Introduction to the dstoolbox

Ian Townend

The dstoolbox

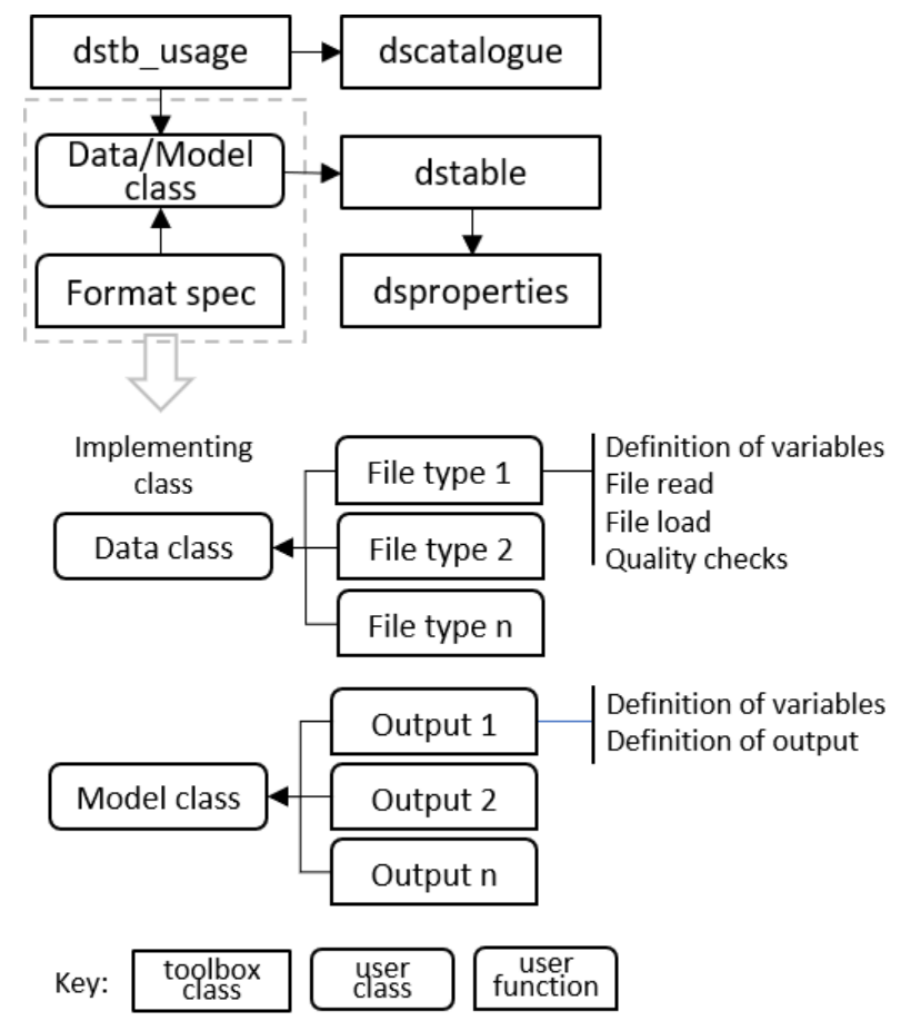
The *dstoolbox* is designed to store and manage multi-dimensional data sets, including meta-data of the variables and all dimensions and manage access to a collection of classes that hold data sets using a catalogue.

In the schematic outline *dstb_usage* is a class to illustrate how *dstable*, *dsproperties* and *dscatalogue* are used.

Data are loaded into a *dstable* with relevant metadata added to the table and made accessible using *dsproperties*. Each time a class adds data a record is added using *dscatalogue*.

The 'Format Spec' user functions, shown in the upper part of the figure, are implemented with functions, indicated by 'File Type' and 'Output Type', shown in the lower part of the figure.

These define the meta-data of the data set being saved (and any input parameters, or details needed to read and load data from a file, depending on the application).



COASTALSEA.UK

Installing the Toolbox

Download *dstoolbox.mltbx file* from:
https://github.com/CoastalSEA/dstoolbox/releases/

The toolbox can be installed using the

Add-Ons>Manage Add-Ons option on 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 (option on Home tab)

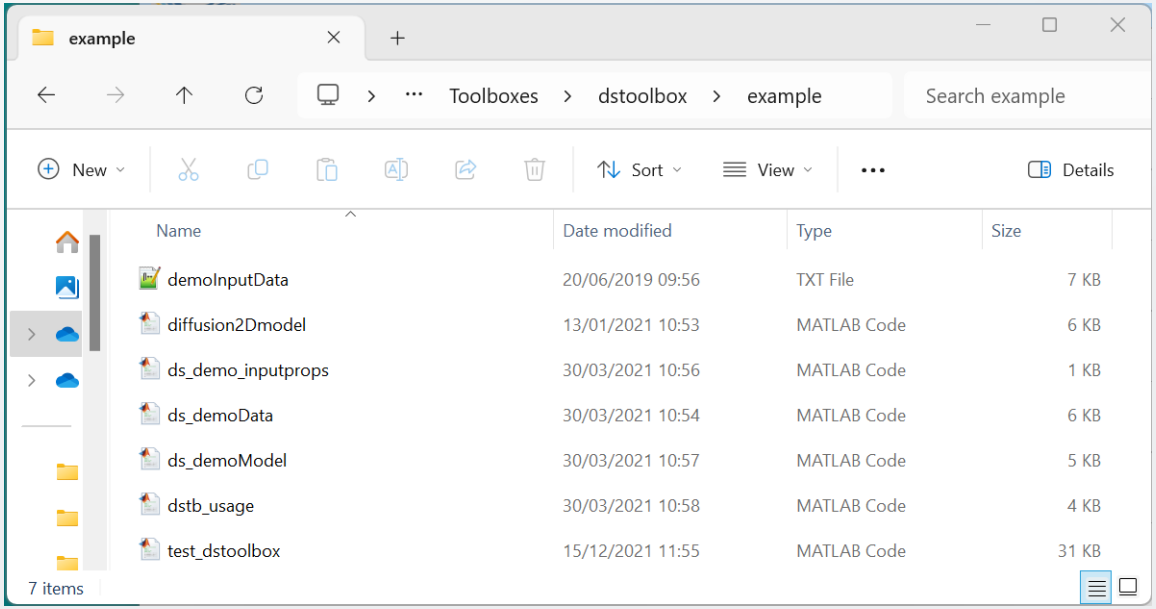
When dstoolbox is installed, documentation can be accessed using the Matlab™ Supplemental Software documentation.

Install toolbox and copy templates

Create a new working folder

Copy files from …\MATLAB Add-Ons\Toolboxes\dstoolbox\example

Folder contains some examples of Matlab code to use dstoolbox, with some supporting functions and data:



COASTALSEA.UK

Typical uses of dstable

dstable handles data sets that vary in any combination of variable, v, and dimensions x, y, z and t.

Models provided in the ModelUI App illustrate different types of application:

- Vertical Tidal Profile: simple x-v data set
- SimpleTide: time data (t-v and no x, y, z)
- 2D diffusion: variable in x, y, t (can also include z for 3D case)

Other Apps illustrate a wide range of applications

- Data analysis – CoastalTools, SedTools
- Models – Asmita, ChannelForm, WaveRayModel, CSTmodel
- Analysis – ModelSkill
- Design – MRBreach

Data storage options (1)

dstable is an extension of Matlab™ *table* class

- The data are stored in a *table* with a wrapper that provides for direct access to dimensions and variables as well as the data set properties (*dsproperties*) that contain the metadata for both variables and dimensions.
- Unlike a *table*, row names can be of any data type and are treated as one dimension of N-dimensions that can be defined.
- By convention, rows are often used for the time dimension, with the other dimensions defining spatial or other dimensions. But this is not a requirement.
- The dimension assigned to rows is checked to ensure that all values are unique and match the first dimension of the variables stored. Additional dimensions are not checked.
- By adding multiple Dimensions, different Dimensions can be used in conjunction with different variables.

Data storage options (2)

Each variable is stored in a cell of the table and can be scalar or an N-dimensional array. For example:

1. Variables are scalar and there are no dimensions;
2. Variables are row vectors [1xn] and dimensions defined to match, D1(1:n) – the row dimension is not used;
3. Variables are column vectors [mx1] and a dimension is defined for the row dimension, D1(1:m);
4. Variables are matrices [mxn] and dimensions are defined for a row, D1(1:m) and one additional dimension, D2(1:n);
5. Variables are matrices [1xmxn] and dimensions are defined for two dimensions D1(1:m) and D2(1:n) – the row dimension is not used;
6. Variables are temporal matrices [txmxn] and dimensions are defined for a time as a row, T(1:t) and two additional dimensions D1(1:m) and D2(1:n);
7. Variables are multi-dimensional [mxnxpxqx…N] and dimensions are defined using row for D1(1:m) and additional dimensions for D2(1:n) through to DN(1:N).
8. Variables are multi-dimensional [1xmxnxpxqx…N] and dimensions are defined for D1(1:m) through to DN(1:N) – the row dimension is not used.

Data set properties

Properties area defined using *dsproperties* for each variable and the dimensions, which may need to split into row and other dimensions. The format for each of these is similar defining:

- Name
- Description
- Unit
- Label
- QCflag or Format

```
dsp = struct('Variables',[],'Row',[],'Dimensions',[]);
dsp.Variables = struct(... %cell arrays can be column or row vectors
        'Name',{'Var1','Var2','Var3'},...
        'Description',{'Variable 1','Variable 2','Variable 3'},...
        'Unit',{'u1','u2','u3'},...
        'Label',{'Label1','Label2','Label3'},...
        'QCflag',{'qc1','qc2','qc3'});
dsp.Row = struct(...
        'Name',{'Time'},...
        'Description',{'Row Description'},...
        'Unit',{'time'},...
        'Label',{'s'},...
        'Format',{'dd-MM-yyyy'}); %only used for datetime and duration formats
dsp.Dimensions = struct(...
        'Name',{'Dim1';'Dim2'},...
        'Description',{'Distance 1';'Distance 2'},...
        'Unit',{'u1';'u2'},...
        'Label',{'Label1';'Label2'},...
        'Format',{'-';'-'}); %only used for datetime and duration formats
```

dstoolbox classes

dstable, holds a collection of one or more datasets with one or more common dimension vectors and the associated metadata.

dsproperties, defines and maniputates the struct used to assign the metadata to a *dstable*.

dscatalogue, manages a catalogue of data sets that handle a collection of data sets (e.g. imported and model data), which are loaded into *dstables* and catalogued using *dscatalogue*.

See dstoolbox documentation for details of properties and methods available in each class.

dstoolbox example folder (1)

The ‘example’ folder contains 3 classes and some functions to help illustrate the use of the toolbox:

- *dstb_usage*: a class to illustrate the combined use of data and model classes that use *dstable* and *dsproperties*, with a record for each data set held in *dscatalouge*. An option to run *dstb_usage* is included in the function *test_dstoolbox*.

- *ds_demoData*: a class to load data from a file and store it in a *dstable*. The class includes methods to define the *dsproperties*, read the input file format, load the data into a *dstable* and plot some output.

- *ds_demoModel*: a class to run a simple model (2D diffusion using hard code parameter settings) The class includes methods to run the model, save the results, and plot the model output.

Example of usage

```
%test the components of the toolbox using the dstb_usage class

%initialise class that manages calls to models and data classes
dm = dstb_usage;
%run model twice and load two data sets
run_a_model(dm);
load_data(dm);
run_a_model(dm);
load_data(dm);
%plot results
plotCase(dm);
%display DSproperties of a selected Case
displayProps(dm);
```

The code for each method in the *dstb_usage* class illustrates how the toolbox classes can be used. The above script can be run from the prompt using:

```
>> h = test_dstoolbox('dstb_usage');
```

dstoolbox example folder (2)

The ‘example’ folder also contains the function *test_dstoolbox* to experiment with some of the workflow options for *dstable*, *dsproperties* and *dscatalogue*. The comments included in the code explain each function call.

Call format is as follows:

>> h = test_dstoolbox(‘classname’, casenum, options);

e.g.

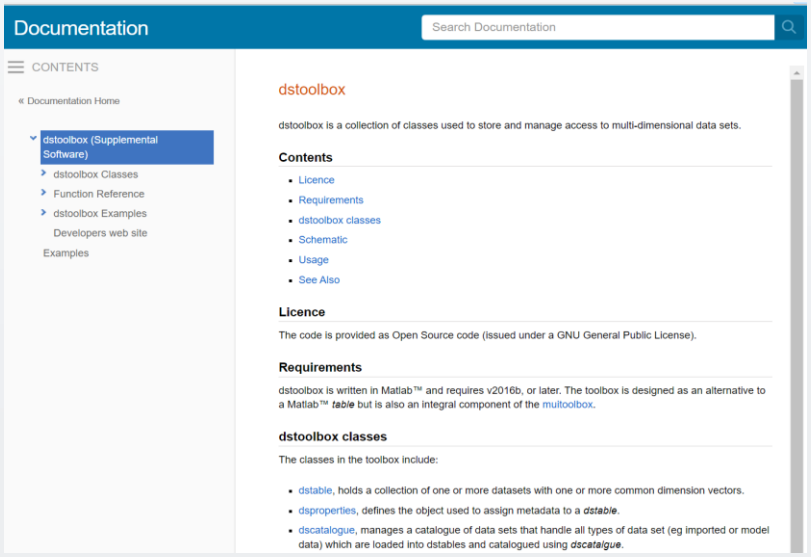
```
>>  h = test_dstoolbox('dscatalogue'); %add and remove records to a catalogue
>> h = test_dstoolbox('dsproperties', 2);            %call and set properties individually
>> h = test_dstoolbox('dsproperties', 6, [4,1,6]);   %assign a struct array
>> h = test_dstoolbox('dstable', 2, 6);              %create a simple table with rows and no dims
>> h = test_dstoolbox('dstable', 3, [1,3,5]);        %create table with t+2d array.
```

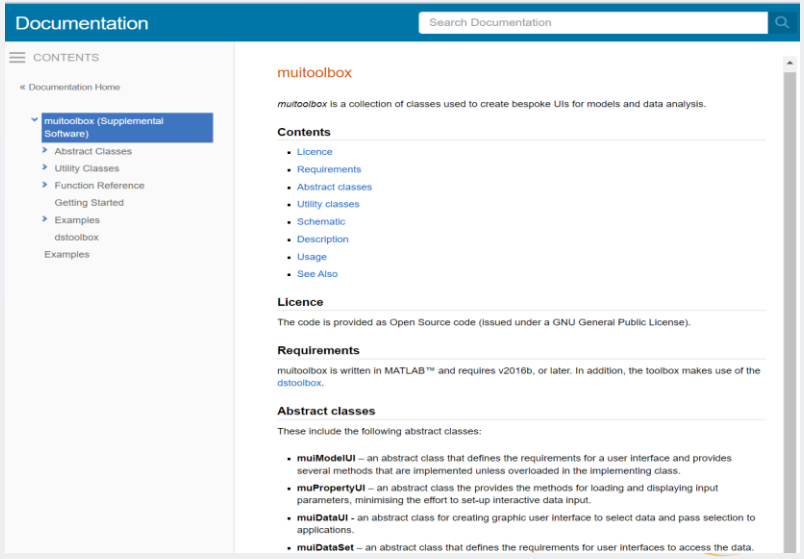
See *dstoolbox* documentation for further details about the options available.

Further information

Toolboxes and Apps are explained at www.coastalsea.uk and can be downloaded at https://github.com/CoastalSEA

When dstoolbox and muitoolbox are installed, further documentation can be accessed using the Matlab™ Supplemental Software documentation.





The muitoolbox

The purpose of the *muitoolbox* is to minimise the effort in creating or prototyping an interface for a model or data analysis tool.

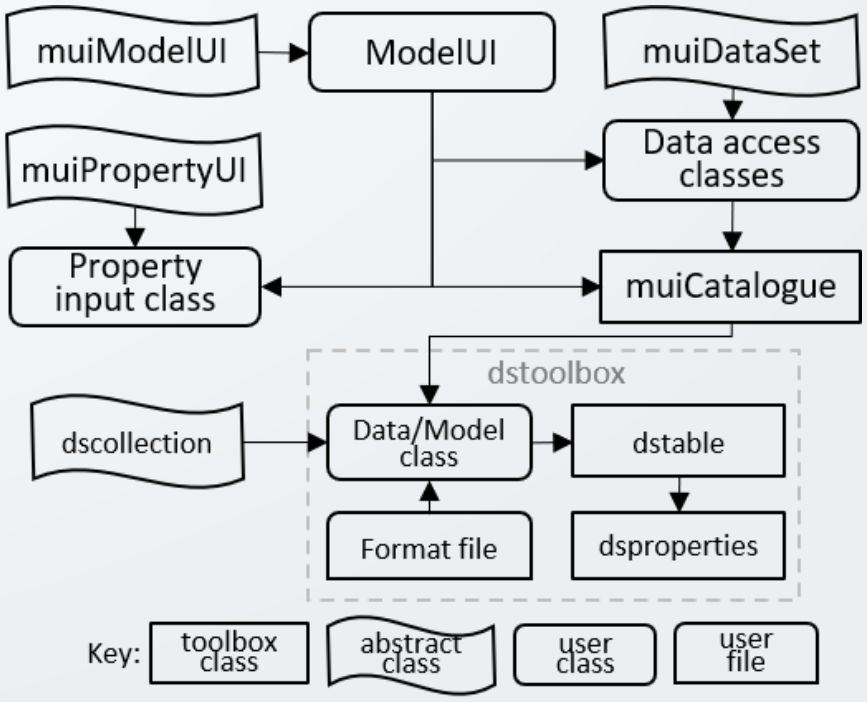
Creating a new model requires 3 components to be defined, namely the interface (ModelUI in the above illustration), one or more classes to manage the input of model parameters (if required) and the classes to hold imported data, or running a model and storing the output.

Central to this is the holding of input data in the Inputs property and accessing the data via the Cases property. In this context, Cases comprise a record of each Case and a dataset. The records are held in the Catalogue property and the dataset (an instance of the data or model class) in the DataSets property of *muiCatalogue*.

Each data or model class stores the dataset in the Data property, with additional information held in the RunData property (e.g. holding input parameters of a model run).

Any type of dataset can be stored in the Data property but when using the dstoolbox multidimensional data sets can be stored using *dstable* and a full set of meta-data attached using *dsproperties*.

The overall architecture and the properties that provide the links between one class and another are shown in the flow chart below



COASTALSEA.UK