Preface

SedTools is a Matlab^{TM} App for the analysis of settling column data.

Requirements

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

Resources

The SedTools App and two toolboxes (muitoolbox and dstoolbox) can be downloaded from www.coastalsea.uk.

Cite as:

Townend, I.H., 2021, SedTools manual, CoastalSEA, UK, pp18, www.coastalsea.uk.

Bibliography

See source of core program and further details at: http://neumeier.perso.ch/matlab/cal_settling.html

Acknowledgements

Settling analysis is based on the Matlab function cal_settling developed by Urs Neumeier, 2005

Revision history

Version	Date	Changes
2.0	Feb 2021	Ported to work in muitoolbox
1.0	Feb.2019	First release via www.coastalsea.uk

Contents

1   Introduction

SedTools is a Matlab^{TM} App for the analysis of settling column data to derive and plot sediment grain-size distributions.

2   Getting Started

2.1   Configuration

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

dstoolbox: dstoolbox.mltbx

muitoolbox: muitoolbox.mltbx

The App file: SedTools.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^{TM}. 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:
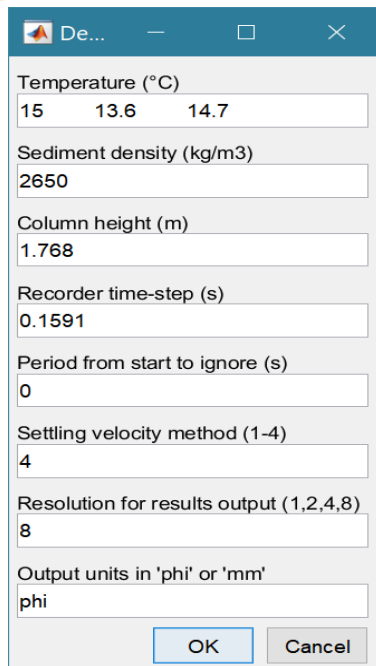
>> SedTools;

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

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

2.2   Model Set-up

File>New to create a new project space.

Setup>Settling Parameters: Allows the inputs required for the settling column analysis to be defined along with various options to control the output. Once added the current set of variables can be viewed using the *Settling* tab.

Temperature during the settling experiment (*).

Density of the sediment used (*).

Height of the settling column (*).

The time step used to record the settling data

The period from the start of the recording to omit from the analysis

Computation method for settling velocity (+)

The resolution to be used in the output (values are 1/phi)

Units to display output

(*) If multiple data sets are being averaged, multiple values can be entered, with the number of values equal to the number of input files. Otherwise the first value is replicated for all data sets.

(+) Methods: 1 = Gibbs et al. (1971); 2 = Baba and Komar (1981); 3 = van Rijn (1993); 4 = Soulsby (1997).

Run>Run Settling Analysis: Runs the settling analysis. The user is prompted to name the output, the default being the input file name.

Plot>Plot menu

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

Plot>Output Definitions: provides the ability to edit the Output Definitions used to define output variable names and labels for each model or data set. This edits the model or data set class property ResDef within the project but does not change the class definitions.

The *Plot* tab provides a summary plot of the output as marginal and cumulative frequency curves, along with a statistical summary table. The statistical data can be copied to the clipboard using the button below the table.

3   Application Menus

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

3.1   File

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

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

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

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

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

3.2   Tools

Tools>Refresh: updates *Cases* tab.

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

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

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

3.3   Project

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

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

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

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

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

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

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

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

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

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

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

3.4   Setup

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

Setup>Import Data: dialog with sub-menu options to Load, Add, Delete, Quality Control. The availability of these options may vary depending on what is defined in the data specific format file.

Select one or more files to load. Once added the current set of variables can be viewed using the *Inputs* tab. When the data has been loaded, the user is prompted to provide a description of the data set (scenario) and is listed on the *Cases* tab. The source file(s) area also listed on the *Inputs* tab.

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

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

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

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

Setup> Input Parameters: enter and edit the specified model parameters.

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

3.5   Run

Run> Run Model: runs model, prompts for Case description which is added to the listing on the *Cases* tab and the results can be displayed and plotted from the Q-Plot tab.
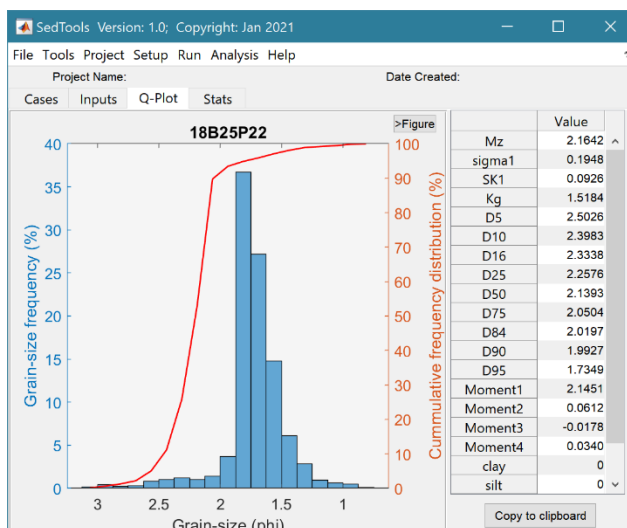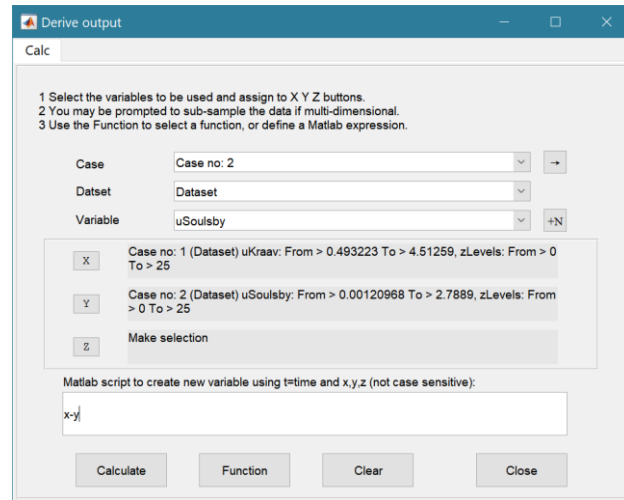

The >Figure button can be used to create the plot as a stand-alone figure.

The table can be copied to the clipboard using the Copy button.

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

The selection is assigned by clicking one of the X, Y or Z buttons. The user is prompted to assign a Variable, XYZ dimension, or Time (the options available varies with the type of variable selected) – see Section 3.9 for details of how this works. An equation is then defined in the text box below using the x, y, z or t variables[1]. 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. Comments can be used to pass additional instructions, such as the inclusion of the RowNames in the output to be saved as a new dataset, using either %time or %rows.

For example, any of the following could be entered into the equation box:

    x.^2+y   %time

    myfunction1(x,y,t,'usertext')

    myfunction2(x,mobj)

The output from function calls can be figures or tables, a single numeric value, or a dataset to be saved (character vector or array). External functions must return output as a cell array with the new variable in the first cell and data to be used to define RowNames in the second cell. If the %time or %rows instruction is included in the call, row data are added providing that the length of the input dataset matches the output dataset. If there is no output to be passed back the function should return a cell array containing the string 'no output' to suppress the message box, which is used for single value outputs. For expressions that return a result that is the same length as one, or more, of the variables used in the call, there is also the option to add the variable to one of the input datasets as a new variable. In all there are three ways in which results can be saved:

1. Expression or function returns a result that is the same number of rows as one or more of the input datasets. Option to (a) add as a new variable to an existing data set, or (b) create a new dataset with no assignement to the RowNames property.
2. As (1) with the comment of %time or %rows in the call. Attempts to use the RowNames property of one of the inputs to define RowNames in a new dataset. Requires the input variables to have same number of rows. (Plan to add interpolation so this may change).
3. Expression or function returns a result with a new variable and time time in a 2-element cell array. The variable is saved as a new 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 can be selected but they are defined in the call using dst, for example:

---

[1] 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.

> myfunction3(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 cell array containing one or more data sets. The options for saving the data are the same, with the additional option that when a dstable, or struct of dstables, is returned, these are saved directly and it is assumed that the dsproperties have been defined in the function called.

Some further details on using this option and the '**Function**' library available are provided in Section 4.2.

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.
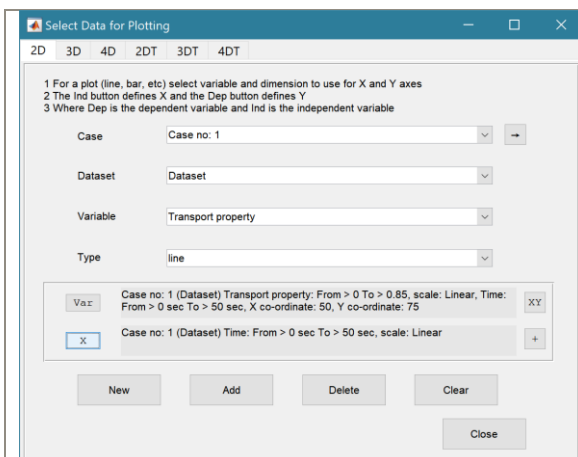
<table>
<tr><td>

</td><td>

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.

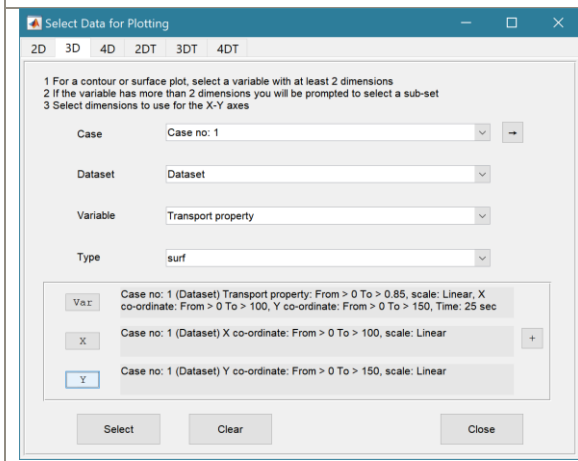> Select plot type (line, bar, scatter, stem, etc)

<u>Control Buttons</u>:

$\rightarrow$: 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.

</td></tr>
<tr><td>

</td><td>

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.

<u>Control Buttons</u>: see 2D plot above.

</td></tr>
</table>

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

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 psfunctions folder and illustrates the workflow to a simple line plot using x-y data from the 2D tab and a surface plot using x-y-z data from the 3D tab.
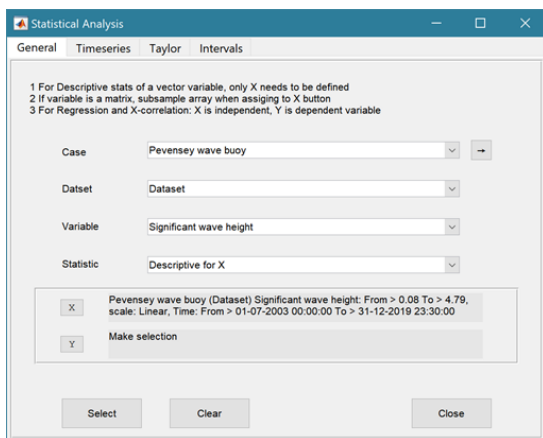
3.6.2   Statistics

Analysis> Statistics: several statistical analysis options have been included within the Statistical Analysis GUI. The tabs are for *General* statistics, *Timeseries* statistics, 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.
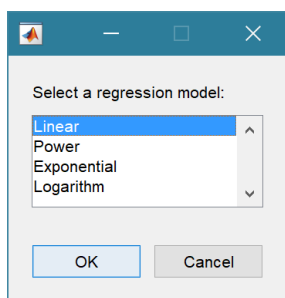
2) **Regression**: generates a regresion plot of the dependent variable, Y, against the independed variable, X. For time series data, the default data range is the maximum period of overlap of the two records. For other data types the two variables must have the same number of data points. After pressing 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-corrleation 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.
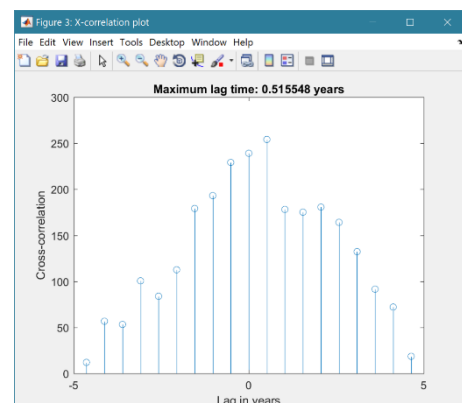
4) **User**: calls the function user_stats.m, in which the user can implement their own analysis methods and display results in the UI or add output to the project Catalogue. Currently implements an analysis of clusters as detailed for Timeseries data below.

Taylor tab

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

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

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

Selecting New generates a new Taylor Plot. Selecting the Add button adds the current selection to an existing plot and the Delete button deletes the current selection. The Clear button resets the UI to a blank selection.
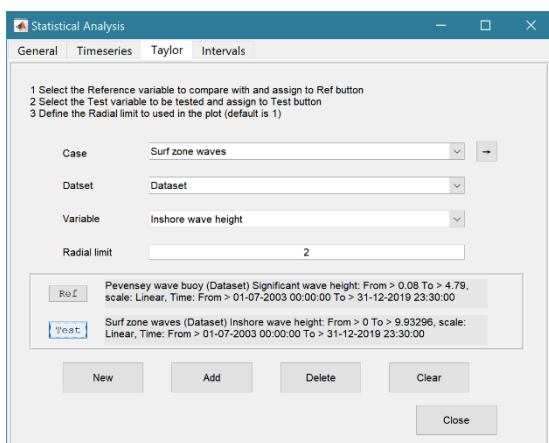
Once New or Add are selected, the user is asked whether they want to plot the skill score (Yes/No). If Yes, then the user is prompted to set the skill score parameters. As further points are added to the plot, this selection remains unchanged (i.e. the skill score is or is not included). To reset the option it is necessary to close and reopen the Statistics UI.

If the number of points in the Reference and Test datasets are not the same the user is prompted to select which of the two to use for interpolation.
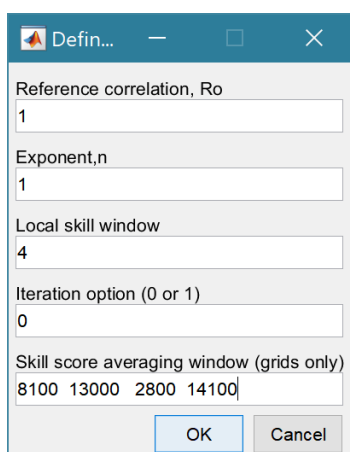
This is the maximum achievable correlation (see Taylor (2001) for discussion of how this is used).

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

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

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

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

| (a) time series skill score plot	(b) grid skill score plot
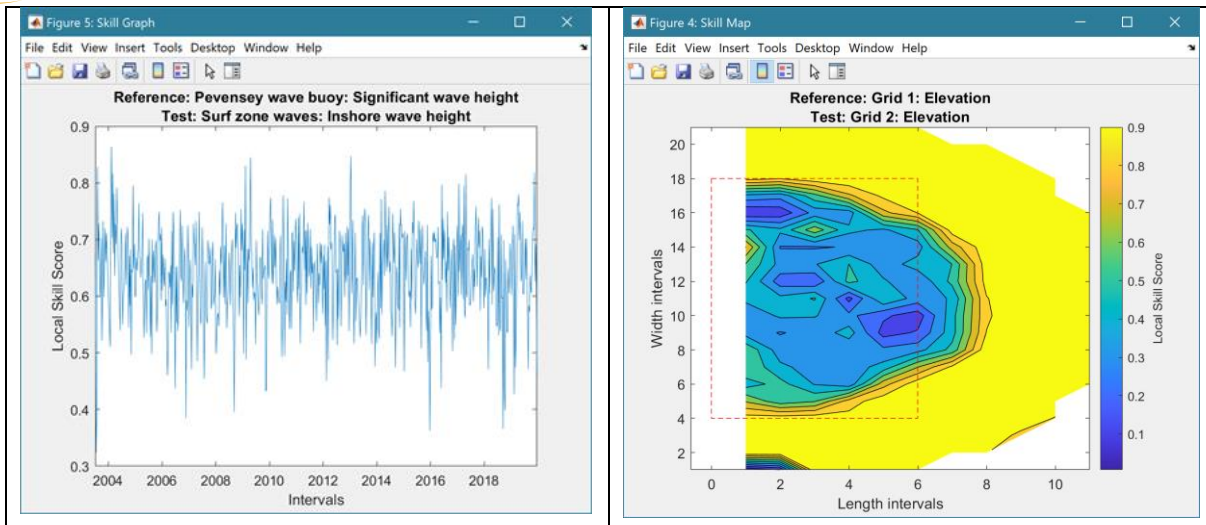
The Taylor Plot shows the Reference point as a green cross and the Test points as coloured circles. The legend details the summary statistics and the Case List button generate a table figure listing all the results. These can be copied to the clipboard.
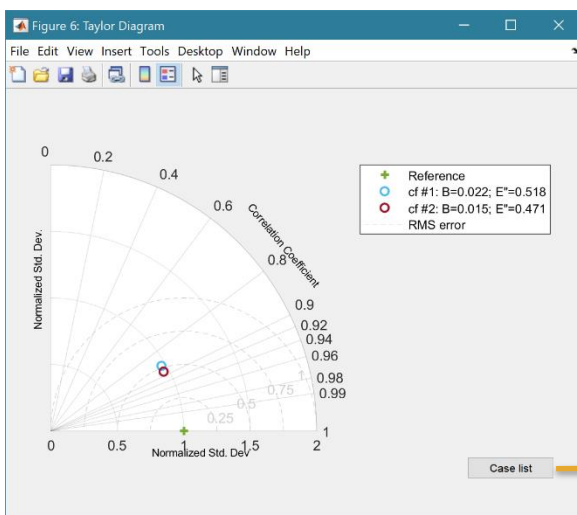
Taylor diagram legend includes: B – bias; E'' – normalised RMS difference

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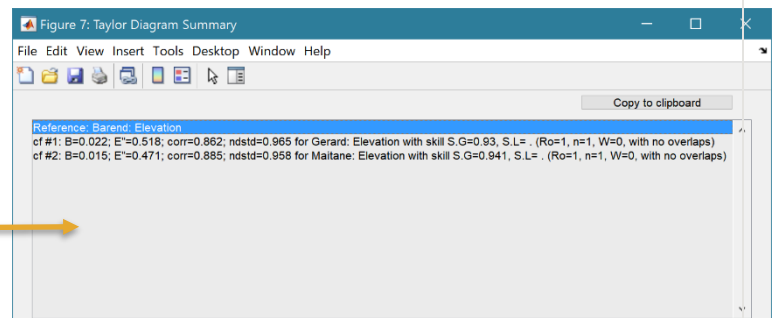
3.7 Help

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

3.8 Tabs

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

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

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

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

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

3.9   UI Data Selection

Functions such as Derive Output (3.5), Plotting (3.6.1) and Statistics (3.6.2) use a standardised UI for data selection. The Case, Dataset and Variable inputs allow a specific dataset to be selected from drop down lists. One each of these has been set to the desired selection the choice is assigned to a button. The button varies with application and may be X, Y, Z, or Dependent and Independent, or Reference and Sample, etc. Assigning to the button enables further sub-sampling to be defined if required. Where an application requires a specific number of dimensions (eg 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) is also selected on this UI. The selection is defined in two steps:
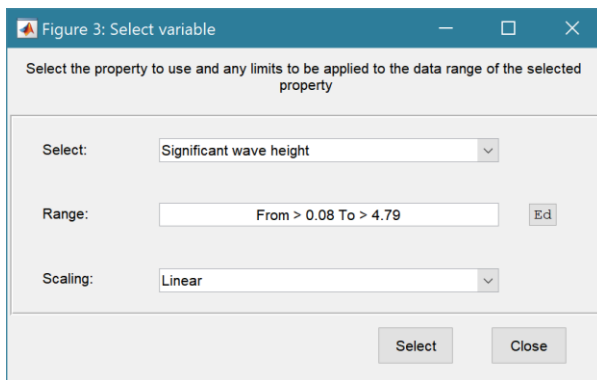
Step 1.

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

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

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

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

The number of variables listed on the UI depends on the dimensions of the selected variable. For each one Select the attribute to use and the range to be applied.
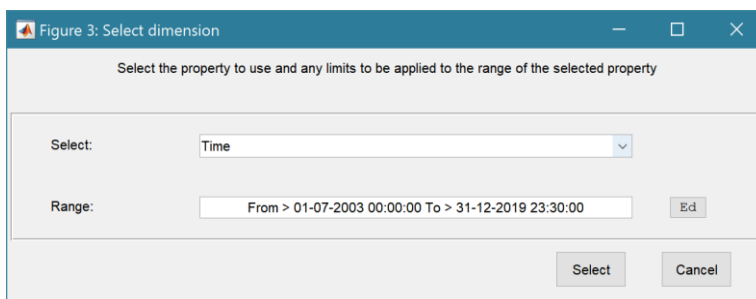
Step 2 - Variable only has dimension of time.

No selection to be made.
Edit range if required.

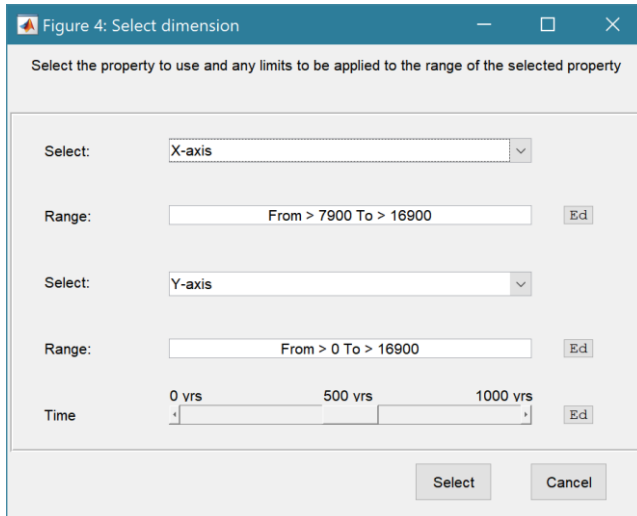**Step 2 - Variable has 3 dimensions but only 2 are needed for the intended use**.

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

Select the 2nd variable to use as a dimension
Edit range if required.

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

Press Select to accept the selection made.

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

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

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

11

4   Background to Analysis Methods

4.1   Cal_settling function

Cal_settling was written to process the data of the settling column of the School of Ocean and Earth Science, Southampton Oceanographic Centre, University of Southampton.

- The input data must be the readings of the scale (units do not matter) recorded with a fixed time-step. The time-step MUST be written as a constant in the routine. The input data can either be given as a text file with one weight record per line (and nothing else), or as Matlab array.

- The routine can use 4 different formulae for converting the settling velocity into grain size: Gibbs et al. (1971), Baba and Komar, (1981), van Rijn (1993) and Soulsby (1997). The used formula is selected by a constant in the routine. It is recommended to use the formula of Soulsby (1997).

- The results are given in Phi units, by default with 1/8 Phi resolution, but other resolutions can be selected.

- It is possible to average together several analyses, for example replicates produced from sub-samples, by giving the multiples filenames in a cell array of strings. Different temperatures or settling heights can be specified. It is strongly recommended to FIRST process each replicate separately and to examine the quality and the similarity of the results BEFORE averaging the replicates of one sample.

- Cal_setting produces a figure representing the grain-size frequencies and the cumulative frequency distribution (see example below). The statistical parameter (either from the graphical method or the moment method, selection through a constant in the routine) can be printed on this figure. The figure drawing can be deactivated with an option. This figure can be printed from Matlab, or copied-pasted from Matlab into another program.

- With a particular option, the routine copy the numerical results in the system clipboard, so that they can be pasted into another program like Excel.

- The routine can also return the settling velocities for a given temperature and grain-size density.

- The settling column in Southampton often records a slight mass increase in the first seconds, which does not correspond to settling grains but represent some noise. This produces a peak in the coarsest grain-size class. To remove this effect, use the option "ixx", which ignore the xx first seconds (example: "i08" for 8 seconds).

See source of core program and further details at: http://neumeier.perso.ch/matlab/cal_settling.html.

4.2   Derive Output

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

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

```
heq = str2func(['@(t,x,y,z,utext,mobj) ',usereqn]); %handle to anonymous
function
var = heq(t,x,y,z,utext,mobj);
```

This function is then evaluated with the defined variables for *t, x, y*, and *z* and optionally *utext* and *mobj*. *utext* allows text string to be passed (any string enclosed in single quotes, e.g. 'Test') and *mobj* passes the handle to the main UI.

Some functions may alter the length of the reference co-ordinates (x, y, z, t), or return more than one variable. These can be handled by passing a comment appended to the function definition e.g.

`subsample(x,t, thr,mobj) %time`. In the current version of the toolbox, only '*time*' is handled as a key word and this allows functions to derive a new timeseries with different time input values to those of the input variable(s).

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 3.5). The variables selected are sub-sampled based on any adjustments made when each variable is assigned to a button. The dimensions of the vector or array with these adjustments applied need to be dimensionally correct for the function being called.

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 derived 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.

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.

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.

4.2.1   Calling an external function

This interface 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 SedTools are detailed in the next Section.

The input variables for the function must match the syntax used for the call from the Derive Output UI. In addition, functions must return output as a cell array. This can be a single value, a new variable vector or array or a time series with one or more variables. If *time* is included then this should be in the first cell with variables in the subsequent cells.

If the %time or %rows instruction is included in the call, row data are added providing that the length of the input dataset matches the output dataset. If there is no output to be passed back, the function should return a cell array containing the string 'no output' to suppress the message box which is used for single value outputs.

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 but they are defined in the call using dst, for example:

`mufunction3(dst, 'utext', mobj)`   where '*utext*' and *mobj* are options as defined above.

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 cell array containing one or more data sets.

13

Adding functions to the Function library

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

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

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

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

4.2.2   Pre-defined functions

The following examples are provided within SedTools, 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 ca be used: y, d, h, m, or s. Returns a time series based on the defined *tstep*, where the time used is for the beginning of each stepping interval, i.e. every *tstep* from the start of the record to the nearest interval that is less than *tdur* from the end of the record.

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

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

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

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

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

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

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

   `subsample(X, t,  thr, mobj)`, where X and t are the variable to be subsampled, *thr* is the threshold value and *mobj* is the UI handle (must be *mobj*). The user is prompted to select the dataset and variable to be used to define the condition and a condition operator (<=, ==, 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. ***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.

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

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

5   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
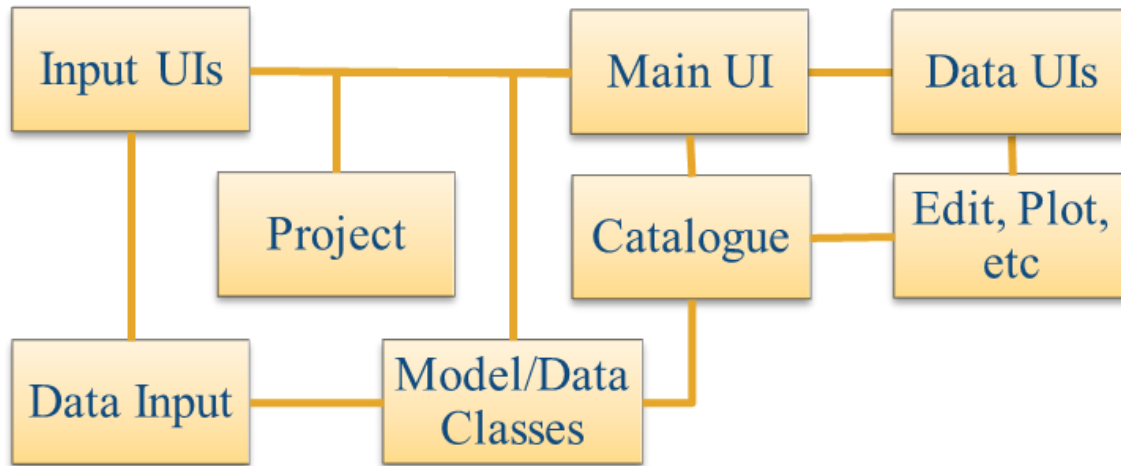
The interfaces and default functionality are implemented in the SedTools App using the following muitoolbox classes depicted in Figure 2, which shows a more detailed schematic of the program structure. See the muitoolbox and dstoolbox documentation for more details.

Figure 2 – schematic of program structure showing how the main classes from muittoolbox and dstoolbox are used
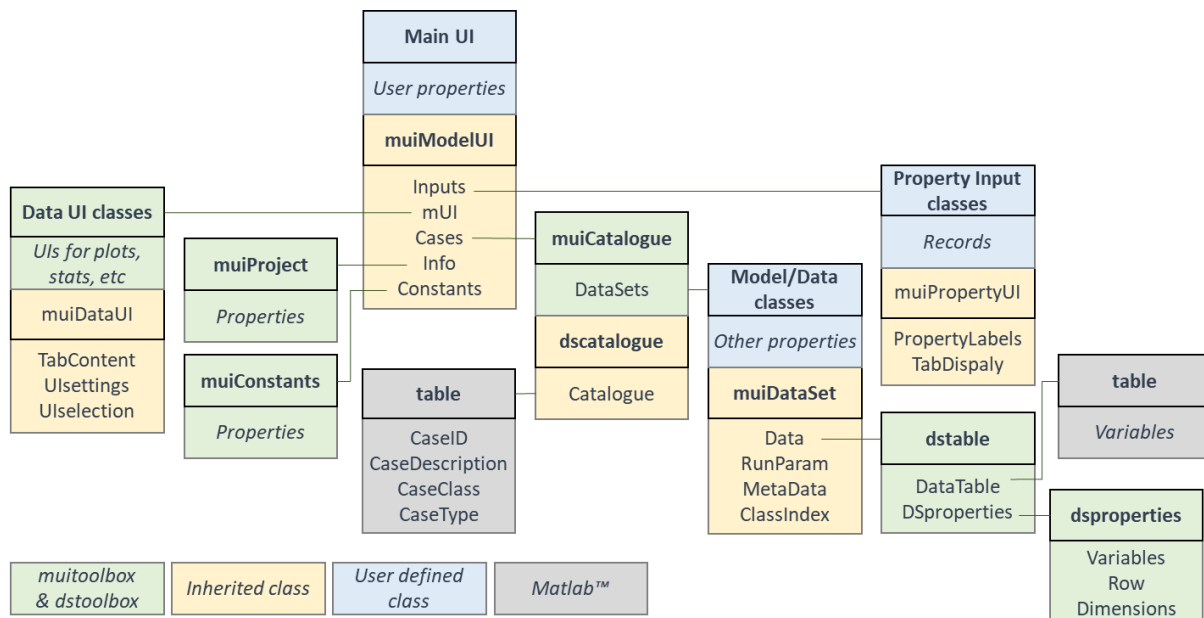
In addition, the SedTools App uses the following classes and functions:

SedTools – UI for SedTools

SettlingParams – data entry for settling column analysis

SettlingAnalysis – settling column results and plotting

settling_column – function that implements cal_settling by Urs Neumeier, 2005.