Memo

To		
The OpenEarth community		
Date	**Reference**	**Number of pages**
June 13th, 2014	1.1.10843	10
From	**Direct line**	**E-mail**
Wim van Balen	+31 (0)88 335 8592	wim.vanbalen@deltares nl
Subject		
Description of the Matlab conversion tool Delft3D to D-Flow FM		
Copy to		
Theo van der Kaaij		

1 Functionality

The Matlab tool **dflowfmConverter** is a set of scripts that could help you to convert a Delft3D flow model to a D-Flow FM flow model. The tool consists of a straightforward Graphical User Interface with underlying scripts dedicated to reading, writing and converting specific components of the flow model. In the brief memorandum, the working of the converter is described. The reader is supposed to have at least fair knowledge of the very basics of D-Flow FM.

The converter (in its current version) facilitates the following items:

- the conversion of a curvilinear Delft3D grid to an unstructured D-Flow FM grid,
- the administration of the Delft3D bathymetry within the D-Flow FM grid file,
- the translation of the boundary conditions locations to polylines,
- the boundary conditions data conversion from Delft3D to D-Flow FM (supported boundary condition types are timeseries and harmonic/astronomic components for waterlevels, discharges (*not* yet discharge per cell), velocities and Neumann-boundaries),
- the boundary conditions data conversion for salinity; other active or passive constituents (temperature, sediment, etc.) are not yet supported by the converter,
- the coupling of wind-files; currently, only unimagdir-winds and cyclone-winds are supported by the converter,
- the conversion of output-related files, being the observation stations data and the cross-sections data,
- the conversion of spatially varying, model specific data, namely the roughness, the viscosity, the initial conditions for waterlevels, dry points and thin dams, and weirs,
- the production of the elementary external forcing file and the master definition file for D-Flow FM.

The converter is developed for the conversion of **two-dimensional** flow models. Hence, the conversion of the boundary conditions for three-dimensional flow models is not supported. The converter does not manipulate provided model data; it only transfers data to a suitable place in the D-Flow FM model setup.

A known drawback of the current tool has to do with the conversion of boundary condition data for discharges and velocities. These quantities have a certain sign which is, in Delft3D, related to the grid definition and, hence, of the orientation of the quantity under consideration. In D-Flow FM, an inflowing discharge is of positive sign (definition). This discrepancy is not dealt with yet and is therefore prone to be resolved in the near future. For the time being, discharges are always interpreted as *inflowing* fluxes having a positive sign.

2 Disclaimer

Although this conversion tool has successfully 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). Anybody who uses this converter for own purposes will stay fully responsible for the full definition of his/her D-Flow FM.

3 Configuration

The conversion tool has been developed in a Matlab R2012b 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.. All these files are available in the OpenEarth repository:

```
https://svn.oss.deltares.nl/repos/openearthtools/
```

The conversion tool consists of a GUI, described by the binary file `dflowfmConverter.fig` and the file `dflowfmConverter.m`, and a set of Matlab `.m`-files available at:

```
.../trunk/matlab/applications/delft3d/conversion/src/
```

The converter heavily relies on scripts that are already available within OpenEarth. Particularly the following repository directories are relevant:

- `.../trunk/matlab/applications/delft3d/flow/`,
- `.../trunk/matlab/applications/delft3d/nest_matlab/d3d2dflowfm/`,
- `.../trunk/matlab/applications/delft3d/nest_matlab/dflowfm_io/`.

for Delft3D I/O, conversion and D-Flow FM I/O, respectively.

4 Workflow

Before launching the converter, make sure you have run the `oetsettings`-script to connect with all scripts in the OpenEarth toolbox. Moreover, make sure that the OpenEarth scripts are up-to-date with the latest version within the Tortoise Subversion system. The converter can be launched from the commandline in Matlab through:

```
>> dflowfmConverter
```

Then follow the steps as described below. The window, shown in Figure 1, will appear.

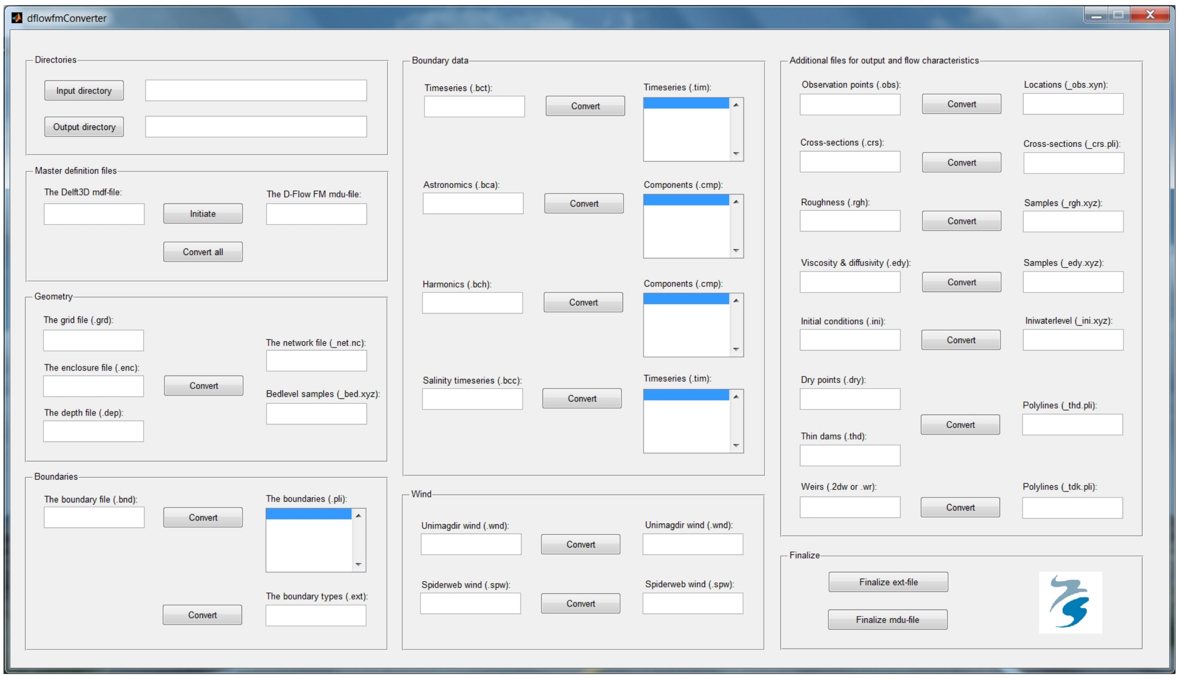


Figure 1: The GUI of the converter as it will appear after launching it in Matlab.

Step 1: Set the directories

The first step is to set the directory that contains the input, i.e. the Delft3D model to be converted. In this directory, a Delft3D `mdf`-file is automatically searched for and displayed in the GUI. The assigned output directory will be use to put all the D-Flow FM model files in.

Step 2: Initiate the session and convert the model in one action

If the directories are properly set and a Delft3D `mdf`-file is found, then the second step is to initialize the converter. By pressing the button *Initiate*, the Delft3D `mdf`-file is read and checked on its keywords. Depending on the existence of several keywords, the remaining fields for the Delft3D input are filled with the specification of the Delft3D-model to be converted. Moreover, the names of the D-Flow FM files that are soon be generated are shown in the D-Flow FM fields of the GUI. One can change these names, if desired.

Now the user comes at a choice. Either the user can choose to press the button *Convert all*, which will automatically convert all relevant files in one single step, or the user can choose to perform the conversion himself just by subsequently following all the conversion parts step by step.

If you choose the option *Convert all*, a waitbar appears that monitors the progress of the overall conversion. In that case, you are done, leaving the following steps irrelevant. If it progresses succesfully, you are done as soon as the converter mentions it has finished. It might happen that an error message pops up, if a certain file appears to be missing in the input directory. Then, the converter stops, leaving you in the position to act adequately by either removing the name of the assigned (but apparently missing) file or by searching the missing file and placing it in the input directory.

Step 3: Convert the grid

An essential difference between Delft3D and D-Flow FM comprises the definition of the computational grid. In Delft3D, a curvilinear grid is defined by indices (m, n) together with an associated array of (x, y)-coordinates. In D-Flow FM, the grid is unstructured, invoking a need to apply an entire different type of grid administration. The change from curvilinear to unstructured is applied when pressing the button *Convert* within the 'Geometry' box.

The bathymetry, provided as a `dep`-file, is administrated within the network file. It is remarked that in Delft3D, the depth is defined positive in negative z-direction, whereas the depth is oppositely defined in D-Flow FM, namely positive in positive z-direction. If no `dep`-file is assigned, then the bathymetry will be provided a value in the D-Flow FM `mdu`-file, through the keyword `Bedlevuni`.

Step 4: Convert the boundary conditions locations

In D-Flow FM, the locations where boundary conditions are to be imposed, are described by means of polyline files: `pli`-files. The next step is 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 resulting `pli`-files (four polylines in case of Figure 2) are stored separately and listed in the listbox. For the constituents, separate but identical polyline files are stored with the addition of the characters `_sal` in the file names. The file name of the polyline files correspond to the file name in the `bnd`-file.

After the conversion of the boundary conditions locations, the *type* of the boundary conditions is specified in the D-Flow FM external forcing file. It is remarked that 'discharge-per-cell'-type boundary and the Riemann-boundary are not supported by the converter yet. In principle, the 'discharge-per-cell' demands for a polyline for each grid cell for which a special discharge-value is prescribed; this is not yet available.

Step 5: Convert the boundary conditions data

The boundary conditions for the flow simulation are given in `bct`-files (timeseries), `bch`-files (harmonic components) or in `bca`-files (astronomic components). Their D-Flow FM equivalents are `tim`-files (timeseries) and `cmp`-files (components), respectively. 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.

Once the button *Convert* is pressed on, 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. This

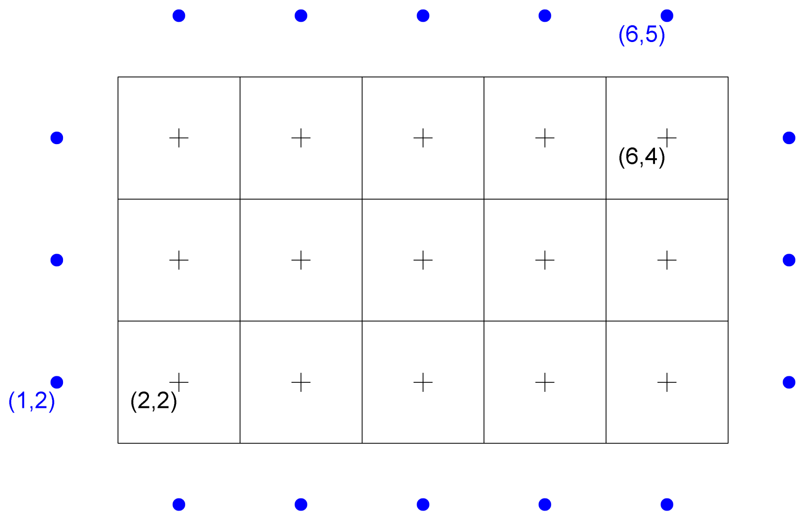


Figure 2: Definition of the grid in Delft3D, illustrated by a Cartesian grid with 5 $\times$ 3 cells. The corners are tabulated in the `grd`*-file, the + markers denote the cell centers, the blue dots denote the mirrored cell centers at the boundaries, referred to in the* `bnd`*-file. The blue dots span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

conversion of data can be a costly activity in case of extensive data files.

Regarding constituents, the conversion tool is currently only suitable to process salinity data. For instance, temperature data conversion is not supported yet. 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 *Convert* is quite comparable with the conversion of hydraulic data. Again, agreement in boundary name specification is searched for. This conversion only works for polylines that are created using this conversion tool.

Regarding discharge boundaries and velocity boundaries, one should be aware of the fact that the *sign* of the boundary condition values is directly converted as well, irrespective of the orientation of the quantity with respect to the grid definition. Always check the sign of discharges and velocities yourself. We currently work on this issue.

Step 6: Transfer the wind data

Regaring wind, one could better speak of a *transfer* of data instead of a *conversion* of data. Wind files are just copied from the input directory to the output directory, since their format remains the same. Currently, only unimagdir-wind (i.e. timeseries with velocity magnitudes and directions) and spiderweb-wind (i.e. cyclone wind) are supported.

Step 7: Convert additional files for output and flow characteristics

The next step is to convert relevant additional model files. The converter currently supports the following data files:

- Observation points: whereas in Delft3D the observation points are defined by their (m, n)-indices, D-Flow FM needs (x, y)-coordinates to specify these points. Hence, a `grd`-file is necessary for the conversion. The result of the conversion is a sample file.
- Cross-sections: whereas in Delft3D the cross-sections are defined by their (m, n)-indices, D-Flow FM needs (x, y)-coordinates to specify the cross-sections. Hence, a `grd`-file is necessary for the conversion. The result of the conversion is a polyline file.
- Roughness: the roughness can be specified in a separate `rgh`, given a certain a grid specified as `grd`-file. The specified `rgh`-file can be converted to D-Flow FM, with a sample file as output.
- Viscosity and diffusivity: analogue to the conversion of the roughness.
- Initial conditions: analogue to the conversion of the roughness. Only waterlevels are currently supported as possible initial condition.
- Dry points and/or thin dams: the conversion of thin dams (i.e. infinitely thin walls that obstruct the flow) comprises a translation of (m, n)-indices to (x, y)-coordinates in a polyline file. Dry points (i.e. Delft3D grid cells that will never contain water) are represented in D-Flow FM by means of 4 thin dams, located at the faces of the cell under consideration.
- Weirs: comparable to thin dams, weirs are represented within D-Flow FM as polylines. Weir files could either have the extension `2dw` or `wr`.

Step 8: Finalize the model

D-Flow FM runs on a master definition file: a `mdu`-file. As much as possible, the Delft3D-settings from the associated `mdf`-file are transferred to equivalent keywords in the D-Flow FM `mdu`-file. Recall that the converter only supports the conversion of two-dimensional flow models, currently.

An important keyword in the `mdu`-file comprises the external forcing file, with extension `ext`. Already after the conversion of the boundary conditions locations, the start of the external forcing file is made. As soon as the additional files (mentioned under the previous step) are converted, these could be coupled to the model by representation in the external forcing file.

Hence, the final step comprises finalizing the external forcing file (i.e. adding the specification of the attribute files of the previous step to it) and then converting the Delft3D master definition file into the D-Flow FM master definition file.

5 Example

As an example, a Delft3D-model is present within the directory:

```
.../trunk/matlab/applications/delft3d/conversion/test/
```

The model comprises a two-dimensional flow model for the Westernscheldt estuary. At the sea-side, an astromical waterlevel signal and a constant non-zero salinity are imposed, whereas at the river-side a discharge is imposed of fully fresh water.

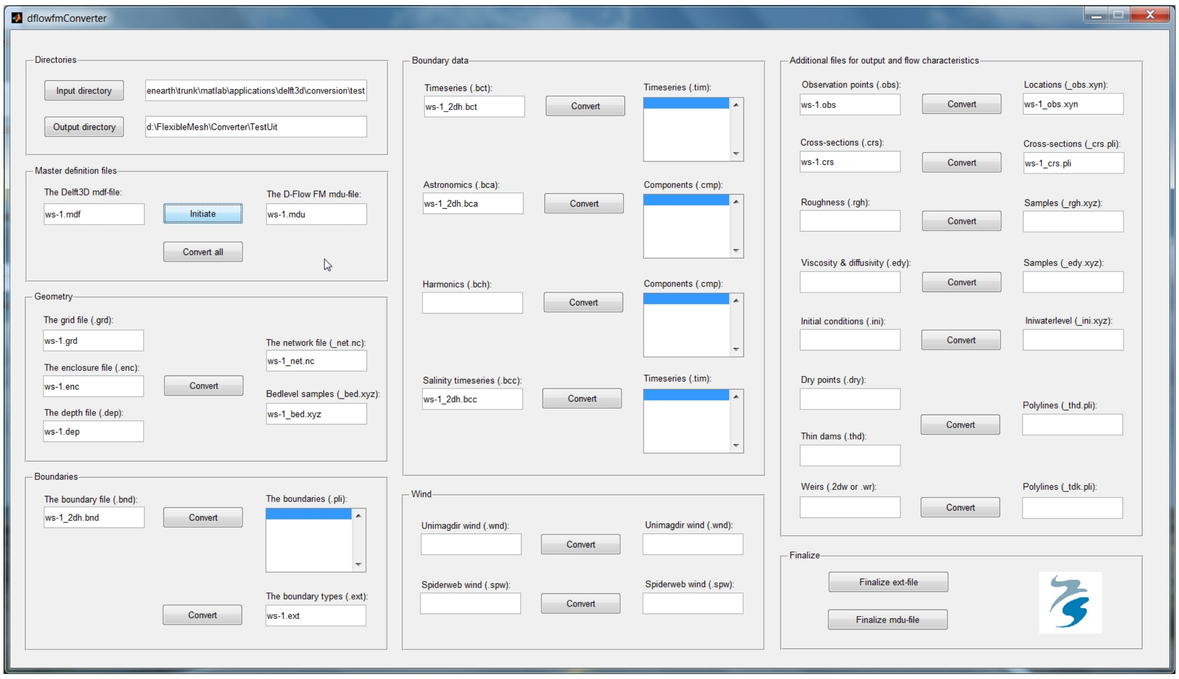
In order to convert the entire flow, the user of the converter needs only to follow two steps: first, set the appropriate directories (for input and output); second, initiate the model, which will set default names for all the file names relevant for the model. Finally, the button *Convert all* can be pressed.

The result of **Step 1** is straightforward. The button *Input directory* enables browsing to the directory in which the Delft3D model is located. The button *Output directory* enables setting the directory in which the D-Flow FM model files will be put in.

The user interface consist of the columns, containing Delft3D input on the left side and D-Flow FM output on the right side. After having set the appropriate input directory, the Delft3D `mdf`-file will appear.

The next step, **Step 2**, comprises the initiation of the filenames of the model under consideration. As a part of this step, the keywords within the `mdf`-file are checked whether these contain links to other files (for instance, the `bnd`-file). The relevant files are displayed on the left side of each of the three columns within the GUI. Within the same step, the supposed D-Flow FM equivalents are provided default names based on the name of the Delft3D `mdf`-file.

The result of Step 2, is shown in Figure 3. In this figure, it is shown that the result of the initiation of the conversion delivers the insight that only a `bct`-file and a `bca`-file are present for the boundary conditions, as well as an observation points file and a cross-sections file for output monitoring.

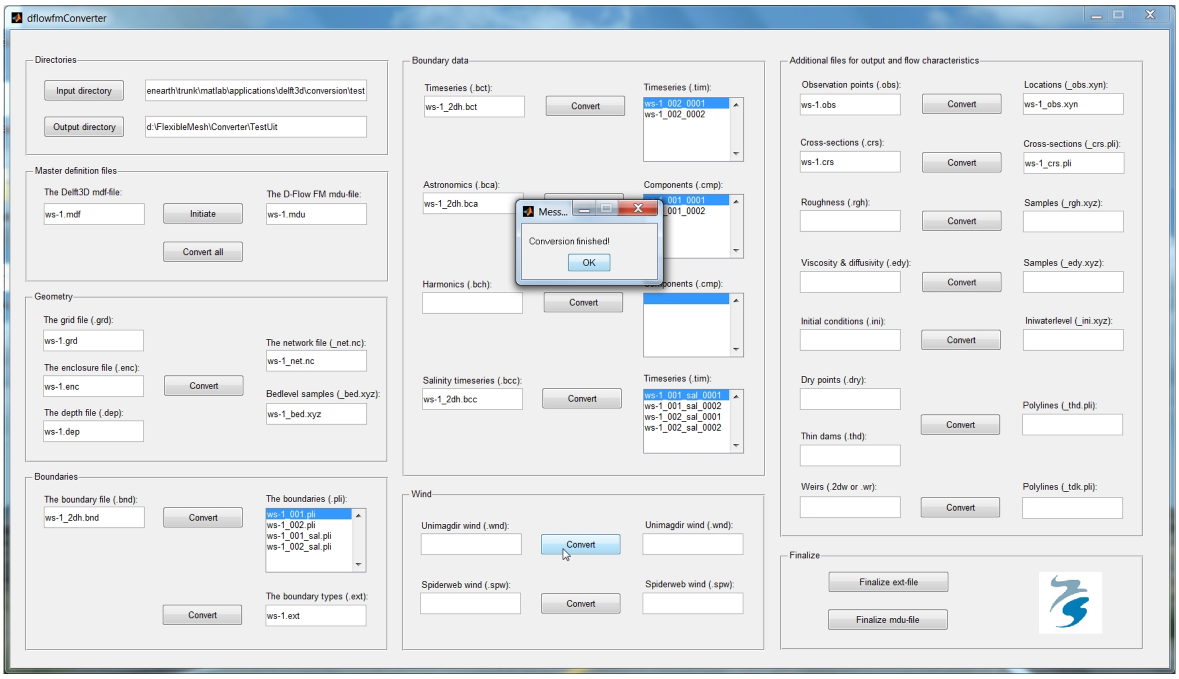


Input directory: enearth\trunk\matlab\applications\delft3d\conversion\test
Output directory: d:\FlexibleMesh\Converter\TestUit
The Delft3D mdf-file: ws-1.mdf
The D-Flow FM mdu-file: ws-1.mdu
The grid file (.grd): ws-1.grd
The enclosure file (.enc): ws-1.enc
The depth file (.dep): ws-1.dep
The network file (_net.nc): ws-1_net.nc
Bedlevel samples (_bed.xyz): ws-1_bed.xyz
The boundary file (.bnd): ws-1_2dh.bnd
The boundary types (.ext): ws-1.ext
Timeseries (.bct): ws-1_2dh.bct
Astronomics (.bca): ws-1_2dh.bca
Salinity timeseries (.bcc): ws-1_2dh.bcc
Observation points (.obs): ws-1.obs
Locations (_obs.xyn): ws-1_obs.xyn
Cross-sections (.crs): ws-1.crs
Cross-sections (_crs.pli): ws-1_crs.pli

Figure 3: Converter screen after having initiated the conversion. The GUI contains of three columns. On the left side of each column, the Delft3D files are provided. On the right side of each column, the D-Flow FM files should be provided.

Once all the relevant files (both for input and for output) are specified, the actual data can finally be converted. Through pressing the button *Convert all*, all the present files are converted in batch. In case some file appears to be missing, the conversion is halted while showing an error message.

Besides converting the entire model in one single step, through *Convert all*, it is also possible to convert only parts of the model. If one, for instance, would like to convert only the observation points file, the one should specify the grid file and the observation points file from Delft3D and the name of the observation points file for D-Flow FM.



Input directory: enearth\trunk\matlab\applications\delft3d\conversion\test
Output directory: d:\FlexibleMesh\Converter\TestUit

The Delft3D mdf-file: ws-1.mdf
The D-Flow FM mdu-file: ws-1.mdu

The grid file (.grd): ws-1.grd
The enclosure file (.enc): ws-1.enc
The depth file (.dep): ws-1.dep
The network file (_net.nc): ws-1_net.nc
Bedlevel samples (_bed.xyz): ws-1_bed.xyz

The boundary file (.bnd): ws-1_2dh.bnd
The boundaries (.pli): ws-1_001.pli, ws-1_002.pli, ws-1_001_sal.pli, ws-1_002_sal.pli
The boundary types (.ext): ws-1.ext

Timeseries (.bct): ws-1_2dh.bct
Timeseries (.tim): ws-1_002_0001, ws-1_002_0002
Astronomics (.bca): ws-1_2dh.bca
Components (.cmp): ws-1_001_0001, ws-1_001_0002
Harmonics (.bch):
Components (.cmp):
Salinity timeseries (.bcc): ws-1_2dh.bcc
Timeseries (.tim): ws-1_001_sal_0001, ws-1_001_sal_0002, ws-1_002_sal_0001, ws-1_002_sal_0002

Conversion finished!

Unimagdir wind (.wnd):
Unimagdir wind (.wnd):
Spiderweb wind (.spw):
Spiderweb wind (.spw):

Observation points (.obs): ws-1.obs
Locations (_obs.xyn): ws-1_obs.xyn
Cross-sections (.crs): ws-1.crs
Cross-sections (_crs.pli): ws-1_crs.pli
Roughness (.rgh):
Samples (_rgh.xyz):
Viscosity & diffusivity (.edy):
Samples (_edy.xyz):
Initial conditions (.ini):
Iniwaterlevel (_ini.xyz):
Dry points (.dry):
Thin dams (.thd):
Polylines (_thd.pli):
Weirs (.2dw or .wr):
Polylines (_tdk.pli):

Figure 4: Converter screen after having converted all the specified data files. In the listboxes, the names of the generated polylines are provided.

6 Common pitfalls

It is very important to always check the outcomes of the converter yourself. Particularly, some attention should be paid to the definition of the boundary conditions. In this section, two aspects regarding the boundary conditions are addressed, namely in case the boundaries are curved and in case the cells at the polyline ends are not properly initialized.

The first aspect comprises boundary curvature. As an example, the standard Frisian Inlet tutorial case from the Delft3D manual is considered. In Delft3D, a boundary condition is specified along the northern boundary, relying on two grid cells (i.e. ghost cells): one at the western end and another at the eastern end. A direct conversion of the boundary condition location results in the picture, shown in the left panel of Figure 5. This undesired result can be resolved by means of manually inserting additional polyline points along the boundary, such that the boundary curvatures is better followed. The result is illustrated in the right panel of Figure 5.

Another known deficiency of the converter has to do with the definition of the polyline for the boundary conditions near grid corners. Delft3D uses the ghost cells of boundary cells for the prescription of boundary conditions. The cell centers of these ghost points are used by the converter to prescribe the polyline for the boundary conditions. In the initialization phase of the D-Flow FM computation, D-Flow FM searches for such boundary polylines. It could happen that the prescription of the boundary polyline is too tight for the computational engine to identify a computational cell to be a boundary cell.

As an illustration, the above converted Westernscheldt is used: see Figure 6. The left panel of Figure 6 shows the result of the direct conversion of the model. An desired effect of the conversion

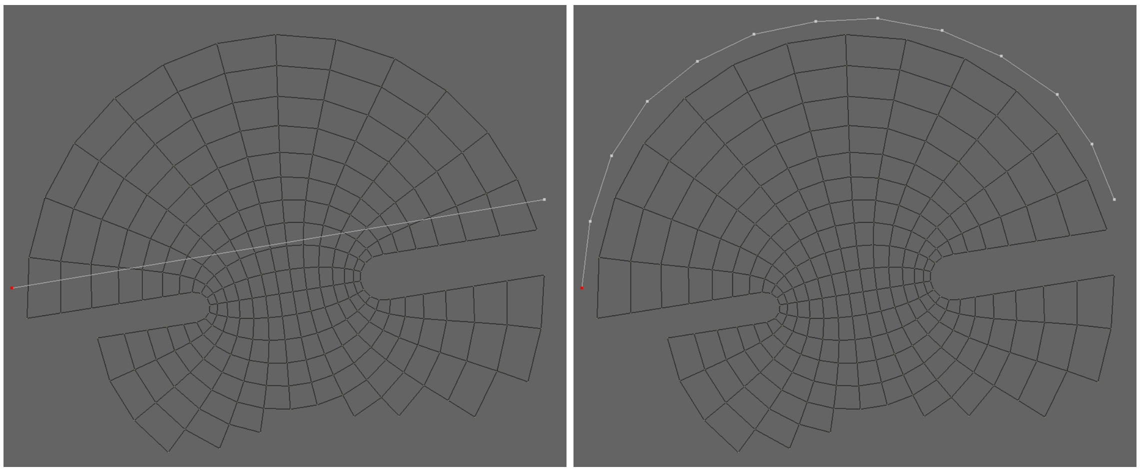


Figure 5: Always check the boundary polyline after conversion. Left panel: if the Delft3D boundary specification relies on two grid cells, a straight polyline is present straight through the domain. Right panel: the issue can be circumvented through manually inserting additional polyline points along the boundary.

is that the corner cell is not identified as a boundary cell in the sense that boundary conditions are imposed at the location of this cell.

This issue can be solved by slightly replacing the end point of the polyline to the favored side. The right panel of Figure 6 shows that replacement somewhat to the left, leads to the proper initialization of the boundary cells. In this case, the corner cell is indeed identified as a computational cell that is affected by the boundary conditions.

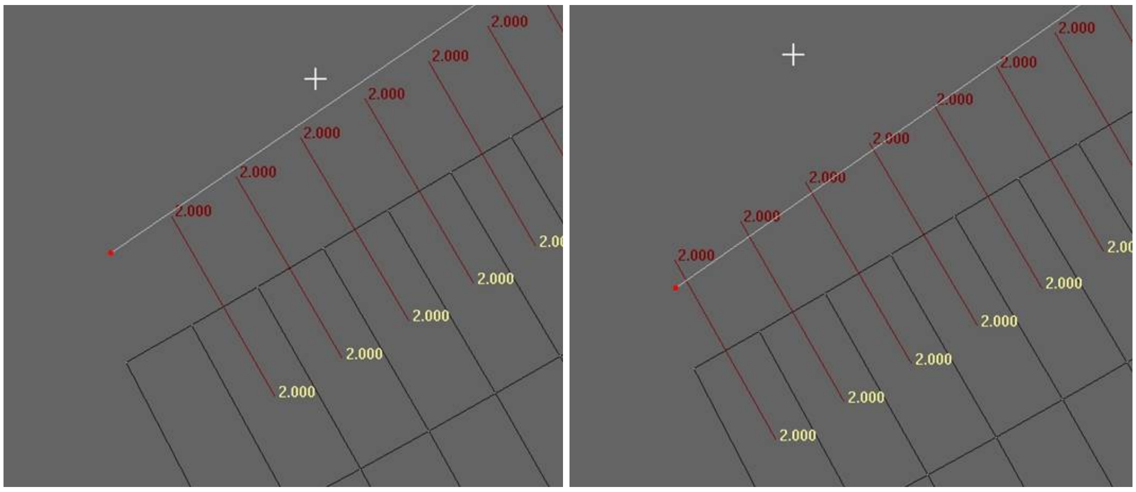


Figure 6: Always check if the positioning of the polyline leads to the proper initialization of the D-Flow FM model. Left panel: the initialization does not identify the corner cell as a boundary cell. Right panel: proper initialization of the D-Flow FM after replacing the polyline end point.

As these two examples illustrate, it is more than worthwhile to manually check the outcomes of the converter. The converter helps you with a quick start, but finalizing requires a thorough comparison.

7 Copyright and license information

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.