling interval along the line and resample the linework at the required intervals;

*Boundary > Smooth*: use either a moving average of 'sgolay' method to smooth the lines;

*Boundary > Reset*: clear the existing lines and generate a new contour using NaN or an elevation.

*Setup>Sections>Channel Network* - Individual reaches are captured by defining the end points and using a function to trace the deepest channel between the two points. When this option is selected, the user is prompted to define the maximum accessible water level, the depth exponent (value between 5 and 10 usually works well, whereas smaller values tend to follow the shore in meandering channels), and the sampling interval for the linework that is generated. The menu options provided include Line and Figure as detailed above and the Centreline menu:

*Centreline > Set*: define start and end points and generate a centre-line that attempts to find the shortest distance between the end points constrained to also find the deepest part of the channel.

*Centreline > Resample*: define the sampling interval along the line and resample the linework at the required intervals;

*Centreline > Smooth*: use either a moving average of 'sgolay' method to smooth the lines;

*Centreline > Reset*: clear the existing lines.

*Setup>Sections>Section Lines* - The Boundary and Centre-line need to be defined before calling this option. In addition to the Figure menu there is a Sections menu, with the following options:

*Sections > Set*: uses the points on the centre-line to define a set of cross-sections. The user is prompted to input the length of the line either side of the centre-line and the sections are then generated;

*Sections > Clip*: clip the cross-section lines to the defined boundary;

*Sections > Add*: add one or more points to the centre-line (either by inserting points, or as an extension of a line). This adds sections at the new centre-line points;

*Sections > Edit*: adjust the end points of existing sections;

*Sections > Delete*: remove one or more sections;

*Sections > Reset*: revert to the initial centre-line defined sections;

*Sections > Label*: add numbered labels to the sections.

NOTE: Clip shortens the lines based on the intersection with the boundary. If the section does not cross the boundary within the specified section length, the section is excluded. Re-running Clip after some editing of the sections can result in more lines being omitted (because the lines do not cross). It is therefore better to do the edits, and then Set the lines before applying the Clip tool.

Using the 3 sets of lines, the cross-sections can be extracted and the channel connectivity can be defined:

*Setup>Sections>Sections* - extract the depths along a set of cross-sections. Section lines need to have been defined before using this option.

*Setup>Sections>Channel Links* - define the connections between the different reaches of the channel network. The Centreline needs to be defined before using this option.



There are then options to view the various types of line work:

*Setup>Sections>View Sections>Layout* - creates a plot of the boundary, centre-line and section lines.

*Setup>Sections>View Sections>Sections* - creates a composite plot of all the sections (the individual sections can be queried by clicking the mouse on a section line).

*Setup>Sections>View Sections>Network* - creates a graph plot of the network with the nodes labelled with the index of the point in the centre-line line set and the distance from the first point in parenthesis, with the line number defined on the edges between the nodes.

### *Setup>Waterbody*

Similar to the boundary tool used for creating Sections, the Waterbody tools enable a bounding polygon to be defined for use when computing the surface area hypsometry (*Setup>Add Hypsometry from Grid>Surface area*). As with Sections there are options to

*Setup>Waterbody >Generate* - use the built in tools to aid the creation of the lines. This option can be used to modify existing linework, create new linework, or copy the *Sections > Boundary* tool linework.

*Setup>Waterbody >Load* - load a shapefile with the lines to be used.

*Setup>Waterbody >Edit* - interactively digitise edit the waterbody linework.

*Setup>Waterbody>Delete* - deletes the saved lines for the selected Case.

*Setup>Waterbody>View* - creates a plot of the boundary.

## 3.4.6 Other Setup options

*Setup>Input Parameters*: enter and edit the specified model parameters. The input parameters can be viewed on the *Inputs* tab.

*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 Tools

Tools has three options: Models, Derive Output and User Tools.

### 3.5.1 Hypsometry and Gross Properties

*Tools >Hypsometry*: use a loaded bathymetry data set to construct surface area, or along-channel width hypsometry datasets. These options replicate the data formats that can be imported using the Import Spatial Data options.

*Tools >Hypsometry >Surface area*: derive the surface area hypsometry from an imported bathymetry (to save space import the grid, run the utility and then delete the grid).

*Tools >Hypsometry >Width*: derive the along-channel width hypsometry from an imported bathymetry (to save space import the grid, run the utility and then delete the grid).

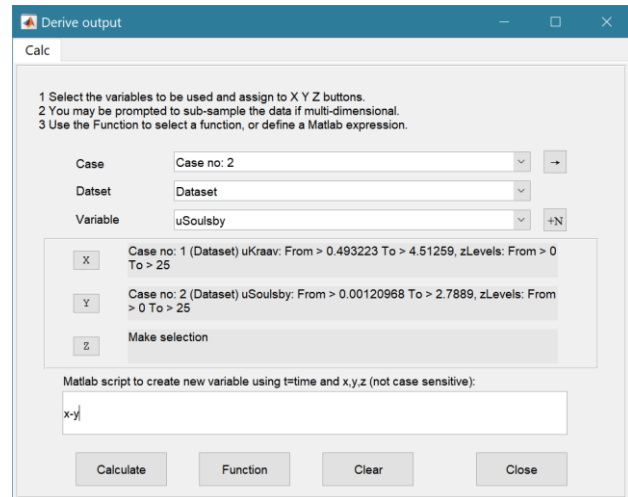
*Tools >Gross Properties*: the large-scale, or gross, morphological properties can be added as a table once, Tidal Levels and a Surface area or Width hypsometry have been defined. Add a row of data to the morphological properties table by selecting a hypsometry and a tidal data set to use.

### 3.5.2 Derived Output

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

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

An equation is then defined in the text box below using the x, y, z or t variables<sup>5</sup>. Based on the user selection the routine applies the defined variable ranges to derive a new variable. In addition text inputs required by the call and the model object (mobj) can also be passed.



Adding the comment %time or %rows, allows the the row dimension to be added to the new dataset. For example if x and y data sets are timeseries, then a Matlab<sup>TM</sup> expresion, 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 (numeroe 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);
```

<sup>5</sup> Various pre-defined function templates can be accessed using the 'Function' button. Alternatively, text can be pasted into the equation box from the clipboard by right clicking in the text box with the mouse.

```
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 4.2.

### 3.5.3 User Tools

*Tools> User Tools*: calls function 'edb\_user\_tools.m', which includes a function to:

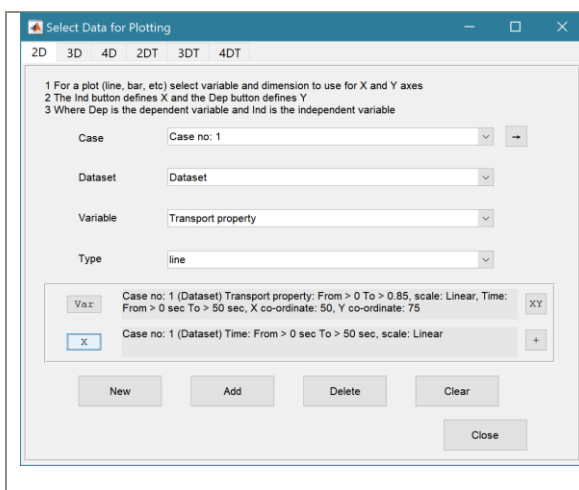
- (i) create a figure tabulating a dataset (tabular data only);
- (ii) add additional properties to a table and (tabular estuary properties data only);
- (iii) run and save the results of a convergence analysis (e.g. for along channel properties);
- (iv) the option for the user to add functions as required.

## 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:

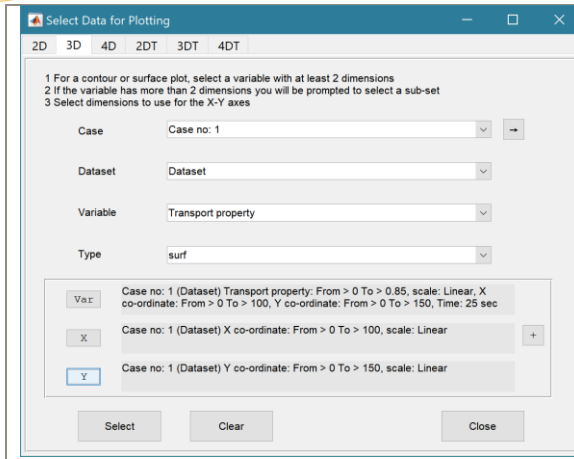
→ : updates the list of Cases

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).

### Selection of 'User' plot type in Plotting UI

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.

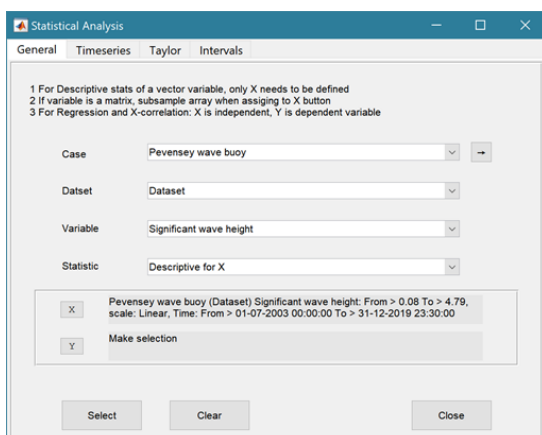
## 3.6.2 Statistics

**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

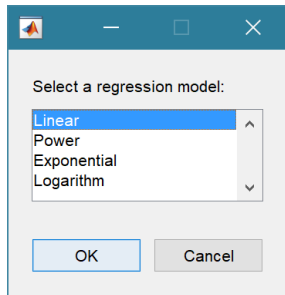
The General tab allows the user to apply the following statistics to data loaded in ModelUI:

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



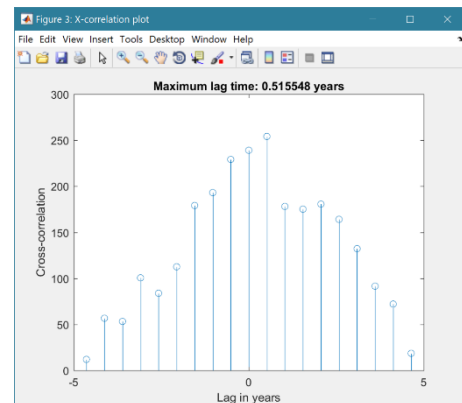
sum and linear regression fit parameters). Only X needs to be defined. The range of the variable can be adjusted when it is assigned to the X button (see Section 3.9). If the variable being used is a multi-dimensional matrix (>2D), the user is prompted to define the range or each additional dimension,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

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.

### 3.6.3 Bespoke Plots

*Analysis > Tabular Plots:* Bespoke plots for scalar tabular data.

- (i) plot a scatter diagram of two or three variables from any case (selected variables must be the same length);
- (ii) plot a bar chart of a variable with the bars coloured based on a selected classification variable (from the same dataset as the main variable);
- (iii) plot a range plot by selecting low-mean-high values of two properties (eg tidal ranges and river discharge);
- (iv) plot a Geyer-McCready plot (Figure 6 in 2014 paper) using the gross properties of the estuaries

*Analysis > Hypsometry Plots:* Bespoke plots for the surface area and width hypsometry data sets.

- (i) convergence plot has panels for along-channel variation of width, csa and depth and a fourth panel for an image of where the sections were taken (if available in the dataset).

*Analysis > User Plots:* calls function 'edb\_user\_plots.m', allowing additional plots to be added as required.

## 3.7 Help

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

### 3.8 Tabs

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

**Data** and **Models**: lists the data loaded and 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, source information (files input or models used) and any user data (e.g., tables of derived parameters) that is saved with the data set.

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

**Table**: tabulates a selected dataset (display only) and has the following sub-Tabs:

**Dataset**: tabulates Case datasets that are scalar, vector or matrices.

**Tides**: tabulates any tidal elevation data added for the specific estuary.

**Rivers**: tabulates any river discharge data added for the specific estuary.

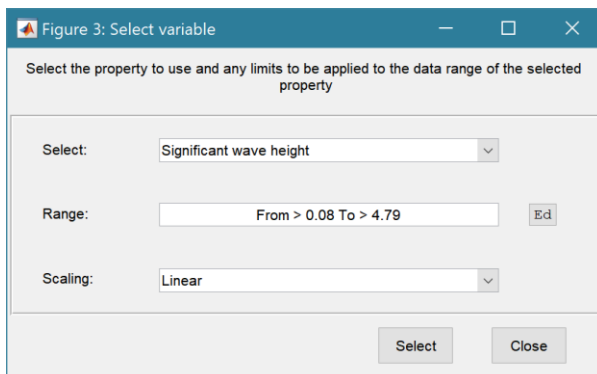
**Morphology**: tabulates the gross properties for the estuary derived from either the surface area hypsometry or the width hypsometry and selected tidal range.

**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 (**Error! Reference source not found.**), 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. Once 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:



+ scaling options include Linear; Log; Relative ( $V - V(x=0)$ ); Scaled ( $V/V(x=0)$ ); Normalised; Normalised ( $=ve$ ); Differences; Rolling mean.

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.

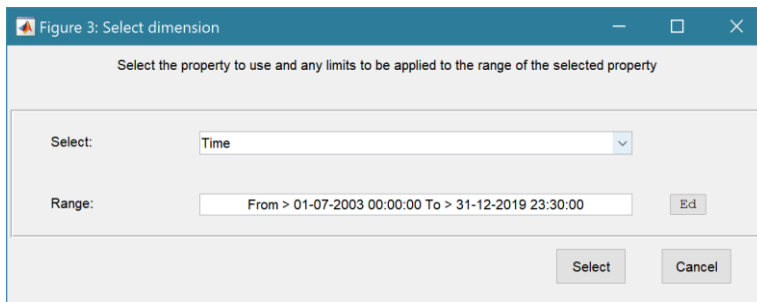
#### Step 1.

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.<sup>+</sup>

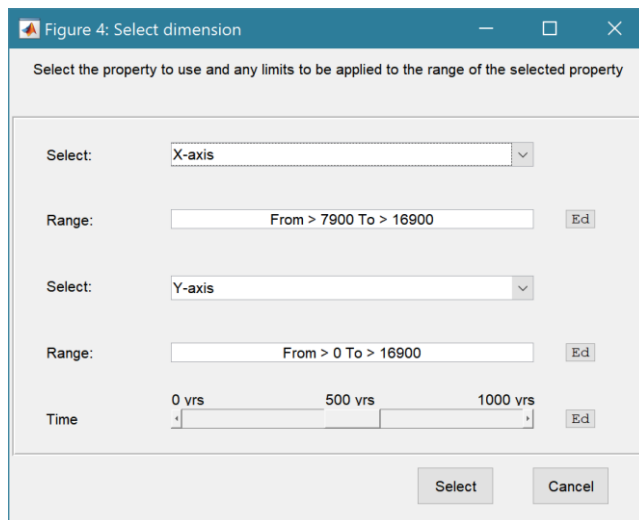
Press Select to go to the next step or Close to quit.



**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 1<sup>st</sup> variable to use as a dimension.

Edit range if required.

Select the 2<sup>nd</sup> variable to use as a dimension.

Edit range if required.

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

Press Select to accept the selection made.

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

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

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

The UI treats any form of text data (char, string, cellstr, categorical, etc) as a list. To do this any variable of these data types uses categories derived by making the list categorical. The range is then based on the order of the categories, selecting all values between the selected end members. To change the order, modify the data type to categorical and order the categories (see *Project>Cases>Modify Variable Type* in Section **Error! Reference source not found.**).

### 3.10 Accessing the tables

Estuary specific data sets – Grid, Surface Area, Width and Images – are all saved as part of the estuary Case record (*mobj.Cases.DataSets.EDBimport.Data.( $\langle$ datasetname $\rangle$ )*, where *mobj* is an instance of the EstuaryDB). The surface area boundary is held in the class *Waterbody* property (*obj.Waterbody*, where *obj* is the Case instance of the EDBimport class). The cross-sections are held in the class *Sections* property, tides and river discharges are held in the *HydroProps* property as a struct with fields *TidalLevels* and *RiverDischarges*, and the morphological properties is held in the *MorphProps* property. All data are held in dstable with defined DSproperties.

### 3.1.1 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:

```
>> myapp.Cases.Catalogue
```

To access imported or model data sets, use:

```
>> myapp.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:

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

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

where *idx* is the index of the class instance (e.g. if there are multiple model runs need to select which one. The Catalogue defines the case record (*caserec* – current position in the catalogue) and case index (*caseid* – a unique case index). [Note: the muiCatalogue class functions *caseRec* and *caseID* allow one to be obtained from the other. The *caseid* is saved in the Property *CaseIndex* for classes that use the abstract class muiDataSet. The muiCatalogue method *classRec* can be used to obtain the classrec index i.e., *classrec* = *classRec*(*muicat*, *caserec*); where *muicat* is the myapp.Cases instance of the muiCatalogue].

To access the underlying *table*, use:



```
>> myapp.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 Supporting Information

### 4.1 Loading spatial data

There are currently options to load vector data and images (see Section 3.4.2). The format for loading surface area and width hypsometry is defined in a data specific format file (see Appendix D – Bespoke data import format file). The formats already provided for can be found in the folder `.../edb_format_files`. Examples of the data for the defined surface area and width formats are given in Appendix D.

The data input for tidal levels and river discharges use similar data formats for the data and the DSproperties. Some examples are provided in Appendix E – Estuary Property Example Formats.

### 4.2 Derive Output

The *Tools > 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 **Error! Reference source not found.**). 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 4.2.2).

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 4.2.2 and **YYY**).

*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™ expresion, or function call, can be used to create a new time series as follows:

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

#### 4.2.1 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 EstuaryDB are detailed in Section 4.2.3.

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 4.2.2). If the variables have a dimension (e.g., *time*) then this should be the first variable, with other variables following. If there is a need to handle additional dimensions then use the option to return a dstable.

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

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

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

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

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

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

#### 4.2.2 Input and output format for external functions

There are several possible use cases:



#### 4.2.2.1 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
```

#### 4.2.2.2 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
```

#### 4.2.2.3 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
```

#### 4.2.2.4 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 dsproperties 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
```

#### 4.2.2.5 Saving additional model parameters

When saving function results as dstable, it is also possible to save additional parameters as part of the table. The following example puts a table of summary statistics in the dstable UserData property.

```
function dst = tidalrange(wl,t,issave,isplot)
    ....
    dsp = setDSproperties();
    results = {R,hwl,lwl};
    dst = dstable(results{:},'RowNames',rt,'DSproperties',dsp);
    %Put fit parameters in UserData
    dst.UserData = summary_stats_table;
    ...
end
```

## Adding functions to the Function library

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

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

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

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

### 4.2.3 Pre-defined functions

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

Some useful examples primarily for timeseries data include :

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

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

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

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

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

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

4. **Interpolate and add noise.** To infill a record with additional points and, if required, add some random noise to the interpolated values. This is called using:

`interpwithnoise(x, t, npad, scale, method, ispos)` , where `X` is the variable, `t` is

time, npad is the number of points to add between the existing data points, scale determines the magnitude of the random noise (a value of 0 results in an interpolated record with no noise), method is the Matlab algorithm used for the interpolation (the default is linear) and ispos is a true/false flag which sets negative values to zero if true.

5. **Subsample one record at the time intervals of another record** (e.g. subsample water levels to be at the same intervals as the wave data). Function is:

`subsample_ts(X, t, mobj)`, where X and t are the variable to be subsampled and *mobj* is the UI handle (must be *mobj*). The user is prompted to select the dataset to be used to define the time intervals. A time series is returned and added as a Derived data set. The user is prompted to define the metadata for the new data set.

6. **Subsample one record based on a threshold defined for another record** (e.g. subsample waves based on a threshold water level). Function is:

`subsample(X, t, thr, mobj)`, where X and t are the variable to be subsampled, *thr* is the threshold value and *mobj* is the UI handle (must be *mobj*). The user is prompted to select the dataset and variable to be used to define the condition and a condition operator ( $\leq$ ,  $=$ , etc). A time series is returned and added as a Derived data set. The user is prompted to define the metadata for the new data set.

7. **Phase plot**. This function is similar to the recursive plot function but generates a plot based on two variables that can, optionally, be functions of time. The call to this function is:

`phaseplot(X, Y, t)`, where X and Y are the variables assigned to the respective buttons and t is time (this does not need to be assigned to a button and t can be omitted if a time stamp for the datapoints is not required).

8. **Recursive plot**. Generates a plot of a variable plotted against itself with an offset (e.g.  $x(i)$  versus  $x(i+1)$ ). This is called from the Derive Output GUI using:

`recursive_plot(x, 'varname', nint)`, where x is the variable, 'varname' is a text string in single quotes and *nint* is an integer value that defines the size of the offset.

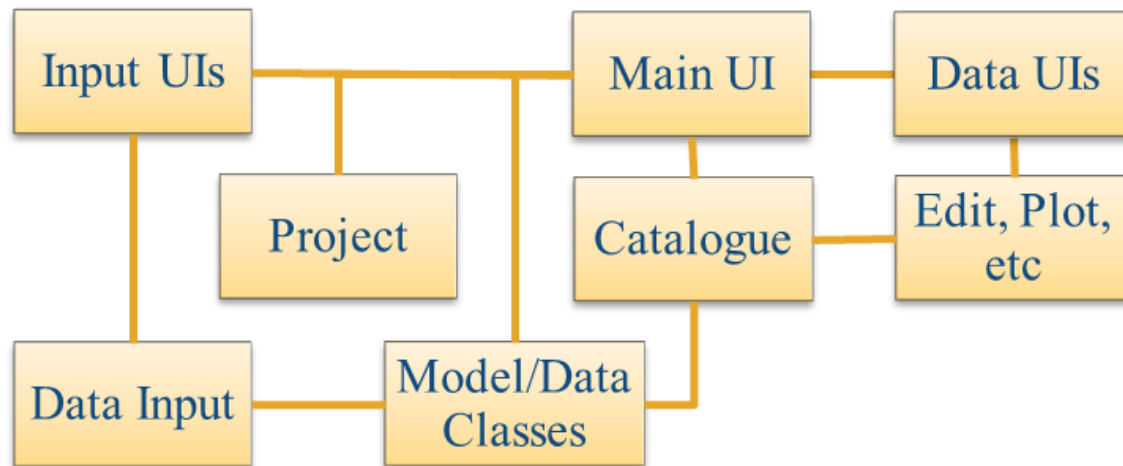
9. **Selection of frequency analysis plots of timeseries data** using the function:

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

## 5 Program Structure

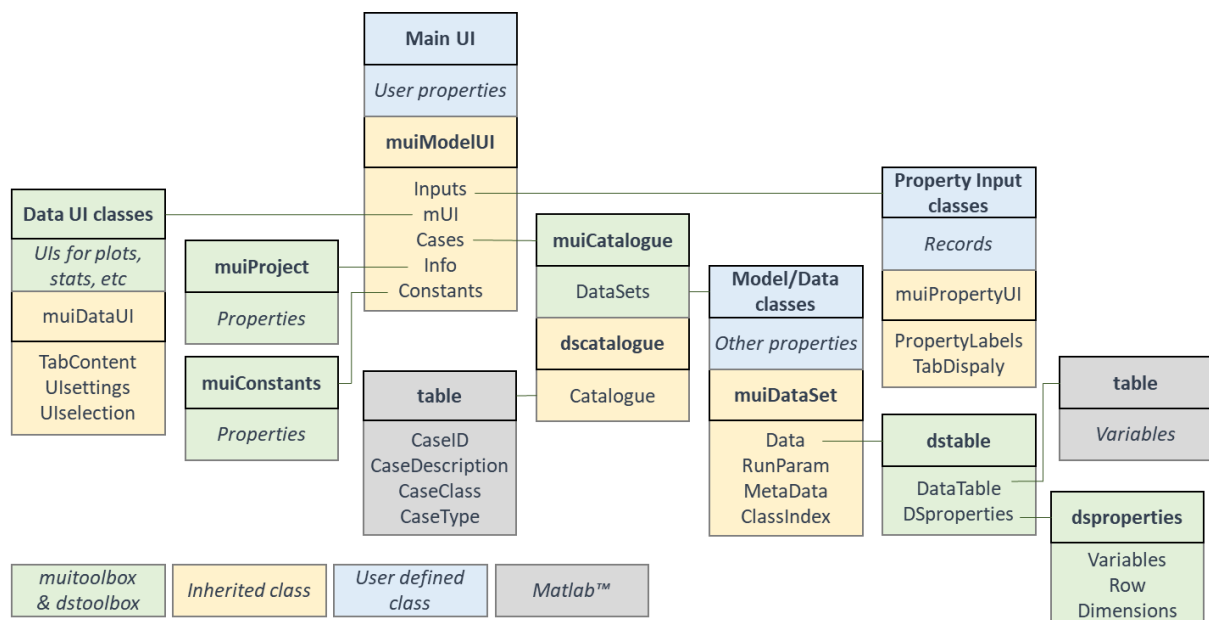
The overall structure of the code is illustrated schematically in Figure 1. This is implemented through several classes that handle the graphical user interface and program workflows (Main UI) and several classes that handle the data manipulation and plotting (Input UIs and Data UIs).

Figure 1 – High level schematic of program structure



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

**EstuaryDB** – class to define the behaviour of the main UI.

**EDBimport** – class to import and provide data specific methods for spatial data (vector and matrix)

**EDBparameters** – class to input parameters. This is a template for the user to define any inputs required for bespoke data analysis functions. The class file has to be edited to define the required inputs.

**EDB\_Probe** – handles additional analysis of the data

**EDB\_ProbeUI** – user interface for additional analysis of the data

*curvespace* - evenly spaced points along an existing curve in 2D or 3D (from Matlab(TM) Forum; Yo Fukushima, <https://www.mathworks.com/matlabcentral/fileexchange/7233-curvedspace>).

*edb\_derived\_props* – functions to derive datasets related to estuary gross properties such as hydraulic depths, prism etc.

*edb\_hypsometry\_plots* – plots of surface area and width hypsometry and channel convergence.

*edb\_plot\_tidelevels* - add the high, mean and low water tide levels to a plot.

*edb\_props\_table* - use surface area hypsometry or width hypsometry to compute the gross properties of an inlet or estuary.

*edb\_regression\_analysis* – regression analysis for selected along-channel variables. Can be plotted or added to database table.

*edb\_regression\_plot* – bespoke plot of the results of a regression analysis showing width, csa, and depth along with image of estuary (if available).

*edb\_surfacearea\_table* - compile the surface area hypsometry dataset.

*edb\_s\_hypsometry* - compute the surface area hypsometry data.

*edb\_table\_plots* – functions for scalar tabular data to plot a scatter diagram of two variables from any case (selected variables must be the same length) and a bar chart of a variable with the bars coloured based on a selected classification variable (from the same dataset as the main variable).

*edb\_user\_plots* – provision for user to add own plotting options.

*edb\_user\_tools* – includes a function to create a figure tabulating a dataset and the option for the user to add functions as required.

*edb\_width\_table* - compile the width hypsometry dataset.

*edb\_w\_hypsometry* - compute the width hypsometry data.

*geyer\_mccready\_plot* – generate a figure that matches Figure 6 in Geyer W R and MacCready P, 2014, The Estuarine Circulation. Annual Review of Fluid Mechanics, 46 (1), 175-197.

Data import format files include:

*edb\_bathy\_format* - defines format for import of xyz bathymetry data.

*edb\_image\_format* - defines format for import of images (eg tiff or jpg).

*edb\_s\_hyps\_format* - defines format for import of surface area hypsometry data ([z,S]).

*edb\_w\_hyps\_format* - defines format for import of along-channel width hypsometry data ([x,z,W]).

*edb\_zm\_data\_format* – format file to load data set of along-channel properties prepared by Dr Zhang Min using SeaZone bathymetry for a selection of UK estuaries.

Additional functions include:

...\muiAppLib\muiAppEstuaryFcns - library of functions for commonly used methods such as sea level rise, simple tide simulation, etc.

...\muiAppLib\muiAppGridFcns - library of functions for a range of grid, point and line manipulation methods.

Matlab™ Mapping toolbox, or M-Map1.4 (<https://www-old.eoas.ubc.ca/~rich/map.html>) are used to read shapefiles. As a faster alternative to Matlab™ *inpolygon* function, the *InsidePoly* functions can be used (<https://uk.mathworks.com/matlabcentral/fileexchange/27840-2d-polygon-interior-detection>).



Other classes and functions are used to provide grid tools and interactive point and line tools. These are detailed in the online documentation.

## 6 Bibliography

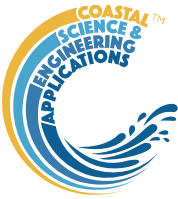
- Abramov V and Khan M K, 2017, A Practical Guide to Market Risk Model Validations (Part II - VaR Estimation). p. 70, <http://dx.doi.org/10.2139/ssrn.3080557>.
- Antoniades I P, Brandi G, Magafas L and Di Matteo T, 2021, The use of scaling properties to detect relevant changes in financial time series: A new visual warning tool. *Physica A: Statistical Mechanics and its Applications*, 565, 10.1016/j.physa.2020.125561.
- Brandi G and Di Matteo T, 2021, On the statistics of scaling exponents and the multiscaling value at risk. *The European Journal of Finance*, pp. 1-22, 10.1080/1351847x.2021.1908391.
- Coles S, 2001, *An Introduction to Statistical Modeling of Extreme Values*, Springer-Verlag, London.
- Davidson N C, Laffoley D d A, Doody J P, Way L S, Gordon J, Key R, Drake C M, Pienkowski M W, Mitchell R and Duff K L, 1991, *Nature conservation and estuaries in Great Britain*, Nature Conservancy Council, Peterborough, UK.
- Di Matteo T, Aste T and Dacorogna M M, 2003, Scaling behaviors in differently developed markets. *Physica A: Statistical Mechanics and its Applications*, 324 (1-2), pp. 183-188, 10.1016/s0378-4371(02)01996-9.
- Geyer W R and MacCready P, 2014, The Estuarine Circulation. *Annual Review of Fluid Mechanics*, 46 (1), pp. 175-197, 10.1146/annurev-fluid-010313-141302.
- Hansen D V and Rattray M, 1966, New dimensions in estuary classification. *Limnology and Oceanography*, 11 (3), pp. 319-326.
- Hume T M and Herdendorf C E, 1988, A geomorphic classification of estuaries and its application to coastal resource management. *Journal of Ocean and Shoreline Management*, 11, pp. 249-274.
- Hume T M, Snelder T, Weatherhead M and Liefing R, 2007, A controlling factor approach to estuary classification. *Ocean & Coastal Management*, 50 (11-12), pp. 905-929, 10.1016/j.ocecoaman.2007.05.009.
- Ihlen E A, 2012, Introduction to multifractal detrended fluctuation analysis in matlab. *Front Physiol*, 3, p. 141, 10.3389/fphys.2012.00141.
- Morales R, Di Matteo T, Gramatica R and Aste T, 2012, Dynamical generalized Hurst exponent as a tool to monitor unstable periods in financial time series. *Physica A: Statistical Mechanics and its Applications*, 391 (11), pp. 3180-3189, <https://doi.org/10.1016/j.physa.2012.01.004>.
- Pacheco J C R, Román D T and Vargas L E, 2008, R/S Statistic: Accuracy and Implementations, 18th International Conference on Electronics, Communications and Computers (conielecomp 2008), IEEE, 10.1109/conielecomp.2008.14.
- Pritchard D W, 1989, Estuarine classification-a help or hindrance, In: *Estuarine Circulation*, Series, Neilson B, Kuo A and Brubaker J (eds), pp. 1-38, Humana Press, New Jersey.
- Sutcliffe J, Hurst S, Awadallah A G, Brown E and Hamed K, 2016, Harold Edwin Hurst: the Nile and Egypt, past and future. *Hydrological Sciences Journal*, 61 (9), pp. 1557-1570, 10.1080/02626667.2015.1019508.
- Taylor K E, 2001, Summarizing multiple aspects of model performance in a single diagram. *Journal of Geophysical Research - Atmospheres*, 106 (D7), pp. 7183-7192, 10.1029/2000JD900719.





Townend I H, 2004, Identifying change in estuaries. *Journal of Coastal Conservation*, 10, pp. 5-12.

Townend I H, 2005, An examination of empirical stability relationships for UK estuaries. *Journal of Coastal Research*, 21 (5), pp. 1042-1053.



## Appendix A - Import file formats

File formats for Load and Add tabular data

ASCII text file:

	Name	id	JNCC	ERP2_Type	Type_id	Range	V mhw	V mtl	V mlw	S mhw	S mtl	S mlw	A mhw	A mtl	A mlw
1	Thames Estuary	117	Coastal Plain	Funnel	5	4.5	9.67E+08	6.11E+08	4.17E+08	2.00E+08	8.19E+07	6.49E+07	5.81E+04	3.85E+04	2.31E+04
2	Medway Estuary	119	Coastal Plain	Spit Enclosed	4	4.1	2.44E+08	1.36E+08	7.96E+07	7.56E+07	2.75E+07	2.17E+07	2.25E+04	1.84E+04	1.52E+04
3	Pegwell Bay	121	Embayment	Spit Enclosed	4	4.5	1.61E+07	2.92E+06	6.03E+05	9.53E+06	2.50E+06	5.26E+05	2.35E+03	9.00E+02	2.84E+02
4	Chichester Harbour	128	Bar Built Estuary	Tidal inlet	7	4.2	9.50E+07	5.14E+07	2.35E+07	3.41E+07	1.66E+07	1.21E+07	8.00E+03	5.45E+03	3.46E+03
5	Langstone Harbour	129	Bar Built Estuary	Tidal inlet	7	4.2	6.19E+07	3.25E+07	1.36E+07	2.68E+07	1.10E+07	8.36E+06	4.70E+03	3.74E+03	2.93E+03
6	Portsmouth Harbour	130	Bar Built Estuary	Tidal inlet	7	4.1	7.21E+07	4.79E+07	2.63E+07	1.95E+07	1.15E+07	9.07E+06	6.23E+03	4.79E+03	3.28E+03
7	Southampton Water	131	Coastal Plain	Spit Enclosed	4	4	1.89E+08	1.33E+08	7.83E+07	6.28E+07	2.67E+07	2.00E+07	1.97E+04	1.55E+04	1.02E+04
8	Beaulieu River	132	Bar Built Estuary	Spit Enclosed	4	3.2	1.38E+07	7.16E+06	3.47E+06	8.41E+06	2.60E+06	1.84E+06	6.43E+03	4.34E+03	1.90E+03
9	Lymington Estuary	133	Coastal Plain	Spit Enclosed	4	2.5	2.45E+06	1.41E+06	6.00E+05	2.38E+06	7.44E+05	4.86E+05	2.88E+03	1.79E+03	6.99E+02
10	Bosham Harbour	134	Coastal Plain	Spit Enclosed	4	3.1	5.55E+06	1.87E+06	5.69E+05	2.19E+06	1.66E+06	3.83E+05	5.27E+03	3.26E+03	1.42E+03
11	Wootton Creek & Ryde Sands	135	Coastal Plain	Spit Enclosed	4	3.8	6.27E+05	2.14E+05	5.09E+04	9.56E+05	1.12E+05	6.33E+04	1.01E+03	4.88E+02	1.17E+02
12	Medina Estuary	136	Coastal Plain	Funnel	5	4.2	4.72E+06	2.94E+06	1.42E+06	2.43E+06	9.45E+05	7.29E+05	2.62E+03	1.81E+03	9.18E+02
13	Newtown Estuary	137	Bar Built Estuary	Spit Enclosed	4	2.9	6.01E+06	4.32E+06	2.79E+06	1.57E+06	1.11E+06	8.01E+05	1.88E+03	1.14E+03	4.37E+02
14	Yar Estuary	138	Coastal Plain	Spit Enclosed	4	2.5	5.35E+05	2.81E+05	9.53E+04	3.66E+05	1.70E+05	1.11E+05	1.41E+03	8.38E+02	2.47E+02
15	Christchurch Harbour	139	Bar Built Estuary	Spit Enclosed	4	1.2	1.95E+06	1.04E+06	5.43E+05	4.62E+06	9.58E+05	7.90E+05	1.18E+02	4.17E+01	1.70E+01
16	Poole Harbour	140	Bar Built Estuary	Spit Enclosed	4	1.4	5.65E+07	4.32E+07	2.20E+07	2.94E+07	2.44E+07	1.82E+07	3.27E+03	3.05E+03	2.64E+03
17	The Fleet & Portland Harbour	141	Bar Built Estuary	Tidal inlet	7	1.9	3.85E+07	2.22E+07	1.66E+07	1.84E+07	1.06E+07	4.84E+06	3.32E+04	2.95E+04	2.66E+04
18	Axe Estuary	142	Bar Built Estuary	Spit Enclosed	4	3.7	1.11E+05	9.53E+03	9.72E+01	1.79E+05	2.19E+04	5.21E+02	2.58E+02	7.70E+01	4.40E+00
19	Otter Estuary	143	Bar Built Estuary	Spit Enclosed	4	4.1	7.47E+04	2.82E+04	5.56E+03	4.20E+04	2.24E+04	9.62E+03	1.78E+02	3.95E+01	2.00E+01
20	Exe Estuary	144	Bar Built Estuary	Spit Enclosed	4	4.1	4.69E+07	2.25E+07	5.23E+06	1.81E+07	1.06E+07	7.46E+06	2.81E+03	1.67E+03	1.06E+03
21	Teign Estuary	145	Ria	Spit Enclosed	4	4.2	5.13E+06	2.48E+06	4.69E+06	1.91E+06	1.03E+06	1.46E+06	5.34E+02	3.31E+02	4.52E+02
22	Dart Estuary	146	Ria	Ria	3	4	4.90E+07	3.47E+07	2.00E+07	1.02E+07	6.70E+06	5.74E+06	9.99E+03	8.24E+03	6.17E+03
23	Salcombe & Kingsbridge Estuary	147	Ria	Ria	3	4.6	1.60E+07	6.85E+06	1.91E+06	1.20E+07	2.77E+06	1.45E+06	1.10E+04	8.21E+03	5.31E+03

Excel spreadsheet:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name	id	JNCC	ERP2_Type	Type_id	Range	V mhw	V mtl	V mlw	S mhw	S mtl	S mlw	A mhw	A mtl	A mlw
2	Hayle Estuary	1	Bar Built Estuary	Spit Enclosed	4	5	3.87E+06	2.73E+06	2.32E+06	1.49E+06	1.50E+05	1.44E+05	6.60E+00	0.00E+00	0.00E+00
3	Gannell Estuary	2	Ria	Ria	3	6.4	3.09E+06	1.59E+06	7.20E+05	1.19E+06	3.16E+05	2.55E+05	7.75E+03	5.62E+03	3.67E+03
4	Camel Estuary	3	Ria	Ria	3	5.9	3.61E+07	1.87E+07	8.23E+06	7.77E+06	4.10E+06	1.81E+06	1.08E+04	6.34E+03	3.02E+03
5	Bridgwater Bay	6	Embayment	Spit Enclosed	4	11.1	3.21E+07	1.01E+07	2.64E+05	7.44E+06	2.94E+06	7.48E+05	7.31E+03	3.48E+03	3.44E+02
6	Savern Estuary	7	Coastal Plain	Funnel	5	12.3	7.14E+09	4.19E+09	2.00E+09	6.17E+08	4.42E+08	3.63E+08	2.81E+05	2.06E+05	1.36E+05
7															
8															
9															

DS

## Appendix B - Data set properties (DSproperties)

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

DSproperties for a set of timeseries variables.

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

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

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

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

DSproperties for a 2D variable with 2 spatial dimensions.

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

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

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

## Appendix C – Sample DSproperties for data import

Text file of variable definitions of DSproperties:

```
1 name→description→unit→label→qcflag/format
2 id→Index→-→Index→none
3 JNCC_Type→JNCC_Type→-→Estuary_type→none
4 ERP2_Type→ERP2_Type→-→Estuary_type→none
5 Type_id→Type.index→-→Estuary_type→none
6 Range→Tidal.range→m→Range→raw
7 Vmhw→Volume.at.Mean.High.Water→m^3→Volume→model
8 Vmtl→Volume.at.Mean.Tidel.Level→m^3→Volume→model
9 Vmlw→Volume.at.Mean.Low.Water→m^3→Volume→model
10 Smhw→Surface.area.at.Mean.High.Water→m^2→Surface.area→model
11 Smtl→Surface.area.at.Mean.Tide.Level→m^2→Surface.area→model
12 Smlw→Surface.area.at.Mean.Low.Water→m^2→Surface.area→model
13 Amhw→Cross-section.area.at.Mean.High.Water→m^2→Cross-section.area→model
14 Amtl→Cross-section.area.at.Mean.Tide.Level→m^2→Cross-section.area→model
15 Amlw→Cross-section.area.at.Mean.Low.Water→m^2→Cross-section.area→model
```

Matlab function with definition of all DSproperties:

```
function dsp = sample_dsproperties()
%sample function to define the dsproperties for a data set
%define a dsproperties struct and add the model metadata
dsp = struct('Variables',[],'Row',[],'Dimensions',[]);
%define each variable to be included in the data table and any
%information about the dimensions. dstable Row and Dimensions can
%accept most data types but the values in each vector must be unique

%struct entries are cell arrays and can be column or row vectors
dsp.Variables = struct(...
    'Name',{'id','JNCC_Type','ERP2_Type','Type_id','Range','Vmhw',...
        'Vmtl','Vmlw','Smhw','Smtl','Smlw','Amhw','Amtl','Amlw'},...
    'Description',{'Index','JNCC Type','ERP2 Type',...
        'Type index','Tidal range',...
        'Volume at Mean High Water',...
        'Volume at Mean Tidel Level',...
        'Volume at Mean Low Water',...
        'Surface area at Mean High Water',...
        'Surface area at Mean Tide Level',...
        'Surface area at Mean Low Water',...
        'Cross-sectional area at Mean High Water',...
        'Cross-sectional area at Mean Tide Level',...
        'Cross-sectional area at Mean Low Water'},...
    'Unit',{'-','-','-','-','m',...
        'm','m','m^3','m^3','m^2','m^2','m^2','m^2','m^2'},...
    'Label',{'Index','Estuary type','Estuary type',...
        'Estuary type','Range','Volume','Volume','Volume',...
        'Surface area','Surface area','Surface area',...
        'Cross-sectional area','Cross-sectional area','Cross-sectional area'},...
    'QCflag',repmat({'data'},1,14));
dsp.Row = struct(...
    'Name',{'Location'},...
    'Description',{'Location'},...
    'Unit',{'-'},...
    'Label',{'Location'},...
    'Format',{''});
dsp.Dimensions = struct(...
    'Name',{''},...
    'Description',{''},...
    'Unit',{'-'},...
    'Label',{''},...
    'Format',{''});
end
```



Excel file with definition of all DSproperties:

	A	B	C	D	E	F
1	Name	Description	Unit	Label	QCflag	
2	id	Index	-	Index	none	
3	JNCC_Type	JNCC Type	-	Estuary type	none	
4	ERP2_Type	ERP2 Type	-	Estuary type	none	
5	Type_id	Type index	-	Estuary type	none	
6	Range	Tidal range	m	Range	raw	
7	Vmhw	Volume at Mean High Water	m <sup>3</sup>	Volume	model	
8	Vmtl	Volume at Mean Tide Level	m <sup>3</sup>	Volume	model	
9	Vmlw	Volume at Mean Low Water	m <sup>3</sup>	Volume	model	
10	Smhw	Surface area at Mean High Water	m <sup>2</sup>	Surface area	model	
11	Smtl	Surface area at Mean Tide Level	m <sup>2</sup>	Surface area	model	
12	Smlw	Surface area at Mean Low Water	m <sup>2</sup>	Surface area	model	
13	Amhw	Cross-section area at Mean High Water	m <sup>2</sup>	Cross-section area	model	
14	Amtl	Cross-section area at Mean Tide Level	m <sup>2</sup>	Cross-section area	model	
15	Amlw	Cross-section area at Mean Low Water	m <sup>2</sup>	Cross-section area	model	
16						
17						

## Appendix D – Bespoke data import format file

The format file for a new data set can be edited from the template file provided in the `..\mui toolbox\toolbox\mui templates` folder.

```
function output = ***_data_format(funcall,varargin) % <<Edit to identify data type
%
%-----function help-----
% NAME
% ***_data_format.m % <<Edit to identify data type
% PURPOSE
% Functions to define metadata, read and load data from file for:
% Zhang Min's estuary cross-section data
% USAGE
% output = ***_data_format(funcall,varargin) % <<Edit to identify data type
% INPUTS
% funcall - function being called
% varargin - function specific input (filename,class instance,dsp,src, etc)
% OUTPUT
% output - function specific output
% NOTES
% ZM analysed UK estuaries using the SEAZONE bathymetry. This file loads
% the along-channel data
%
% Author: Ian Townend
% CoastalSEA (c) Oct 2024
%-----
%
switch funcall
    %standard calls from muiDataSet - do not change if data class
    %inherits from muiDataSet. The function getPlot is called from the
    %Abstract method tabPlot. The class definition can use tabDefaultPlot
    %define a plot function in the class file, or call getPlot
    case 'getFormat'
        output = getFormat(varargin{:});
    case 'getData'
        output = getData(varargin{:});
    case 'dataQC'
        output = dataQC(varargin{Li});
    case 'getPlot'
        %output = 0; if using the default tab plot in muiDataSet, else
        output = getPlot(varargin{:});
end
end
```

There are then 4 functions defined to as defined above.

Data format:

```
%-----
% getFormat
%-----
function obj = getFormat(obj,formatfile)
    %return the file import format settings
    obj.DataFormats = {'muiUserData',formatfile};
    obj.idFormat = 1;
    obj.FileSpec = {'on','*.txt; *.csv; *.xlsx;*.xls;'}; % <<Edit to file types
end
```

Loading the data:

```
%-----
```



Loading the data:

```
%-----
% getData
%-----
function newdst = getData(obj,filename)
    %read and load a data set from a file
    dsp = setDSproperties;           %set metadata for the dataset being imported
    % readtable is a Matlab function to read text and spreadsheet tabular data
    % cleandata is a bespoke function to sort and edit the imported dataset,
    % this also uses the filename to define the table Description.
    % The input table is assigned to a struct field that specifies dataset name
    itable = readtable(filename);
    itable = cleandata(itable,filename);
    idx = 1:height(itable);
    dst = dstable(itable,'RowNames',idx,'DSproperties',dsp);
    dst.Description = itable.Properties.Description; % <<assign 'location'
    newdst.ZMdata = dst;           %<< ZMdata is the dataset name for this format
end
```

Defining the DSproperties:

```
%-----
% setDSproperties
%-----
function dsp = setDSproperties()
    %define the variables in the dataset and the metadata properties
    dsp = struct('Variables',[],'Row',[],'Dimensions',[]);
    %define each variable to be included in the data table and any
    %information about the dimensions. dstable Row and Dimensions can
    %accept most data types but the values in each vector must be unique
    %struct entries are cell arrays and can be column or row vectors
    dsp.Variables = struct(...
        'Name',{'hLW','hMT','hHW'},...
        'Description',{'Depth LW','Depth MT','Depth HW'},...
        'Unit',{'m','m','m'},...
        'Label',{'Depth (m)','Depth (m)','Depth (m)'},...
        'QCflag',repmat({'data'},1,3));
    dsp.Row = struct(...
        'Name',{'XIndex'},...
        'Description',{'X-Index'},...
        'Unit',{'-'},...
        'Label',{'X-index'},...
        'Format',{''});
    dsp.Dimensions = struct(...
        'Name',{''},...
        'Description',{''},...
        'Unit',{''},...
        'Label',{''},...
        'Format',{''});
end
```

Define the plot for use on the Q-Plot tab:

To use the class default simply set ok = 0; This will handle timeseries and other vector data. The following code implements a bespoke Q-Plot.

```
%-----
% getPlot
%-----
function ok = getPlot(obj,src,dsetname)
    %generate a plot on the src graphical object handle
    ok = [];
    tabcb = @(src,evdat)tabPlot(obj,src);
    ax = tabfigureplot(obj,src,tabcb,false);

    %-----
    %the following code illustrates the creation of a plot with 3 variables
    %this should be modified to suit the dataset and application
    %-----

    %get data and variable id - select lw value and plot lw,mt,hw
    varoffset = [1,4,7]; %<< dataset specific, empty to access all variables
    [dst,idv,props] = selectVariable(obj,dsetname,varoffset);
    if isempty(idv), return; end
    %create props to define labels for each variable to be plotted
    props = repmat(props,3,1);
    idv = varoffset(idv);
    props(2).desc = dst.VariableDescriptions{idv+1};
    props(3).desc = dst.VariableDescriptions{idv+2};
    idv = idv:idv+2; %<< dataset specific, one or more variables to use
    %test for a vector data set
    if isvector(dst.(dst.VariableNames{idv(1)}))
        idx = find(strcmp(dst.VariableNames,'xCh')); %variable for x-axis

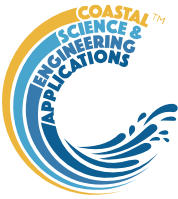
        %use class method to create a vector plot with non-dimensional x and y
        %axes. Pmax is the maximum value of x and each y variable
        pmax = EDBimport.vectorplot(ax,dst,props,idv,idx);

        %add text box with min max range of variable and length
        boxtxt = sprintf('Length: %.0f m\nMax at LW: %.1e, MT: %.1e, HW: %.1e',...
                        pmax.x,pmax.var{:});

        ydist = ax.YLim(1);
        text(0.1,ydist+0.06,boxtxt)
        ok = 1;
    else
        ok = 0; %ok=0 %no plot implemented in getPlot
    end
end
```

Optional Quality Control:

```
%-----
% dataQC
%-----
function output = dataQC(obj) %Quality control can be added if required
    %quality control a dataset
    %to be able to call function the menu for Importing Spatial Data needs to
    %be extended to include a Quality Control option
    % datasetname = getDataSetName(obj); %prompts user to select dataset
    % dst = obj.Data.(datasetname); %selected dstable
    warnDlg('No quality control defined for this format');
    output = []; %if no QC implemented in dataQC
end
```



Sample hypsometry import data set formats:

- (1) Surface area (note the volume data are not loaded). Rows are elevations relative to some datum.

	A	B	C	D	E
1	z	hist	S	V	
2	-40.95	0	0	0	
3	-40.85	0	0	0	
4	-40.75	0	0	0	
5	-40.65	0	0	0	
6	-40.55	0	0	0	
7	-40.45	400	400	40	
8	-40.35	0	400	80	
9	-40.25	0	400	120	
10	-40.15	400	800	200	
11	-40.05	0	800	280	
12	-39.95	0	800	360	
13	-39.85	1200	2000	560	
14	-39.75	400	2400	800	
15	-39.65	800	3200	1120	
16	-39.55	400	3600	1480	
17	-39.45	1200	4800	1960	
18	-39.35	2000	6800	2640	
19	-39.25	800	7600	3400	
20	-39.15	1600	9200	4320	
21	-39.05	3600	12800	5600	
22	-38.95	3200	16000	7200	
23	-38.85	2800	18800	9080	
24	-38.75	2000	20800	11160	
25	-38.65	800	21600	13320	
26	-38.55	800	22400	15560	
27	-38.45	1200	23600	17920	
28	-38.35	2800	26400	20560	
29	-38.25	2000	28400	23400	
30	-38.15	1200	29600	26360	
31	-38.05	400	30000	29360	

- (2) Width. Rows are elevation relative to some datum and columns are distances from the estuary mouth or start of channel.

	A	B	C	D	E	F
1	z	x0	x2000	x4000	x6000	
2	-12.25	5	0	0	0	
3	-12.15	10	0	0	0	
4	-11.25	20	0	0	0	
5	-11.15	50	0	0	0	
6	-10.25	60	0	0	0	
7	-10.15	70	0	0	0	
8	-9.25	80	5	0	0	
9	-9.15	90	10	0	0	
10	-8.25	100	20	0	0	
11	-8.15	110	50	0	0	
12	-7.25	120	60	5	0	
13	-7.15	130	70	10	0	
14	-6.25	140	80	20	0	
15	-6.15	150	90	50	0	
16	-5.25	160	100	60	0	
17	-5.15	170	110	70	0	
18	-4.25	180	120	80	0	
19	-4.15	190	130	90	10	
20	-3.25	200	140	100	20	
21	-3.15	210	150	110	50	
22	-2.25	220	160	120	60	
23	-2.15	230	170	130	70	
24	-1.25	240	180	140	80	
25	-1.15	250	190	150	90	
26	-0.25	260	200	160	100	
27	-0.15	270	210	170	110	
28	0.75	280	220	180	120	
29	0.85	290	230	190	130	
30	1.75	300	240	200	140	
31	1.85	310	250	210	150	
32	2.75	320	260	220	160	
33	2.85	330	270	230	170	
34	3.75	340	280	240	180	

## Appendix E – Estuary Property Example Formats

Tidal levels (variables and definitions can be adjusted to suit)

	A	B	C	D	E	F	G	H	I	J
1	Name	HAT	MHHW	MHW	MLHW	MTL	MHLW	MLW	MLLW	LAT
2	Plymouth Sound	2.50	2.20	2.00	1.60	0.10	-1.60	-2.00	-2.20	-2.50
3	Severn	6.20	5.60	5.00	4.50	-0.10	-4.50	-5.00	-5.60	-6.20
4	Humber	3.00	2.80	2.20	1.80	0.00	-1.80	-2.20	-2.80	-3.00
5										

with DSproperties table:

	A	B	C	D	E
1	Name	Description	Unit	Label	QCflag
2	HAT	Highest astronomical tide	mAD	Tide level (mAD)	data
3	MHHW	Mean high high water	mAD	Tide level (mAD)	data
4	MHW	Mean high water	mAD	Tide level (mAD)	data
5	MLHW	Mean low high water	mAD	Tide level (mAD)	data
6	MTL	Mean tide level	mAD	Tide level (mAD)	data
7	MHLW	Mean high low water	mAD	Tide level (mAD)	data
8	MLW	Mean low water	mAD	Tide level (mAD)	data
9	MLLW	Mean low low water	mAD	Tide level (mAD)	data
10	LAT	Lowest astronomical tide	mAD	Tide level (mAD)	data
11					

River Discharge (variables and definitions can be adjusted to suit)

	A	B	C	D	E	F	G	H	I	J
1	Estuary	Qf_annual_high	Qf_annual_mean	Qf_annual_low	Qf_spring_high	Qf_spring_mean	Qf_spring_low	Qf_summer_high	Qf_summer_mean	Qf_summer_low
2	Humber	209.4	85.6	32.1	194.1	90.0	35.1	94.6	47.6	20.9
3	Plymouth Sound	59.6	22.8	4.4	44.4	18.0	4.6	34.6	7.1	1.3
4	Severn	227.7	77.8	17.0	226.9	72.1	17.5	128.9	33.5	11.0
5										

with DSproperties table:

	A	B	C	D	E
1	Name	Description	Unit	Label	QCflag
2	Qf_annual_high	Annual high river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
3	Qf_annual_mean	Annual mean river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
4	Qf_annual_low	Annual low river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
5	Qf_spring_high	Spring high river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
6	Qf_spring_mean	Spring mean river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
7	Qf_spring_low	Spring low river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
8	Qf_summer_high	Summer high river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
9	Qf_summer_mean	Summer mean river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
10	Qf_summer_low	Summer low river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
11	Qf_autumn_high	Autumn high river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
12	Qf_autumn_mean	Autumn mean river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
13	Qf_autumn_low	Autumn low river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
14	Qf_winter_high	Winter high river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
15	Qf_winter_mean	Winter mean river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
16	Qf_winter_low	Winter low river discharge	m <sup>3</sup> /s	River discharge (m <sup>3</sup> /s)	data
17					
18					