DHI Flexible File Formats

DFSU 2D/3D, Vertical Profile/Column, and Mesh File

Technical Documentation

DHI A/S headquarters
Agern Allé 5
DK-2970 Hørsholm
Denmark

+45 4516 9200 Telephone
mike@dhigroup.com
www.mikepoweredbydhi.com

Company Registration No.: DK36466871

CONTENTS

DHI Flexible File Formats
FSU 2D/3D, Vertical Profile/Column, and Mesh File
Technical Documentation

APPENDICES

1   Introduction

This document presents the technical description, terms and definitions of the flexible mesh data structure. It also describes the DFSU file format and the mesh file format in general, and what restrictions current DHI software put on these formats.

A general flexible mesh is based on a node-element structure with values defined either as element average/element centre values, or on the nodes. Elements can have many different forms in general.

Note that indices in this document and in the flexible mesh files are one-based, as is the natural base in Fortran, in contrast to the natural base in C/C++/C# and alike which is zero based.

Current usage of the mesh file format

The mesh file format supports a 2D mesh, consisting of triangles and/or quadrilaterals.

Current usage of the DFSU file format

In general, the DFSU file format supports 3 spatial variations:

1.   a 2D version where Z coordinates refers to the bathymetry depths,
2.   a 3D layered version, where the number of layers may vary, and
3.   a 2D vertical version, which is a vertical slice through a 3D file.

In the 2 horizontal dimensions the elements can be triangles, quadrilateral, and a mix of the two.

The DFSU files stores 3 types of data. The default is to store values on nodes and/or elements. A variant stores values on the faces between elements. There is also a spectral variant that on each node or element stores values for a number of frequencies and directions.

2 Flexible Mesh Terms and Definitions

A flexible mesh consists of a number of nodes and a number of elements. The nodes and elements are defined in each their own table.

The mesh used in this chapter can be found as an example of a mesh file at the end of this document.

2.1 Node Information

The node information contains the node ID's, coordinates and the boundary code. Example:

```
Index   Id    X       Y       Z    code
 1       1  0.464   0.418   -1.00   1
 2       3  0.469   0.639   -2.00   1
 3      45  0.624   0.354   -1.00   1
 4       5  0.666   0.548   -4.13   0
 5       2  0.659   0.697   -1.81   1
 6     210  0.810   0.442   -3.00   0
 7      18  0.801   0.283   -1.00   1
 8       4  0.874   0.603   -6.00   0
 9     399  0.811   0.777   -2.77   1
10      12  0.993   0.436   -2.79   1
11      26  1.060   0.757   -2.00   1
12      32  1.116   0.573   -4.50   0
```

The index is 1-incrementing. Note that the index and the ID in general are not alike, and the range of the ID's in general is not the same as the range of the indices. The ID must be unique, i.e., two nodes cannot have the same ID.

The code is used for the boundary; see Section 2.4 for more details.
Nodes are also sometimes referred to as vertices.

2.2 Element Information

The element information contains the element ID, the element type and the element-node connectivity, also called the element table.

```
Index   Id   Type   Element-table
 1       4    25     11   8  10  12
 2       8    21      9   8  11
 3      12    21     10   8   6
 4       3    21      6   7  10
 5      45    21      6   8   4
 6       6    25      4   8   9   5
 7     321    25      7   6   4   3
 8      26    25      2   1   3   4
```

```
         9        5    21      4    5    2
```

Note that the index and the ID in general are not alike, and the range of the IDs and the indices is in general not the same. The IDs must be unique.

The element table specify which nodes that each element uses. Each element can have an individual number of node indices. The type specifies how the element uses the nodes to define its geometry. Hence, the first element has ID 4, type 25 means a quadrilateral element, and is using nodes with indices 11, 8, 10 and 12. Note that the element table references the nodes in the Node table using the node index, not the ID.

Elements are also sometimes referred to as cells. Element type is also often referred to as element code.

The type 21 is a triangle, 25 is a quadrilateral. See Appendix A "Element Types" for definitions of element types. For type 21 and 25 the nodes of the elements must be specified in counter clockwise order in the element-node table.

It is not allowed to have nodes that are not referenced in the element table. Those nodes should be deleted, as they are not considered part of the mesh.

2.3   Faces

A face defines the boundary between two elements, or between an element and the outside of the mesh.

A face in a 2D mesh is a line, while in 3D a face is a 2D geometry, often a triangle or a quadrilateral, depending on the element type.

The example element table in previous section define element number two using node 9, 8 and 11, and this element has three faces: The first face is the line between node 9 and 8, face 2 is the line between node 8 and 11, and face 3 is the line between node 11 and 9. How the faces are defined depends on the element type, see Appendix A "Element Types" for details.

A face on the boundary of the mesh is called a boundary face. Boundary faces in 2D meshes have a direction, which is defined to be positive when walking along the face with the left hand side inside the mesh and the right hand side outside the mesh. On a circular mesh this corresponds to walking at the mesh boundary in counter clockwise direction.

2.4   Boundary Code Values

Each node has a boundary code. For internal nodes this is zero. For nodes on the boundary, this must be larger than zero. For land boundaries, the code must be one. All other boundary nodes can be given a user specified boundary code, which must be larger than 1.

In a 2D mesh each face on the boundary is also given a boundary code. This boundary face code is calculated from the node codes. Engines set boundary conditions based on the boundary code values of the faces and not the nodes.

To determine the boundary face code from the node codes, assuming a boundary face is oriented in positive direction from node A to node B:

1. In case any of the nodes A and B is a land node (code value 1) then the boundary face is a land face, given boundary code value 1.

2. If both nodes A and B have code values larger than 1, the face code is the boundary code value of node B (the last node).

The figure below shows boundary codes on nodes (marked by coloured circles) and faces (marked by coloured line) in a typical boundary setup.
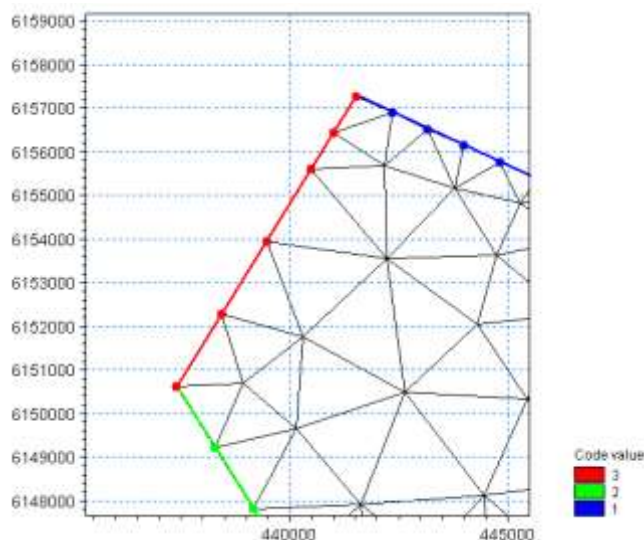
Note the label with the code values. Also note how the node code values extend to the two faces that have nodes with different code values: The land code value is extended to the first node with code value 3 due to the first rule, and the code value 2 is extended to the last node with code value 3 due to the second rule.

If adding an extra node on a boundary face, the code of the new node must match the face code. If collapsing a boundary face, the newly created centre node code value is given the code value of the original face first node.

In a 3D layered mesh each face is given the boundary code value of the 2D horizontal mesh face in its column.

3 The Mesh File Specification

The mesh file has the .mesh extension. It is a text/plain file. It basically lists the node and a simplified element table in text format.

The file uses newline and spaces as separators. Several consecutive space separators count as one separator. A line can begin with a number of spaces and these must be ignored. Several consecutive newline separators are not allowed.

New lines: DOS text files traditionally have carriage return and line feed pairs as their newline characters while unix text files have only the line feed as their newline. The mesh file is valid in any case.

The mesh file has the following form

Header line: One line containing item type integer, item unit integer, the number of nodes and the projection string.

```
100079 1000 12 LONG/LAT
```

The item type/unit integer defines the quantity stored in the Z coordinate of the mesh file. The item type is always "Bathymetry" (`eumIBathymetry = 100079`), while the unit can be any of the allowed unit for the Bathymetry item type. A value of 1000 tells that Z values are specified in meters.

The projection string can be quite long. Currently the longest projection string is more than 700 characters long, and it may grow even bigger. The projection string can contain spaces. An example of a UTM-33 projection string, where the line has been broken a number of times in order to fit on this page:

```
PROJCS["UTM-33",GEOGCS["Unused",DATUM["UTM Projections",SPHEROID["WGS
1984",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["Degree",0.017453292
5199433]],PROJECTION["Transverse_Mercator"],PARAMETER["False_Easting",500000]
,PARAMETER["False_Northing",0],PARAMETER["Central_Meridian",15],PARAMETER["Sc
ale_Factor",0.9996],PARAMETER["Latitude_Of_Origin",0],UNIT["Meter",1]]
```

Node lines: As many lines as stated in the header line. Each line contains information for one node on the form

```
Id X Y Z code
```

The Id must be strictly positive. The code is zero for internal nodes, 1 for a node on land boundaries, and larger than one for all other boundaries. Example:

```
 1  0.464  0.418  -1.00  1
 2  0.469  0.639  -2.00  1
 3  0.624  0.354  -1.00  1
 4  0.666  0.548  -4.13  0
 5  0.659  0.697  -1.81  1
 6  0.810  0.442  -3.00  0
 7  0.801  0.283  -1.00  1
 8  0.874  0.603  -6.00  0
 9  0.811  0.777  -2.77  1
10  0.993  0.436  -2.79  1
11  1.060  0.757  -2.00  1
12  1.116  0.573  -4.50  0
```

Element header line: One line containing 3 numbers: The number of elements, the maximum number of nodes per element, and the element type/code. Example

```
9 4 25
```

Element lines: As many lines as stated in the element header line. Each line contains information for one element on the form:

   Id node1 node2 node3 etc...

Each line contains at most as many nodes as stated in the element header line. If an element has fewer nodes than the maximum number of nodes per element stated in the element header line, the remaining node values are zero. Example

```
1 11  8 10 12
2  9  8 11  0
3 10  8  6  0
4  6  7 10  0
5  6  8  4  0
6  4  8  9  5
7  7  6  4  3
8  2  1  3  4
9  4  5  2  0
```

Each element type defines how to handle if a node is missing. See Appendix B for an example of a mesh file.

The current version of MIKE Zero supports the following element types in a mesh file:

• 21: The mesh only contains triangular elements. Number of nodes per element must be 3.

• 25: The mesh contains mixed triangular/quadrilateral elements, or pure quadrilaterals. Number of nodes per element must be 4. If only 3 nodes are defined for an element, it is a triangle.

Presently the mesh file cannot handle more than one element type. Hence element type 25 covers elements using as well 3 as 4 nodes, and degenerates to a triangle, element type 21, when having 3 nodes.

4 The DFSU File Specification

The dfsu file has the .dfsu extension. It is a binary file based on the dfs (data file system) format. The u in dfsu is short for unstructured.

The exact binary format of each section of the dfs will not be described here, see the DFS User Guide for details. The information here will refer to which sections of the dfs format that are used, and what data is put in each section.

The dfsu files exists in 3 types

- Standard type, storing values on nodes and/or elements. This is the default type from most of the engines.

- Face value type, storing values on element faces. This is used e.g. for HD decoupling files, to store the discharge between elements.

- Spectral value type, for each node or element, storing vales for a number of frequencies and/or directions. This is the file type for spectral output from the spectral engines.

The following spatial variations of the dfsu file format are currently used and supported:

- 2D horizontal. This is the default 2D output file.

- 3D layered. For version 2008 and back, only an equal number of layers everywhere are supported. From version 2009 a varying number of layers have been introduced.

- 2D vertical profile. This corresponds to a vertical slice through a 3D layered file.

- 1D vertical column. This corresponds to a vertical dfs1 file and is produced by taking out one column of a 3D layered file.

- 1D FEM (1D BW)

- 3D/4D SW, two horizontal dimensions and 1-2 spectral dimensions.

The following dfs sections will be described here

- Data type
- Custom block
- Static items
- Dynamic items.

When creating a new file, also the temporal axis needs to be specified. Currently only the equidistant calendar type time axis is compatible with MIKE Zero.

4.1 Layered Files

Layered files are files with a vertical extent which are structured in the vertical direction.

There are certain restrictions on how layered files are to be defined, so that a user program is able to recognise each vertical column.

4.1.2   3D layered file

The 3D layered grids are 2D grids that are extended in the third dimension to a number of layers. The number of layers for each element in the 2D grid can vary.

Nodes on top of each other (the same column/position in 2D) must have consecutive indices, starting from the node with the smallest Z (the bottom).

Elements on top of each other must have consecutive indices, starting from the element with the smallest Z.

Consider the figure, showing 3 layers elements:
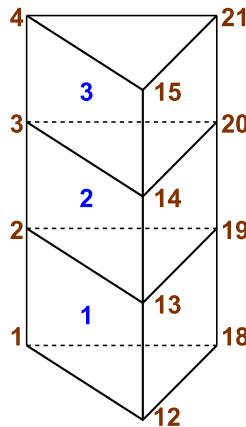
The element table for a mesh of prismatic elements will consist of 6 nodes and for hexahedra of 8 nodes. Example from the figure above:

```
Index  Id   Type   Element-node-indices
1      1    32    1   12  18    2   13  19
2      2    32    2   13  19    3   14  20
3      3    32    3   14  20    4   15  21
```

For each element the bottom nodes are listed first, and then the top nodes following, in the same order, as defined by the element type.

To check when two consecutive elements are in the same column, the last half of the nodes in the lower element must match exactly in order the first half of the nodes in the element above it. This is the restriction allowing an easy way to check when two elements are on top of each other.

4.1.3   2D vertical profile

The 2D vertical profiles follow the rules from the 3D layered file, with the exception of the element table definition.

A 2D vertical layered element is a quadrilateral, consisting of 4 nodes. The nodes in the element table must be defined counter clockwise, starting from the lower left corner.

To check when two consecutive elements are in the same column, the last two nodes of the first element must match the first two nodes of the second element in reverse order. Example with 3 elements on top of each other:
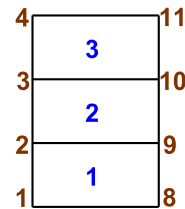

```
Index  Id  Type  Element-node-indices
1      1   25    1   8   9   2
2      2   25    2   9   10  3
3      3   25    3   10  11  4
```

4.2   DFS Data Type

The dfs data type for a dfsu file is 2000, 2001, 2002, 2003 or 2004. It indicates the type of values stored in the file.

2000: Node values are saved in the dynamic items. The MIKE software does not currently support files of data type 2000.

2001: Cell values are saved in the dynamic items. This is the most common output.

2002 and 2003 indicate that the file contains spectral information. The output depends on the spatial extend and the number of frequency and/or the number of directions. 2002 stores data on nodes and 2003 stores data on elements.

2004: Face values are stored in the dynamic item, indicating value at a face, for example discharge over the face.

4.3   Custom Block

All dfsu file versions can be distinguished from other dfs file versions by the content of the custom blocks.

The dfsu file uses one custom block which is named

```
MIKE_FM
```

The custom block for data type 2000 and 2001 contains 4 or 5 integer values. Files from version 2008 and before only have 4 values, while files from 2009 and forward have 5 values.

1. Number of nodes
2. Number of elements
3. Dimension
4. Maximum number of layers
5. Number of sigma layers

When reading a file, it must be checked if the size of the custom block is 4 or 5. If the size is 4, then maximum number of layers equals number of sigma layers.

The number of nodes and elements must match those of the element and node table.

For the different flavours currently supported by the DHI tools, the dimension (3:), maximum number of layers (4:) and the number of sigma layers (5:) are used as follows:

File type		Value
2D Horizontal	3:	2
	4:	0
	5:	0
3D layered, constant number of layers	3:	3
	4:	Number of layers
	5:	Must equal value in 4.
3D layered, varying number of layers	3:	3
	4:	Max number of layers
	5:	Number of sigma layers, less than value in 4
2D vertical profile constant number of layers	3:	2
	4:	Number of layers
	5:	Must equal value in 4.
2D vertical profile, varying number of layers	3:	2
	4:	Max number of layers
	5:	Number of sigma layers, less than value in 4
1D vertical column	3:	1
	4:	Number of layers
	5:	Number of sigma layers, can differ from value in 4.

The custom block for data type 2002 and 2003 contains 6 integer values

1. Number of nodes
2. Number of elements
3. Dimension
4. Not used
5. Number of frequencies
6. Number of directions

The custom block for data type 2004 contains 6 integer values

1. Number of nodes
2. Number of elements
3. Number of faces
4. Dimension
5. Maximum number of layers
6. Number of sigma layers

4.4 Static Items

The block of static items contains the mesh information.

For data type 2000 and 2001 the dfsu file has the following 9 static items.

Static item name	Description	Type	Size of item data
Node id	Node id's	Integer	#nodes
X-coord	X coordinates of nodes	Double (Float)	#nodes
Y-coord	Y coordinates of nodes	Double (Float)	#nodes
Z-coord	Z coordinates of nodes	Float	#nodes
Code	Node boundary codes	Integer	#nodes
Element id	Element id's	Integer	#elements
Element type	Element type	Integer	#elements
No of nodes	#nodes in each element	Integer	#elements
Connectivity	Indices of nodes in each element	Integer	Sum of all "No of nodes"

The (Float) type in parenthesis indicates that there exist older files storing x and y coordinates in single precision.

For a file with a vertical dimension the node Z coordinates are the initial coordinates and will generally not match the dynamic item Z coordinates. For these files, the Z node coordinates here should not be used, use instead those in the dynamic item.

Values in the "# nodes in elements" must match the "element code" definition, which specifies the type of each element.

The unit of the x and y coordinates are specified in the static item EUM system. For projected coordinate systems this is a length unit, and for geographical coordinate systems it is lon/lat degrees, i.e. with EUM item type and unit respectively:

• Projected EUM item/unit: `eumIGeographicalCoordinate`, lengt unit
• Geographical EUM item/unit: `eumILatLong`, `eumUdegree`

The unit of the z coordinate is similarly specified in the static item EUM system

- EUM item/unit: `eumIItemGeometry3D`, lengt unit

The EUM item type for the z coordinate is `eumIItemGeometry3D` and the unit can be set to any length unit supported by the EUM item type `eumIItemGeometry3D`.

The remainder of the static items all have EUM item `eumIIntegerCode` and unit `eumIintCode`.

For older files the EUM values are not defined for the static items. Then the x and y coordinates are in meters or degrees, depending on the projection string, and z is always in meters.

The spatial axis is irrelevant for all items and set as follows:

```
AxisType: F_EQ_AXIS_D1
XMin: 0
DX: 1
NoDim: 1
```

All the static items have been introduced in previous parts. In a mesh file the overall code is specified in the element header line. In a dfsu file, the code is given element by element.

The following values are currently used:
- 21: Triangle in 2D, using 3 nodes.
- 25: Quadrilateral in 2D, using 4 nodes
- 32: Prisms (3D triangle), using 6 nodes
- 33: hexahedra/cubes (3D rectangle), using 8 nodes.

There is no information on the layer number in the file. The user application must find layer numbers themselves by, e.g., checking the element node table: The last half of the table for one element matches the first part of the element above it. Also, elements on top of each other have consecutive indices. When considering an element you just need to check whether the next element is on top of it, otherwise there are not any elements on top of it.

For data type 2002 and 2003 the dfsu file has 10 or 11 static items. The first 9 items are identical to the items for data type 2000 and 2001. When the file contains spectral information, the number of frequencies and the number of directions are required, and one or both of the following 2 static items are present

Static item name	Description	Type	Size of item data
Frequency	Frequency values in /s	Integer	#frequencies
Direction	Direction values in degrees	Integer	#directions

If the file contains spectral information which depends only on the number of frequencies, or only on the number of directions, only the corresponding static item is present.

For data type 2004 the dfsu file has 12 static items. The first 9 items are identical to the items for data type 2000 and 2001. The following 3 additional items are also included.

Static item name	Description	Type	Size of item data
Face id	Face id's	Integer	#faces
No of faces	#nodes in each face	Integer	#faces
Connectivity	Indices of nodes in each face	Integer	Sum of all "No of faces"

4.5   Dynamic Items

A dfsu file can have any number of dynamic items. They are usually of type float.

For files with a vertical dimension, the first item is a dynamic Z node coordinate item. It is specified as follows:

```
Item name: Z coordinate
Item EUM type: Item geometry 3-dimensional
Item EUM unit: meter
```

Its size is #nodes. It is the only dynamic item with the size of #nodes.

The sizes of the remaining dynamic items depend on the data type.

- For data type 2000 and 2001 the dynamic items lengths are #elements.

- For data type 2002 the items lengths are
  #data values = #nodes x #frequencies x #directions

- For data type 2003 the dynamic items lengths are
  #data values = #elements x #frequencies x #directions

- For data type 2004 the dynamic items lengths are #faces.

Vector items are stored as 2 or 3 ordinary items. The user is responsible for recognising the items as a vector.

EUM item types and units are user specified. A unit must match the type, if not, the behaviour is undefined. (Currently the dfs routines fail when reading such a file).

The spatial axis is irrelevant for all items and set as follows

```
AxisType: F_EQ_AXIS_D1
XMin: 0
DX: 1
NoDim: 1
```

APPENDICES

A   Element Types

The following list all element types (could be supplied with a small figure of each element type)

1D:

- 11: (Finite element)
- 12: (Finite element)

2D:

- 21: Triangular linear element
- 22: (Finite element)
- 23: (Finite element)
- 24: (Quadrilateral linear element)
- 25: Quadrilateral elements.
- 26: (Finite element)
- 28: (Finite element)

3D:

- 32: Triangular prism/wedge (triangle layered)
- 33: Regular hexahedron (quadrilateral layered)

B   Mesh File Examples

This is the example used in the text. It contains a mix of triangles and quadrilaterals.

```
100079 1000 12   LONG/LAT
 1   0.464   0.418   -1.00   1
 2   0.469   0.639   -2.00   1
 3   0.624   0.354   -1.00   1
 4   0.666   0.548   -4.13   0
 5   0.659   0.697   -1.81   1
 6   0.810   0.442   -3.00   0
 7   0.801   0.283   -1.00   1
 8   0.874   0.603   -6.00   0
 9   0.811   0.777   -2.77   1
10   0.993   0.436   -2.79   1
11   1.060   0.757   -2.00   1
12   1.116   0.573   -4.50   0
9 4 25
1 11   8 10 12
2   9   8 11   0
3 10   8   6   0
4   6   7 10   0
5   6   8   4   0
6   4   8   9   5
7   7   6   4   3
8   2   1   3   4
9   4   5   2   0
```