Memo

To
The OpenEarth community

Date	**Reference**	**Number of pages**
March 9th, 2013	2.0-0.1	8

From	**Direct line**	**E-mail**
Wim van Balen	+31 (0)88 335 8592	wim.vanbalen@deltares nl

Subject
Description of conversion tool Delft3D to D-Flow FM, version 2.1 beta

Copy to
Arthur van Dam, Sander van der Pijl, Herman Kernkamp

This memo briefly describes the abilities of the conversion tool `ddd2dfm`. The name `ddd2dfm` stands for 'Delft3D to D-Flow Flexible Mesh'. The tool facilitates conversion of a Delft3D model to a D-Flow Flexible Mesh (DFM) model.

1 Configuration instructions

The conversion tool has been developed in a Matlab R2012a environment. The working of the conversion tool has not been tested on other Matlab versions. The conversion tool does not need further configuration before use. The files, together with this `pdf`-file, should remain located in one single directory if you run the tool independently from the OpenEarth repository. All these files are available in the OpenEarth repository.

2 Installation instructions

The conversion tool consists of a GUI, described by `ddd2dfm.fig`, and nine Matlab `.m`-files (all in one directory). The main file is `ddd2dfm.m`, the supporting files are, in alphabetic ordering:

- `bca2cmp.m`: converts Delft3D `bca`-files to DFM `cmp`-files (component files),
- `bcc2tim.m`: converts Delft3D `bcc`-files to DFM `tim`-files (timeserie files),
- `bct2tim.m`: converts Delft3D `bct`-files to DFM `tim`-files (timeserie files),
- `bnd2pli.m`: converts Delft3D `bnd`-files to DFM `pli`-files (polyline files),
- `dir2fil.m`: puts the Delft3D files from the directory into the listboxes before conversion,
- `grd2net.m`: converts the Delft3D grid from the directory into a DFM net format (netCDF),
- `mdf2mdu.m`: partly converts the Delft3D `mdf`-file to a DFM `mdu`-file (definition file),
- `pli2ext.m`: connects the created DFM `pli`-files to DFM `mdu`-file by means of a DFM `ext`-file.

The tool uses existing OpenEarth Matlab-scripts, such as `delft3d_io_grd.m`, `delft3d_io_bnd.m`, `delft3d_io_mdf.m` and `writeNet.m`.

3 Operating instructions

The converter can be run by just typing `ddd2dfm` in the command line of Matlab. Make sure that the OpenEarth repository is connected. The window that appears, is shown in Figure 1.

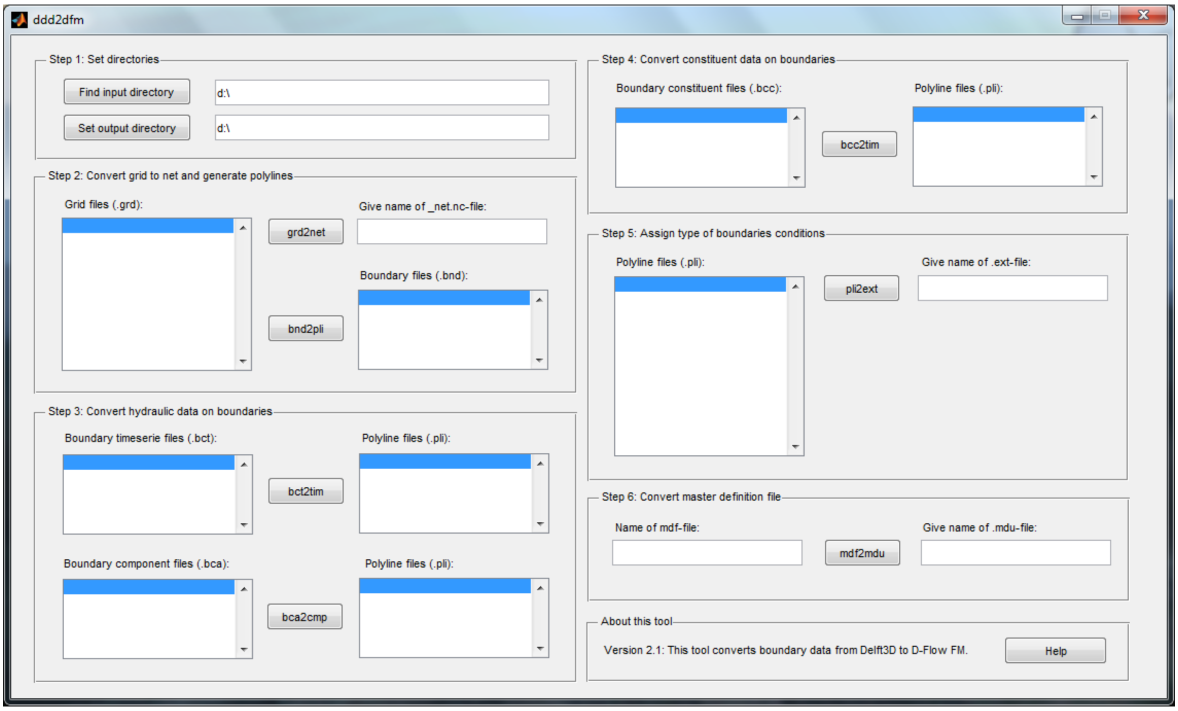

Figure 1: Opening screen of the conversion tool once after having typed `ddd2dfm` in the Matlab command line.

The window consists of a left panel and a right panel, which should be read in lexicographic ordering. The conversion algorithm is divided into multiple steps, which are addressed below subsequently.

Step 1: Set directories

The first step is to set the input directory and the output directory. One can use either the pushbutton and direct textual input in the available field followed by `Enter`. Once the input directory is set, the contents of Delft3D-type are processed and put into the listboxes automatically. The DFM files that will be generated are put in the output directory.

Step 2: Convert grid to net and generate polylines

In DFM, the locations where boundary conditions are to be imposed, are described by means of `pli`-files. The second step is first to convert the grid into a DFM network (in netCDF format) and then to generate these `pli`-files from one available `grd`-file and one available `bnd`-file. In Delft3D, the `grd`-file and the `bnd`-file follow the definition as illustrated in Figure 2.

The following files, which already exist in the OpenEarth repository, are used to import the data from the `grd`-file and the `bnd`-file:

- `delft3d_io_grd.m` to read the grid,
- `delft3d_io_bnd.m` to read the boundary specifications.

The resulting `pli`-files (four polylines in case of Figure 2) are stored separately and listed in the listboxes. For the constituents, separate but identical polyline files are stored with the addition of the characters `_sal` in the file names. The file names of the polyline files correspond to the file name of the `bnd`-files.

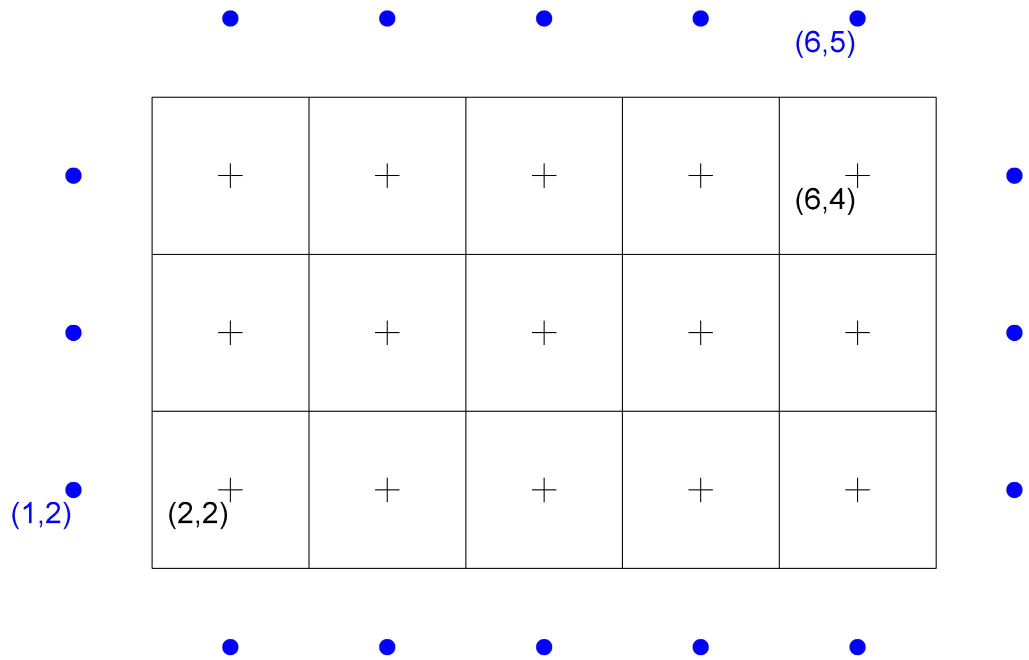

Figure 2: Definition of the grid in Delft3D, illustrated by a Cartesian grid with 5 $\times$ 3 cells. The corners are tabulated in the `grd`-file, the $+$ markers denote the cell centers, the blue dots denote the mirrored cell centers at the boundaries, referred to in the `bnd`-file. The blue dots span the polylines used by DFM.

Step 3: Convert hydraulic data on boundaries

The boundary conditions for the flow simulation should be given in `bct`-files (timeseries) or in `bca`-files (components). Other file types are not supported yet. Their DFM equivalent are `tim`-files and `cmp`-files, respectively. Depending on their presence, the listboxes are enabled or disabled. This conversion only works for polylines that are created using this conversion tool.

One single `bct`-file or `bca`-file can be selected to be coupled with multiple `pli`-files. During the process of conversion, the names of the boundary locations are stored in the `pli`-files. Using the `ctrl` button, one or multiple `pli`-files can be selected.

Once the button bct2tim or bca2cmp is pushed, the tool searches for the proper coupling of the original data to the newly generated polyline(s), based on agreement in specified and stored names. A waitbar sticks at 0% progress before the first agreement in boundary name specification is detected. As soon as agreement in names is detected, this waitbar starts monitoring the progress. This conversion of data can be a costly activity in case of extensive data files.

Step 4: Convert constituent data on boundaries

Currently, the conversion tool is only suitable to process salinity data. Temperature data conversion is not supported yet, since temperature has not yet been implemented in the DFM code. For the salinity, seperate but identical pli-files are used, marked with the addition _sal in the polyline file names.

The conversion by means of the button bcc2tim is quite comparable with the conversion via bct2tim. Again, agreement in boundary name specification is searched for, whereafter the conversion process itself is monitored using a waitbar. This conversion only works for polylines that are created using this conversion tool.

Step 5: Assign type of boundary conditions

DFM uses an ext-file to specify the type of boundary conditions. Since this is seriously different from Delft3D, a separate button is included to generate this ext-file. This conversion only works for polylines that are created using this conversion tool.

In the ext-file, the name of the polylines is connected to the actual time series or component data files. Using the ctrl button, a selection of multiple polylines can be specified. In this way, some boundaries can be ignored, if desired. The name of the ext-file itself is specified later on in the mdu-file.

The user should specify the name of the ext-file in the edit box provided. It does not matter whether or not this filename already contains the extension .ext.

Step 6: Convert master definition file

A Delft3D computation is driven by a master definition file, a mdf-file. Similarly, a DFM computation is driven by a mdu-file. From a given mdf-file, the conversion tool only reads the time-related data, for instance the computational timestep, frequencies of writing output data and so on.

In addition to this input, the names of the output data are slightly manipulated towards the different output data format used by DFM, i.e. netCDF. Hence, the history file and the map file will be specified with _his.nc and _map.nc as extension, respectively. For this, the name specified in the edit box is used.

4 Example

The conversion tool can the best be illustrated by means of an example. For this purpose, a Delft3D model for a coastal region is used. For this area, the following files are available:

- the grid file `coast.grd`,
- the boundary component file `coast.bca`,
- the boundary specification file `coast.bnd` and
- the master definition file `coast.mdf`.

After having set the input directory and an appropriate output directory, the screen looks as shown in Figure 3. It can be seen that the listboxes have been filled. The space that is assigned to the listboxes seems to be overdone, but can however be quite useful in case of, for instance, nested models, or models with multiple versions of boundary conditions.

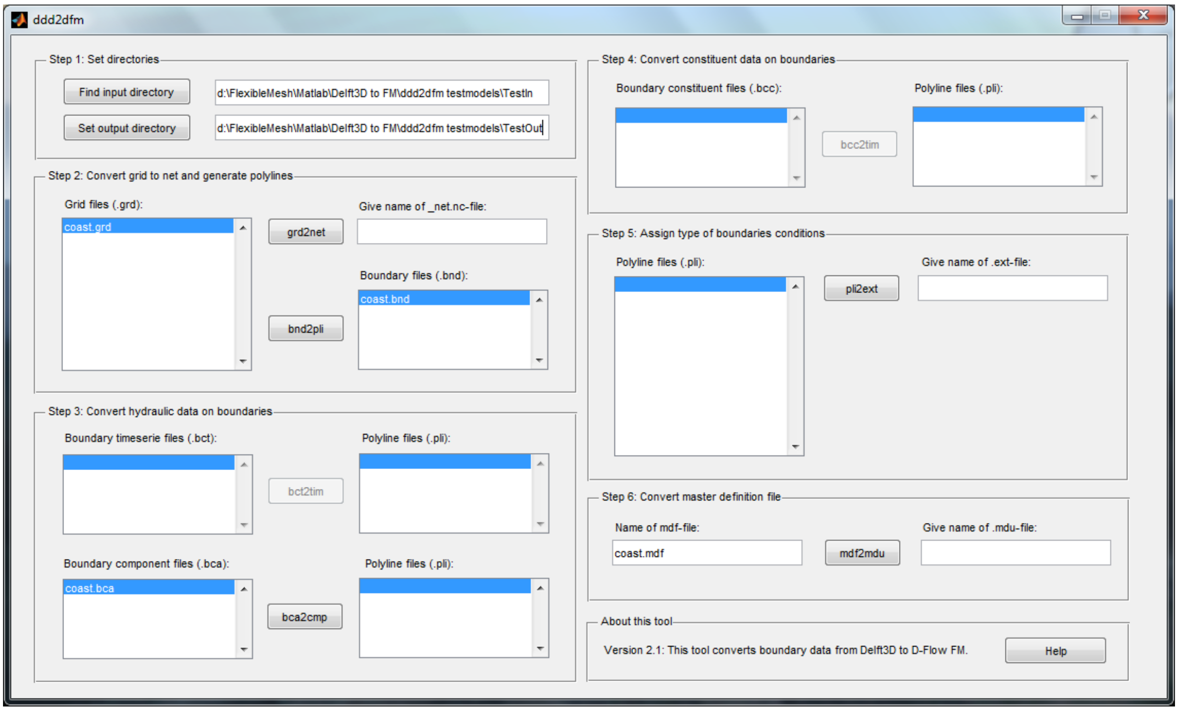

Figure 3: Screen after having set the input directory and an appropriate output directory.

The second step is to convert the grid and to couple the boundary specifications to the grid with polylines, describing these boundaries, as a result. Thereto, press `grd2net` and `bnd2pli`. After having pressed these buttons, three listboxes are filled with the resulting polylines. Since the Delft3D-model contains three boundaries with a dedicated prescription, three polylines appear in the listboxes. A message box will pop up if the operation has been succesfull.

The third step is to couple the component files. Since `bct`-files are absent, only the single `bca`-file should be treated. For this purpose, one can select a subset of all the polylines, or just all the polylines together by means of the `ctrl` button. Once having selected some polylines, just press `bca2cmp`. A message box will pop up if the operation has been succesfull.

Step 4 is irrelevant for this particular model, since salinity prescriptions are absent. However, one can see the differences in file naming for the salinity files compared to the hydraulic boundary files.

It is for the absence of salinity boundary conditions that the polylines for salinity (denoted with `_sal`) are kept out of consideration in step 5. During this step 5, the `ext`-file is generated. First, a file name should be specified, for instance `coast.ext` (with or without extension). Pressing the `pli2ext` button will actually deliver the `ext`-file.

Finally, the `mdu`-file should be created. For this, the Delft3D `mdf`-file is available. If one inserts an appropriate file name, for instance `coast.mdu`, and just presses the button `mdf2mdu`, then the `mdu`-file will be generated that only contains time information borrowed from the `mdf`-file. Other specific flow settings are deliberately kept out of consideration and should be treated by the modeler himself. This `mdu`-file is just the starting point for a DFM computation.

If one needs assistence, the user can press the button `Help`, whereafter this very `pdf`-file will appear. After having carried out all the above actions, the screen will look like shown in Figure 4.

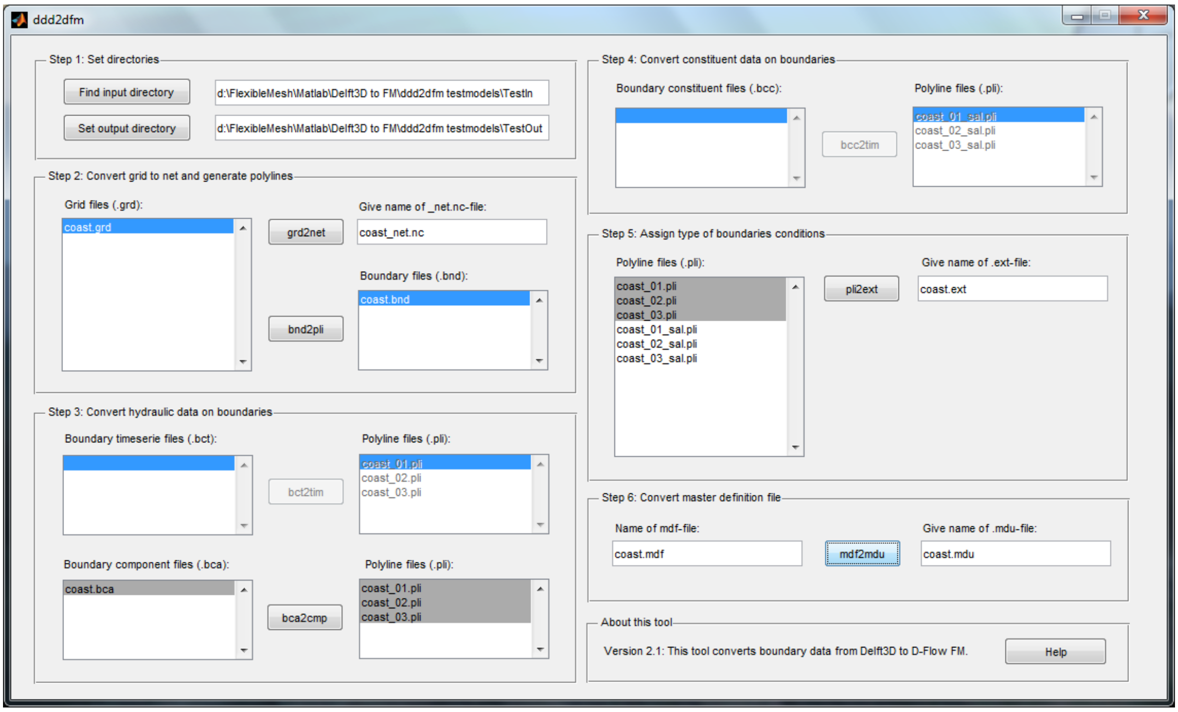

Figure 4: Screen after having followed all the relevant steps for the example model.

In the specified output directory, three `pli`-files (as well as three `pli`-files for salinity) and many `cmp`-files (each of them belonging to a single polyline point) have appeared as well as one `ext`-file, one `mdu`-file and the DFM grid.

Since all the required components of a DFM computation are present now, the `mdu`-file can directly be imported in DFM. The result of the conversion can immediately be examined. An additional polyline-file `coast_all.pli` has been generated to enable inspection of all the boundary polylines in one view. Figure 5 shows the result of importing the `mdu`-file and loading the `coast_all.pli` file, with a focus on the southern edge of the model.

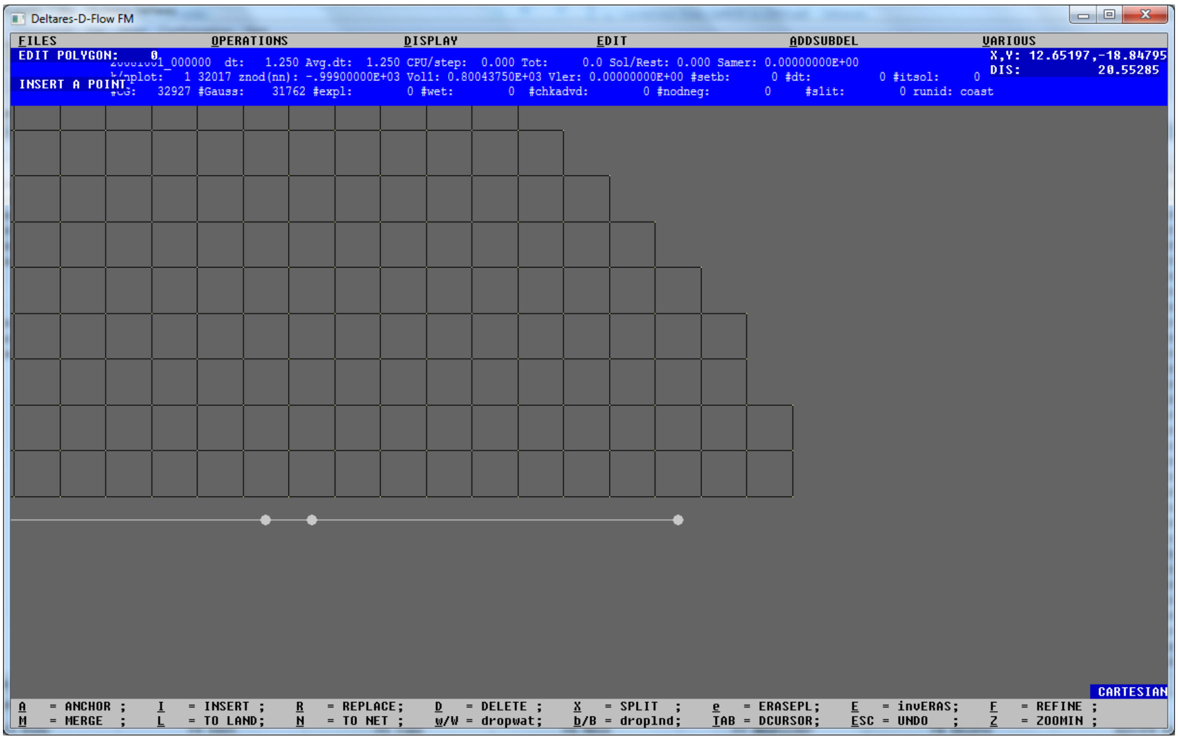

Figure 5: View on the Delft3D grid as well as a polyline at one of the three boundaries of the example model.

5 Disclaimer

The conversion tool has clear advantages regarding the ease with which a Delft3D model can be translated towards a DFM model. In spite of this, several known deficiencies of the conversion tool should be mentioned:

1. Although this conversion tool has been tested on multiple Delft3D models of several kinds, this conversion tool is be no means claimed to be robust. As a result, crashes are not guaranteed to not occur. If crashes do occur, please let us know (see contact information above).
2. Currently, the conversion tool can only convert salinity data. Temperature data, or sediment data, cannot be processed yet. At this moment, DFM does not contain a module that computes the temperature in the first place.
3. Since the code of DFM is only suitable for two-dimensional cases, at this very moment, also the conversion tool is only made suitable for two-dimensional models.
4. The conversion tool only works if the proposed workflow is utilized sequentially. This draw-back is a direct result of the choice to divide the conversion into multiple steps. To establish this division into multiple steps, some data of key importance are stored in the polyline files. Once somebody has manipulated and saved the polylines within the DFM environment, this key information is deleted, thus preventing any further succesfull treatment by means of the conversion tool.
5. Currently, the `mdu`-file is only partly filled with information that stems from the `mdf`-file. A modeler himself should further fill the `mdu`-file with appropriate input.

6 Copyright and license information

7 Change log

The following versions have been put in the OpenEarth repository:

- February 20th, 2013 – version 2.0: basic version, aimed at boundary conditions conversion.
- March 9th, 2013 – version 2.1, with:
  - option to convert a Delft3D-grid to a DFM-grid,
  - adapted GUI,
  - added an overall polyline file for inspection of all boundaries,
  - improved conversion of a Delft3D `mdf`-file to a DFM `mdu`-file.

8 Contact information

For any further information or bug reporting, please use the contact information as given in the header of this memorandum. Feel free to give us feedback on the performance of the conversion tool, either in case of failure and in case of success.