Technical university of Liberec

Faculty of mechatronics, informatics
and interdisciplinary studies

FLOW123D

version 1.6.0

Documentation of file formats and brief user manual.

Liberec, 2010

Acknowledgement. This work was realized under the state subsidy of the Czech Republic within the research and development project "Advanced Remediation Technologies and Processes Center" 1M0554 – Program of Research Centers PP2-D01 supported by Ministry of Education.

Flow123D

Introduction

Flow123D is a software for simulation of water flow, solute transport and sorption in a heterogenous porous and fractured medium. In particular it is suited for simulation of underground processes in a granite rock masive. The program is able to describe explicitely processes in 3D medium, 2D fractures, and 1D chanels and exchange between those dimensions. The computational mesh is therefore collection of 3D tetrahedrons, 2D trinagles and 1D line segments.

The water flow model assumes a saturated medium described by Darcy law. For discretization we use mixed hybrid FEM. This version allows only calculation of steady water flow.

The solute transport model can deal with several dissolved substances. It contains non-equilibrium dual porosity model, i.e. exchange between mobile and immobile pores. There is also model for several types of sorption in both the mobile and immobile zone. The imlemented sorption models are linear sorption, Freundlich isotherm and Langmuir isotherm. The solute transport model uses finite volume discretization with upwinding in space and explicit Euler discretization in time. The dual porosity and the sorption are introduced into transport by operator spliting. The dual porosity model use analytic solution and the non-linear adsorption is solved numerically by the Newton method.

The program is implemented in C/C++ using essentialy PETSC library for linear algebra. The water flow as well as the transport simulation can be computed in parallel using MPI environment. This version also support output into VTK format, which is widely supported. In particular we recommend Paraview for visualization and postprocessing of the results.

The program is distributed under GNU GPL v. 3 licence and is available on the project web page: http://dev.nti.tul.cz/trac/flow123d

Usage

On the Linux system the program can be started either directly or through a script `run_flow.sh`. When started directly by command

```
> flow123d -s example.ini
```

the program accepts one argument after swith `-s` which is the name of the principila input file. When you want to start a parallel job you shloud rather use starting script. Basic usage is:

```
> run_flow.sh -np 2 -s example.ini
```

which run simulation on 2 processes using the same INI file as before. For other possible arguments see the begining of the script.

On the Windows system you can start a squential run by command:

```
> flow123d.exe -s example.ini
```

or a parallel run by command:

```
> mpiexec.exe -np 2 flow123d.exe -s example.ini
```

The principial input file of the program is an INI file which contains names of other necesaryinput files. Those are the file with calculation mesh (`*.msh`), the file with specification of neigbourings between dimensions (`*.ngh`), the file with material description (`*.mtr`) and the file with boundary conditions for the water flow problem (`*.bcd`).

In the case of transport simulation one have to specify also the file with transport boundary conditions (`*.tbc`) and the file with transport initial condition for individual substances (`*.tic`).

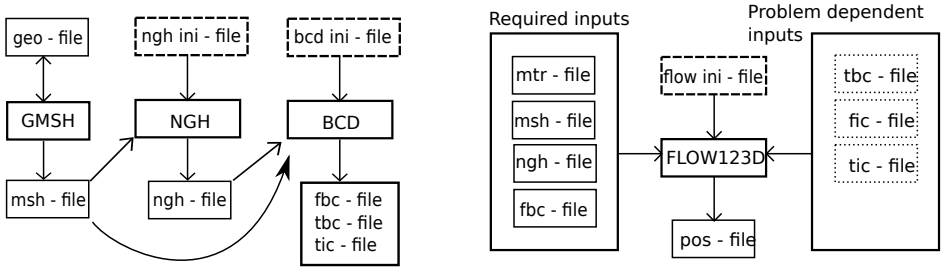

Figure 1: Preparation of input files.

For the preparation of input files we use several utitlities (see Figure 1). We usualy begin with a `*.geo` file as a description of the domain geomery. This come as an input for the GMSH mesh generator, which produce the mesh file. Then we run program `ngh` to produce file of neigbourings. Finally we can use program `bcd` for the preparation of files with boundary and initial conditions. The file with material properties has to be created manualy, preferably by modifying some of the example problems. The programs `ngh` and `bcd` are distributed together with flow123d with their own limited documentation.

The output files can be either `*.pos` files accepted by the GMSH or one can use VTK format that can be postprocessed by Paraview.

In following sections we briefly describe structure of individual input files.

Advection-Diffusion equation

Solute transport is governed by advection equation which can be written in the form

$$\frac{\partial c}{\partial t} + \boldsymbol{v}\frac{\partial c}{\partial x} = 0, \tag{1}$$

where c is concentration $[M^3 \cdot L^{-3}]$, t is time $[T]$, v is velocity $[L \cdot T^{-1}]$, and x is coordinate in cartesian system $[L]$. Assuming solution which is constant on every element (cell centered finite volume method) and integrating equation (1) we get

$$\int\limits_{e_i} \frac{\partial c}{\partial t} dV + \int\limits_{e_i} \boldsymbol{v}\frac{\partial c}{\partial x}\, dV = 0.$$

After some rearrangements we obtain on i-th element (e_i)

$$\frac{\partial c_i}{\partial t} V_i + c \int\limits_{\partial e_i} \boldsymbol{v}\, \mathbf{dS} = 0, \tag{2}$$

where c_i is average concentration in e_i and V_i its volume, c will be specified later (there are two main possibilities - c_i or concentration from neighbouring element). Term $\frac{\partial c}{\partial t}$ we approximate by explicit Euler difference

$$\frac{\partial c}{\partial \mathrm{t}} \approx \frac{c_i^{n+1} - c_i^n}{\Delta t}. \tag{3}$$

Where Δt is a time step and upper index at c_i means values in the discrete time steps $n+1$ and n. We assume that all elements have piecewise smooth element boudary ∂e with outwards directed normal. Inside the area Ω we introduce internal flows. With respect to e_i, we define internal flow intake U_{ij}^- (from element e_j) and internal flow drain U_{ij}^+ (to element e_j) as follows

$$\begin{aligned} U_{ij}^- &= \min(\int\limits_{\partial e_i \cap \partial e_j} \boldsymbol{v}\, \mathbf{dS}, 0), \\ U_{ij}^+ &= \max(\int\limits_{\partial e_i \cap \partial e_j} \boldsymbol{v}\, \mathbf{dS}, 0). \end{aligned} \tag{4}$$

Those flows realizes solute transport in the area Ω. On the $\partial\Omega$ we define external flows which will be important for transport Dirichlet boundary conditions. In the same way as for internal flows we assume (with respect to element e_i) external flow intake U_{ij}^{e-} (from $\partial\Omega$) and external flow drain U_{ij}^{e+} (to $\partial\Omega$).

$$\begin{aligned} U_{ik}^{e-} &= \min(\int\limits_{\partial e_i \cap \partial\Omega} \boldsymbol{v}\, \mathbf{dS}, 0), \\ U_{ik}^{e+} &= \max(\int\limits_{\partial e_i \cap \partial\Omega} \boldsymbol{v}\, \mathbf{dS}, 0). \end{aligned} \tag{5}$$

Direction of the velocity $\boldsymbol{v}$, which affects sign of the U-terms is significant for the construction solution. For the solution stability it is suitable to use an upwind scheme, which can by written for finite difference on simple 1D geometry in the form

$$\begin{aligned} v > 0 : \frac{\partial c}{\partial \mathrm{x}} &\approx \frac{c_i^n - c_{i-1}^n}{\Delta x}, \\ v < 0 : \frac{\partial c}{\partial \mathrm{x}} &\approx \frac{c_{i+1}^n - c_i^n}{\Delta x}. \end{aligned} \tag{6}$$

This scheme can be interpreted as well as in finite volume method - in convection term one can get c value opposite the flow of the quantity $\boldsymbol{v}$ direction. For every e_i we introduce itemsets $\mathcal{N}_i, \mathcal{B}_i$ which contains indexes of neighbourging elements, local boundary conditions respectivelly. Assuming upwind scheme, using (4), (5), and (3) we can write solution of the equation (2) (relation between two consecutive time steps) on e_i in the form

$$c_i^{n+1} = c_i^n - \frac{\Delta t}{V_i}\left[\sum_{j\in\mathcal{N}_i}\left[U_{ij}^+ c_i + U_{ij}^- c_j\right] + \sum_{k\in\mathcal{B}_i}\left[U_{ik}^{e+} c_i + U_{ik}^{e-} c_{B_{ik}}\right]\right]. \tag{7}$$

Where $c_{B_{ik}}$ are values of Dirichlet boundary conditions which belong to e_i. Formula (12) can be rewritten into the matrix notation

$$\mathbf{c}^{n+1} = (\mathbf{I} + \Delta t\mathbf{A}) \cdot \mathbf{c}^n + \Delta t\mathbf{B} \cdot {\mathbf{c_B}}^n \tag{8}$$

Where $\mathbf{c}$ is vector of c_i^{n+1}, $\mathbf{A}$ is a square matrix composed from $\frac{U_{ij}^+}{V_i}$, $\frac{U_{ij}^-}{V_i}$, and $\frac{U_{ij}^{e+}}{V_i}$. $\mathbf{B}$ is in general rectangular matrix composed from $\frac{U_{ij}^{e-}}{V_i}$ and ${\mathbf{c_B}}^n$ is vector of Dirichlet boundary conditions.matrix definition. There is one stability condition for time step which is called Courant-Friedrich-Levy condition. For the problem without sources/sinks it can be written as

$$\Delta t_{max} = \min_i\left(\frac{V_i}{\sum_j U_{ij}^+ + \sum_k U_{ik}^{e+}}\right) = \min_i\left(\frac{V_i}{\sum_j |U_{ij}^-| + \sum_k |U_{ik}^{e-}|}\right). \tag{9}$$

This condition has a physical interpretation, which can be understood as conservation law - volume that intakes/drains to/from element e_i can not be higher then element volume V_i. From algebraical point of view this condition can be seen as a condition which bounds norm of the evolution operator as follows

$$\|\mathbf{I} + \Delta t\mathbf{A} \quad \Delta t\mathbf{B}\| \leq 1. \tag{10}$$

Generalization

This approach can be used as well as for more general element connections – for compatible/non-compatible element interconnection, if we know the flow integral values (U_{ij}^+ or U_{ij}^-). The most

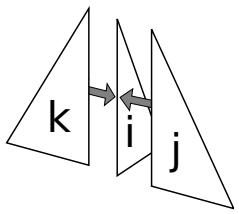

Figure 2: Edge with 3 elements

general case of connection is relation among n elements like in figure (2). For this case we define edge element indexset $\mathcal{G}_l$ that contains all the indexes of elements which sides make l-th edge (g_l), so that $\mathcal{G}_l = \{i, j, k\}$. For $\mathcal{G}_l$ we introduce its subsets $\mathcal{G}_{ij}$, $\mathcal{G}_{ji}$, $\mathcal{G}_{ik}$, $\mathcal{G}_{ki}$, $\mathcal{G}_{kj}$, and $\mathcal{G}_{jk}$, where $\mathcal{G}_{ij} = \mathcal{G}_{ik} = \mathcal{G}_l \backslash i = \{j, k\}$, $\mathcal{G}_{ji} = \mathcal{G}_{jk} = \mathcal{G}_l \backslash j = \{i, k\}$, and $\mathcal{G}_{ki} = \mathcal{G}_{kj} = \mathcal{G}_l \backslash k = \{i, j\}$.

It can be written in the same way for any edge g with more than 3 elements, it is hold $|\mathcal{G}_g| - 1 = |\mathcal{G}_{ab}|; \forall a, b \in \mathcal{G}_g$. For l-th edge (g_l) we can define total edge flow U_{g_l} eg. as

$$\begin{aligned} U_{g_l} &= \sum_{m \in \mathcal{G}_{ji}} \left[U^+_{jm} + |U^-_{mj}|\right] = \sum_{m \in \mathcal{G}_{jk}} \left[U^+_{jm} + |U^-_{mj}|\right] \\ &= \sum_{m \in \mathcal{G}_{ij}} \left[U^+_{im} + |U^-_{mi}|\right] = \sum_{m \in \mathcal{G}_{ik}} \left[U^+_{im} + |U^-_{mi}|\right] \\ &= \sum_{m \in \mathcal{G}_{ki}} \left[U^+_{km} + |U^-_{mk}|\right] = \sum_{m \in \mathcal{G}_{kj}} \left[U^+_{km} + |U^-_{mk}|\right], \end{aligned} \tag{11}$$

U_{g_l} with respect to any e_m; $m \in \mathcal{G}_l$ has to have the same value because continuity equation, for assumed incompresible flow, has to be fulfilled in every edge. Edges with more than two elements and two and more nonzero intakes to edge realize an ideal mixing (to an average concentration) with weights which will be specified later. This fact modifies equation (12) on the general mesh into the form

$$c_i^{n+1} = c_i^n - \frac{\Delta t}{V_i} \left[\sum_{j \in \mathcal{N}_i} \left[U^+_{ij} c_i + \frac{U^-_{ij}}{\sum\limits_{k \in \mathcal{G}_{ij}} \left[U^+_{ki} + |U^-_{ik}|\right]} \sum_{k \in \mathcal{G}_{ij}} U^+_{ki} c_k \right] + \sum_{k \in \mathcal{B}_i} \left[U^{e+}_{ik} c_i + U^{e-}_{ik} c_{B_{ik}}\right] \right]. \tag{12}$$

Concentrations c_k, $k \in \mathcal{G}_{ij}$ that may intakes into element e_i are weighted with weights

$$\alpha_k = \frac{U^+_{ki}}{\sum\limits_{k \in \mathcal{G}_{ij}} \left[U^+_{ki} + |U^-_{ik}|\right]}, \tag{13}$$

so that the ideal mixing in this edge leads to the average concentration

$$c_{av} = \frac{\sum\limits_{k \in \mathcal{G}_{ij}} U^+_{ki} c_k}{\sum\limits_{k \in \mathcal{G}_{ij}} \left[U^+_{ki} + |U^-_{ik}|\right]}. \tag{14}$$

Matrix notation is the same as in (8). Finally ...

Flow123D ini file format

Section: [**Global**]

KEY	TYPE	DEFAULT	DESCRIPTION
`Problem_type`	**int**	NULL	Type of solved problem. Currently supported: 1 = steady saturated flow
`Description`	**string**	*undefined*	Short description of solved problem - any text.
`Stop_time`	**double**	1.0	Time interval of the whole problem.[time units]
`Save_step`	**double**	1.0	The output with transport is written every `Save_step`. [time units]

Section: [**Input**]

KEY	TYPE	DEFAULT	DESCRIPTION
`Mesh`	**string**	NULL	Name of file containig definition of the mesh for the problem.
`Material`	**string**	NULL	Name of file with hydraulical properties of the elements.
`Boundary`	**string**	NULL	Name of file with boundary condition data.
`Neighbouring`	**string**	NULL	Name of file describing topology of the mesh.
`Sources`	**string**	NULL	Name of file with definition of fluid sources. This is optional file, if this key is not defined, calculation goes on without sources.

Section: [**Transport**]

KEY	TYPE	DEFAULT	DESCRIPTION
`Transport_on`	**YES/NO**	NO	If set "YES" program compute transport too.
`Sorption`	**YES/NO**	NO	If set "YES" program include sorption too.
`Dual_porosity`	**YES/NO**	NO	If set "YES" program include dual porosity too.
`Reactions`	**YES/NO**	NO	If set "YES" program include reactions too.
`Concentration`	**string**	NULL	Name of file with initial concentration.
`Transport_BCD`	**string**	NULL	Name of file with boundary condition for transport.
`Transport_out`	**string**	NULL	Name of transport output file.
`Transport_out_im`	**string**	NULL	Name of transport immobile output file.
`Transport_out_sorp`	**string**	NULL	Name of transport sorbed output file.
`Transport_out_im_sorp`	**string**	NULL	Name of transport sorbed immobile output file.
`N_substances`	**int**	-1	Number of substances.
`Substances`	**string**	*undefined*	Names of the substances separated by commas.
`Substances_density_scales`	**list of doubles**	1.0	Scales of substances for the density flow calculation.

Section: [**Constants**]

KEY	TYPE	DEFAULT	DESCRIPTION
`g`	**double**	1.0	Gravity acceleration.
`rho`	**double**	1.0	Density of fluid.

Section: [**Run**]

KEY	TYPE	DEFAULT	DESCRIPTION
`Screen_verbosity`	**int**	8	Amount of messages printed on the screen. (0 = no messages, ..., 7 = all messages)
`Pause_after_run`	**YES/NO**	NO	If set to "YES", the program waits for a key press before it finishes.

Section: **[Solver]**

KEY	TYPE	DEFAULT	DESCRIPTION
`Use_last_solution`	**YES/NO**	NO	If set to "YES", uses last known solution for chosen solver.
`Solver_name`	**string**	petsc	Type of linear solver. Supported solvers are: `petsc`, `petsc_matis` (experimental)

`Solver_params`	**string**	NULL	PETSc options to override default choice of iterative solver and preconditioner (use with care). In particular to use UMFPACK sequantial direct solver set:
			`Solve_params = "-ksp preonly -pc_type lu -pc_factor_mat_solver_package umfpack"`
			To use parallel direct solver MUMPS use:
			`Solve_params = "-ksp preonly -pc_type lu -pc_factor_mat_solver_package mumps -mat_mumps_icntl_14 5"`

`Keep_solver_files`	**YES/NO**	NO	If set to "YES", files for solver are not deleted after the run of the solver.
`Manual_solver_run`	**YES/NO**	NO	If set to "YES", programm stops after writing input files for solver and lets user to run it.
`Use_control_file`	**YES/NO**	NO	If set to "YES", programm do not create control file for solver, it uses given file.
`Control_file`	**string**	NULL	Name of control file for situation, when `Use_control_file` ȲES.
`NSchurs`	**int**	2	Number of Schur complements to use. Valid values are 0,1,2. The last one should be the fastest.
`Solver_accuracy`	**double**	1e-6	When to stop solver run - value of residum of matrix. Useful values from 1e-4 to 1e-10. Bigger number = faster run, less accuracy.
`max_it`	**int**	200	Maximum number of iteration of linear solver.

Section: [**Output**]

KEY	TYPE	DEFAULT	DESCRIPTION
`Write_output_file`	**YES/NO**	NO	If set to "YES", writes output file.
`Output_file`	**string**	NULL	Name of the output file (type 1).
`Output_file_2`	**string**	NULL	Name of the output file (type 2).
`Output_digits`	**int**	6	Number of digits used for floating point numbers in output file.
`Output_file_type`	**int**	1	Type of output file 1 - GMSH like format 2 - Flow data file 3 - both files (two separate names)
`POS_view_params`	**double[8]**	0 0 0 1 1 1 0 0	[x y z] angle of rotation "RotationX" [x y z] scaling "ScaleX" [x y] screen position shift "TranslationX"
`Pos_format`	**string**	ASCII	Output file format. One can use: ASCII, BIN, or VTK_SERIAL_ASCII

Description: Options controling output file of the programm

Section: [**Semchem_module**]

KEY	TYPE	DEFAULT	DESCRIPTION
`Compute_reactions`	**Yes/No**	"No"	NO = transport without chemical reactions YES = transport influenced by chemical reactions
`Output_precission`	**int**	1	Number of decimal places written to output file created by Semchem_module.
`Number_of_further_species`	**int**	0	Concentrations of these species are not computed, because they are ment to be unexghaustible.
`Temperature`	**double**	0.0	Temperature, one of state variables of the system.
`Temperature_Gf`	**double**	0.0	Temperature at which Free Gibbs Energy is specified.
`Param_Afi`	**double**	0.0	Parameter of the Debuy-Hückel equation for activity coeficients computation.
`Param_b`	**double**	0.0	Parameter of the Debuy-Hückel equation for activity coeficients computation.
`Epsilon`	**double**	0.0	Epsilon specifies relative norm of residuum estimate to stop numerical algorithms used by Semchem_module.
`Time_steps`	**int**	1	Number of transport step subdivisions for Semchem_module.
`Slow_kinetics_substeps`	**int**	0	Number of substeps performed by Runge-Kutta method used for slow kinetics simulation.
`Error_norm_type`	**string**	"Absolute"	Through wich kind of norm the error is measured.
`Scalling`	**boolean**	"No"	Type of the problem preconditioning for better convergence of numerical method.

Section: [**Aqueous_species**]

KEY	TYPE	DEFAULT	DESCRIPTION
`El_charge`	**int**	0	Electric charge of an Aqueous_specie particleunder consideration.
`dGf`	**double**	0.0	Free Gibbs Energy valid for TemperatureGf.
`dHf`	**double**	0.0	Enthalpy
`Molar_mass`	**double**	0.0	Molar mass of Aqueous_species.

Section: [**Further_species**]

KEY	TYPE	DEFAULT	DESCRIPTION
`Specie_name`	**string**	""	Name belonging to Further_specie under consideration.
`dGf`	**double**	0.0	Free Gibbs Energy valid for TemperatureGf.
`dHf`	**double**	0.0	Enthalpy
`Molar_mass`	**double**	0.0	Molar mass of Further_species.
`Activity`	**double**	0.0	Activity of Further_species.

Section: [**Reaction_i**]

KEY	TYPE	DEFAULT	DESCRIPTION
`Reaction_type`	**string**	"unknown"	Type of considered reaction (Equilibrium, Kinetics, Slow_kinetics).
`Stoichiometry`	**int**	0	Stoichiometric coeficients of species taking part in i-th reaction.
`Kinetic_constant`	**double**	0.0	Kinetic constant for determination of reaction rate.
`Order_of_reaction`	**int**	0	Order of kinetic reaction for participating species.
`Equilibrium_constant`	**double**	0.0	Equilibrium constant defining i-th reaction.

Section: [**Decay_module**]

KEY	TYPE	DEFAULT	DESCRIPTION
`Compute_decay`	**Yes/No**	"No"	Enables to switch on simulation of radioactive decay.
`Nr_of_decay_chain`	**int**	0	How many steps is defined decay chain consisting of?

Section: [**Decay_i**]

KEY	TYPE	DEFAULT	DESCRIPTION
`Reaction_type`	**enumerate**	"decay"	In this time just decay type is possible.
`Nr_of_isotopes`	**int**	0	How many isotopes does the section work with.
`Substance_ids`	**array of int**	NULL	Sequence of ids describing the order of isotopes in decay chain.
`Half_lives`	**array of double**	NULL	Contain half-lives belonging to isotopes defined by ids.
`Bifurcation_on`	**Yes/No**	"No"	Make it possible to switch of bifurcation in current [Decay_i].
`Bifurcation`	**array of double**	NULL	Gives a percentage, which is the first isotope in current [Decay_i] decaying to products.

Mesh file format version 2.0

The mesh file format comes from the GMSH system. Following text is copied from the GMSH documentation.

`============== BEGIN OF INSERTED TEXT ==============`

Version 2.0 of the `.MSH` file format is Gmsh's new native mesh file format. It is very similar to the old one (Version 1.0), but is more general: it contains information about itself and allows to associate an arbitrary number of integer tags with each element.
The `.MSH` file format, version 2.0, is divided in three sections, defining the file format (`$MeshFormat-$EndMeshFormat`), the nodes (`$Nodes-$EndNodes`) and the elements (`$Elements-$EndElements`) in the mesh:

`$MeshFormat`
`2.0` *file-type data-size*
`$EndMeshFormat`
`$Nodes`
number-of-nodes
node-number x-coord y-coord z-coord
`...`
`$EndNodes`
`$Elements`
number-of-elements
elm-number elm-type number-of-tags <tags> node-number-list
`...`
`$EndElements`

where:

file-type is an integer equal to 0 in the ASCII file format.

data-size is an integer equal to the size of the floating point numbers used in the file (usually, *data-size* = sizeof(double)).

number-of-nodes is the number of nodes in the mesh.

node-number is the number (index) of the *n*-th node in the mesh. Note that the *node-number*s do not have to be given in a consecutive (or even an ordered) way.

x-coord y-coord z-coord are the floating point values giving the X, Y and Z coordinates of the *n*-th node.

number-of-elements is the number of elements in the mesh.

elm-number is the number (index) of the *n*-th element in the mesh. Note that the *elm-number*s do not have to be given in a consecutive (or even an ordered) way.

elm-type defines the geometrical type of the *n*-th element:

1 Line (2 nodes)
2 Triangle (3 nodes)
3 Quadrangle (4 nodes)
4 Tetrahedron (4 nodes)
5 Hexahedron (8 nodes)
6 Prism (6 nodes)
7 Pyramid (5 nodes)
8 Second order line (3 nodes)
9 Second order triangle (6 nodes)
11 Second order tetrahedron (10 nodes)
15 Point (1 node)

number-of-tags gives the number of tags for the n-th element. By default, Gmsh generates meshes with two tags and reads files with an arbitrary number of tags: see below.

tag is an integer tag associated with the n-th element. By default, the first tag is the number of the physical entity to which the element belongs; the second is the number of the elementary geometrical entity to which the element belongs; the third is the number of a mesh partition to which the element belongs.

node-number-list is the list of the node numbers of the n-th element (separated by white space, without commas). The ordering of the nodes is given in Gmsh node ordering; for second order elements, the first order nodes are given first, followed by the nodes associated with the edges, followed by the nodes associated with the faces (if any). The ordering of these additional nodes follows the ordering of the edges/faces given in Gmsh node ordering.

`============== END OF INSERTED TEXT ==============`

More information about GMSH can be found at its homepage:
`http://www.geuz.org/gmsh/`

Comments concerning `1-2-3-FLOW`:

- Every inconsistency of the file stops the calculation. These are:
  - Existence of nodes with the same *node-number*.
  - Existence of elements with the same *elm-number*.
  - Reference to non-existing node.
  - Reference to non-existing material (see below).
  - Difference between *number-of-nodes* and actual number of lines in nodes' section.
  - Difference between *number-of-elements* and actual number of lines in elements' section.
- By default `1-2-3-FLOW` uses meshes with *number-of-tags* = 2.

  tag1 is number of region in which the element lies.

  tag2 is number of material (reference to `.MTR` file) in the element.

- Currently, line ($type = 1$), triangle ($type = 2$) and tetrahedron ($type = 4$) are the only supported types of elements. Existence of an element of different type stops the calculation.
- Wherever possible, we use the file extension `.MSH`. It is not required, but highly recomended.

Material properties file format, version 1.0

The file is divided in two sections, header and data. The extension `.MTR` is highly recomended for files of this type.

`$MaterialFormat`
`1.0` *file-type data-size*
`$EndMaterialFormat`
`$Materials`
number-of-materials
material-number material-type <material-type-specific-data> [text]
...
`$EndMaterials`
`$Storativity`
material-number <storativity-coefficient> [text]
...
`$EndStorativity`
`$Geometry`
material-number geometry-type <geometry-type-specific-coefficient> [text]
...
`$EndGeometry`
`$Sorption`
material-number substance-id sorption-type <sorption-type-specific-data> [text]
...
`$EndSorption`
`$SorptionFraction`
material-number <sorption-fraction-coefficient> [text]
...
`$EndSorptionFraction`
`$DualPorosity`
material-number <mobile-porosity-coefficient> <immobile-porosity-coefficient>
<nonequillibrium-coefficient-substance(0)> ...<nonequilibrium-coefficient-substance(n-1)>
[text]
...
`$EndDualPorosity`
`$Reactions`
reaction-type <reaction-type-specific-coefficient> [text]
...
`$EndReactions`

where:

file-type `int` — is equal 0 for the ASCII file format.

data-size `int` — the size of the floating point numbers used in the file. Usually *data-size* = sizeof(double).

number-of-materials `int` — Number of materials defined in the file.

material-number `int` — is the number (index) of the n-th material. These numbers do not have to be given in a consecutive (or even an ordered) way. Each number has to be

given only onece, multiple definition are treated as inconsistency of the file and cause stopping the calculation (exception $Sorption section).

material-type `int` — is type of the material, see table.

<*material-type-specific-data* > — format of this list depends on the *material - type*.

<*storativity-coefficient*> `double` — coefficient of storativity

geometry-type `int` — type of complement dimension parameter (only for 1D and 2D material), for 1D element is supported type 1 - cross-section area, for 2D element is supported type 2 - thickness.

<*geometry-type-specific-coefficient*> `double` — cross-section for 1D element or thickness for 2D element.

substance-id `int` — refers to number of transported substance, numbering starts on *0*.

sorption-type `int` — type 1 - linear sorption isotherm, type 2 - Freundlich sorption isotherm, type 3 - Langmuir sorption isotherm.

<*sorption-type-specific-data* > — format of this list depends on the *sorption - type*, see table.

Note: Section $Sorption is needed for calculation only if *Sorption* is turned on in the *ini* file.

<*sorption-fraction-coefficient*> `double` — ratio of the "mobile" solid surface in the contact with "mobile" water to the total solid surface (this parameter (section) is needed for calculation only if *Dual_porosity* and *Sorption* is together turned on in the ini file).

<*mobile-porosity-coefficient*> `double` — ratio of the mobile pore volume to the total volume (this parameter is needed only if *Transport_on* is turned on in the ini file).

<*immobile-porosity-coefficient*> `double` — ratio of the immobile pore volu-me to the total pore volume (this parameter is needed only if *Dual_porosity* is turned on in the ini file).

<*nonequilibrium-coefficient-substance(i)*> `double` — nonequilibrium coefficient for substance i, $\forall i \in \langle 0, n-1 \rangle$ where n is number of transported substances (this parameter is needed only if *Dual_porosity* is turned on in the ini file).

reaction-type `int` — type 0 - zero order reaction

<*reaction-type-specific-data* > — format of this list depends on the *reaction - type*, see table.

material-type	*material-type-specific-data*	Description
11	k	$\mathbf{K} = (k)$
-11	a	$\mathbf{A} = \mathbf{K}^{-1} = (a)$
21	k	$\mathbf{K} = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix}$
22	$k_x \quad k_y$	$\mathbf{K} = \begin{pmatrix} k_x & 0 \\ 0 & k_y \end{pmatrix}$
23	$k_x \quad k_y \quad k_{xy}$	$\mathbf{K} = \begin{pmatrix} k_x & k_{xy} \\ k_{xy} & k_y \end{pmatrix}$
-21	a	$\mathbf{A} = \mathbf{K}^{-1} = \begin{pmatrix} a & 0 \\ 0 & a \end{pmatrix}$
-22	$a_x \quad a_y$	$\mathbf{A} = \mathbf{K}^{-1} = \begin{pmatrix} a_x & 0 \\ 0 & a_y \end{pmatrix}$
-23	$a_x \quad a_y \quad a_{xy}$	$\mathbf{A} = \mathbf{K}^{-1} = \begin{pmatrix} a_x & a_{xy} \\ a_{xy} & a_y \end{pmatrix}$
31	k	$\mathbf{K} = \begin{pmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & k \end{pmatrix}$
33	$k_x \quad k_y \quad k_z$	$\mathbf{K} = \begin{pmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{pmatrix}$
36	$k_x \quad k_y \quad k_z \quad k_{xy} \quad k_{xz} \quad k_{yz}$	$\mathbf{K} = \begin{pmatrix} k_x & k_{xy} & k_{xz} \\ k_{xy} & k_y & k_{yz} \\ k_{xz} & k_{yz} & k_z \end{pmatrix}$
-31	a	$\mathbf{A} = \mathbf{K}^{-1} = \begin{pmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{pmatrix}$
-33	$a_x \quad a_y \quad a_z$	$\mathbf{A} = \mathbf{K}^{-1} = \begin{pmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \\ 0 & 0 & a_z \end{pmatrix}$
-36	$a_x \quad a_y \quad a_z \quad a_{xy} \quad a_{xz} \quad a_{yz}$	$\mathbf{A} = \mathbf{K}^{-1} = \begin{pmatrix} a_x & a_{xy} & a_{xz} \\ a_{xy} & a_y & a_{yz} \\ a_{xz} & a_{yz} & a_z \end{pmatrix}$

Note: all variables (k, k_x, k_y, k_z, k_{xy}, k_{xz}, k_{yz}, a, a_x, a_y, a_z, a_{xy}, a_{xz}, a_{yz}) are of the `double` type.

sorption-type	*sorption-type-specific-data*	Description
1	$k_D[1]$	$s = k_D c$
2	$k_F[(L^{-3} \cdot M^1)^{(1-\alpha)}] \quad \alpha[1]$	$s = k_F c^\alpha$
3	$K_L[L^3 \cdot M^{-1}] \quad s^{max}[L^{-3} \cdot M^1]$	$s = \frac{K_L s^{max}\ c}{1+K_L c}$

Note: all variables (k_D, k_F, α, K_L, s^{max}) are of the `double` type.

reaction-type	*reaction-type-specific-data*	Description
0	*substance-id*[1] $\quad k[M \cdot L^{-3} \cdot T^{-1}]$	$\frac{\partial c_m^{[substance\text{-}id]}}{\partial t} = k$

Where $c_m^{[substance\text{-}id]}$ is mobile concentration of substance with id *substance-id* and Δt is the internal transport time step defined by CFL condition.

text `char[]` — is a text description of the material, up to 256 chars. This parameter is

optional.

Comments concerning `1-2-3-FLOW`:

- If *number-of-materials* differs from actual number of material lines in the file, it stops the calculation.

Boundary conditions file format, version 1.0

The file is divided in two sections, header and data.

`$BoundaryFormat`
`1.0` *file-type data-size*
`$EndBoundaryFormat`
`$BoundaryConditions`
number-of-conditions
condition-number type <*type-specific-data*> *where* <*where-data*> *number-of-tags* <*tags*> *[text]*
...
`$EndBoundaryConditions`

where

file-type `int` — is equal 0 for the ASCII file format.

data-size `int` — the size of the floating point numbers used in the file. Usually *data-size* = sizeof(double).

number-of-conditions `int` — Number of boundary conditions defined in the file.

condition-number `int` — is the number (index) of the n-th boundary condition. These numbers do not have to be given in a consecutive (or even an ordered) way. Each number has to be given only onece, multiple definition are treated as inconsistency of the file and cause stopping the calculation.

type `int` — is type of the boundary condition. See below for definitions of the types.

<*type-specific-data*> — format of this list depends on the *type*. See below for specification of the *type-specific-data* for particular types of the boundary conditions.

where `int` — defines the way, how the place for the contidion is prescribed. See below for details.

<*where-data*> — format of this list depends on *where* and actually defines the place for the condition. See below for details.

number-of-tags `int` — number of integer tags of the boundary condition. It can be zero.

< *tags* > *number-of-tags*`*int` — list of tags of the boundary condition. Values are separated by spaces or tabs. By default we set *number-of-tags*=1, where *tag1* defines group of boundary conditions, "type of water" in our jargon. This can be used to calculate total fluxes through the boundary group.

[text] `char[]` — arbitrary text, description of the fracture, notes, etc., up to 256 chars. This is an optional parameter.

Types of boundary conditions and their data

type = `1` — Boundary condition of the Dirichlet's type

type = `2` — Boundary condition of the Neumann's type

type = `3` — Boundary condition of the Newton's type

type	*type-specific-data*	Description
1	*scalar*	Prescribed value of pressure or piez. head
2	*flux*	Prescribed value of flux through the boundary
3	*scalar sigma*	Scalar value and the σ coefficient

scalar, *flux* and *sigma* are of the `double` type.

Ways of defining the place for the boundary condition

where = `1` — Condition on a node

where = `2` — Condition on a (generalized) side

where = `3` — Condition on side for element with only one external side.

where	<*where-data*>	Description
1	*node-id*	Node id number, according to `.MSH` file
2	*elm-id sid-id*	Elm. id number, local number of side
3	*elm-id*	Elm. id number

The variables *node-id*, *elm-id*, *sid-id* are of the `int` type.

Comments concerning `1-2-3-FLOW`:

- We assume homegemous Neumman's condition as the default one. Therefore we do not need to prescribe conditions on the whole boundary.
- If the condition is given on the inner edge, it is treated as an error and stops calculation.
- Any inconsistence in the file stops calculation. (Bad number of conditions, multiple definition of condition, reference to non-existing node, etc.)
- At least one of the conditions has to be of the Dirichlet's or Newton's type. This is well-known fact from the theory of the PDE's.
- Local numbers of sides for *where* = 2 must be lower than the number of sides of the particular element and greater then or equal to zero.
- The element specified for *where* = 3 must have only one external side, otherwise the program stops.

Neighbouring file format, version 1.0

The file is divided in two sections, header and data. The extension `.NGH` is highly recomended for files of this type.

`$NeighbourFormat`
`1.0` *file-type data-size*
`$EndNeighbourFormat`
`$Neighbours`
number-of-neighbours
neighbour-number type <*type-specific-data*>
`...`
`$EndNeighbours`

where

file-type `int` — is equal 0 for the ASCII file format.

data-size `int` — the size of the floating point numbers used in the file. Usually *data-size* = sizeof(double).

number-of-neighbours `int` — Number of neighbouring defined in the file.

neighbour-number `int` — is the number (index) of the n-th neighbouring. These numbers do not have to be given in a consecutive (or even an ordered) way. Each number has to be given only onece, multiple definition are treated as inconsistency of the file and cause stopping the calculation.

type `int` — is type of the neighbouring.

<*type-specific-data*> — format of this list depends on the *type*.

Types of neighbouring and their specific data

type = `10` — "Edge with common nodes", i.e. sides of elements with common nodes. (Possible many elements)

type = `11` — "Edge with specified sides", i.e. sides of the edge are explicitly defined. (Possible many elements)

type = `20` — "Compatible", i.e. volume of an element with a side of another element. (Only two elements)

type = `30` — "Non-compatible" i.e. volume od an element with volume of another element. (Only two elements)

type	*type-specific-data*	Description
10	*n_elm eid1 eid2 ...*	number of elements and their ids
11	*n_sid eid1 sid1 eid2 sid2 ...*	number of sides, their elements and local ids
20	*eid1 eid2 sid2 coef*	Elm 1 has to have lower dimension
30	*eid1 eid2 coef*	Elm 1 has to have lower dimension

coef is of the `double` type, other variables are `int`s.

Comments concerning `1-2-3-FLOW`:

- Every inconsistency or error in the `.NGH` file causes stopping the calculation. These are especially:
  - Multiple usage of the same *neighbour-number*.
  - Difference between *number-of-neighbours* and actual number of data lines.
  - Reference to nonexisting element.
  - Nonsence number of side.
- The variables *sid?* must be nonegative and lower than the number of sides of the particular element.

Sources file format, version 1.0

The file is divided in two sections, header and data. The extension `.SRC` is highly recomended for files of this type.

`$SourceFormat`
`1.0` *file-type* *data-size*
`$EndSourceFormat`
`$Sources`
number-of-sources
eid *density*
`...`
`$EndSources`

where

file-type `int` — is equal 0 for the ASCII file format.

data-size `int` — the size of the floating point numbers used in the file. Usually *data-size* = sizeof(double).

number-of-sources `int` — Number of sources defined in the file.

eid `int` — is id-number of the element, where the source lies.

density `double` — is the density of the source, in volume of fluid per time unit. Possitive values are sources, negative are sinks.

Comments concerning `1-2-3-FLOW`:

- Every inconsistency or error in the `.SRC` file causes stopping the calculation. These are especially:
  - Multiple usage of the same *source-number*.
  - Difference between *number-of-sources* and actual number of data lines.
  - Reference to nonexisting element.

ASCII post-processing file format version 1.2

File format of this file comes from the GMSH system. Following text is copied from the GMSH documentation.

`=============== BEGIN OF INSERTED TEXT ===============`

The ASCII post-processing file is divided in several sections: one format section, enclosed between `$PostFormat-$EndPostFormat` tags, and one or more post-processing views, enclosed between `$View-$EndView` tags:

`$PostFormat`
`1.2` *file-type data-size*
`$EndPostFormat`
`$View`
view-name nb-time-steps
nb-scalar-points nb-vector-points nb-tensor-points
nb-scalar-lines nb-vector-lines nb-tensor-lines
nb-scalar-triangles nb-vector-triangles nb-tensor-triangles
nb-scalar-quadrangles nb-vector-quadrangles nb-tensor-quadrangles
nb-scalar-tetrahedra nb-vector-tetrahedra nb-tensor-tetrahedra
nb-scalar-hexahedra nb-vector-hexahedra nb-tensor-hexahedra
nb-scalar-prisms nb-vector-prisms nb-tensor-prisms
nb-scalar-pyramids nb-vector-pyramids nb-tensor-pyramids
nb-text2d nb-text2d-chars nb-text3d nb-text3d-chars
<*time-step-values*>
<*scalar-point-values*>
<*vector-point-values*>
<*tensor-point-values*>
<*scalar-line-values*>
<*vector-line-values*>
<*tensor-line-values*>
<*scalar-triangle-values*>
<*vector-triangle-values*>
<*tensor-triangle-values*>
<*scalar-quadrangle-values*>
<*vector-quadrangle-values*>
<*tensor-quadrangle-values*>
<*scalar-tetrahedron-values*>
<*vector-tetrahedron-values*>
<*tensor-tetrahedron-values*>
<*scalar-hexahedron-values*>
<*vector-hexahedron-values*>
<*tensor-hexahedron-values*>
<*scalar-prism-values*>
<*vector-prism-values*>
<*tensor-prism-values*>
<*scalar-pyramid-values*>

<vector-pyramid-values>
<tensor-pyramid-values>
<text2d> *<text2d-chars>*
<text3d> *<text3d-chars>*
`$EndView`

where:

file-type is an integer equal to 0 in the ASCII file format.

data-size is an integer equal to the size of the floating point numbers used in the file (usually, *data-size* = sizeof(double)).

view-name is a string containing the name of the view (max. 256 characters).

nb-time-steps is an integer giving the number of time steps in the view.

nb-scalar-points, *nb-vector-points*, ... are integers giving the number of scalar points, vector points,... in the view.

nb-text2d, *nb-text3d* are integers giving the number of 2D and 3D text strings in the view.

nb-text2d-chars, *nb-text3d-chars* are integers giving the total number of characters in the 2D and 3D strings.

time-step-values is a list of *nb-time-steps* double precision numbers giving the value of the time (or any other variable) for which an evolution was saved.

scalar-point-value, *vector-point-value*, ... are lists of double precision numbers giving the node coordinates and the values associated with the nodes of the *nb-scalar-points* scalar points, *nb-vector-points* vector points,..., for each of the *time-step-values*.

For example, *vector-triangle-value* is defined as:

coord1-node1 coord1-node2 coord1-node3
coord2-node1 coord2-node2 coord2-node3
coord3-node1 coord3-node2 coord3-node3
comp1-node1-time1 comp2-node1-time1 comp3-node1-time1
comp1-node2-time1 comp2-node2-time1 comp3-node2-time1
comp1-node3-time1 comp2-node3-time1 comp3-node3-time1
comp1-node1-time2 comp2-node1-time2 comp3-node1-time2
comp1-node2-time2 comp2-node2-time2 comp3-node2-time2
comp1-node3-time2 comp2-node3-time2 comp3-node3-time2
...

text2d is a list of 4 double precision numbers:

coord1 coord2 style index

where *coord1* and *coord2* give the coordinates of the leftmost element of the 2D string in screen coordinates, *index* gives the starting index of the string in *text2d-chars* and *style* is currently unused.

text2d-chars is a list of *nb-text2d-chars* characters. Substrings are separated with the '^' character (which is a forbidden character in regular strings).

text3d is a list of 5 double precision numbers

coord1 coord2 coord3 style index

where *coord1*, *coord2* and *coord3* give the coordinates of the leftmost element of the 3D string in model (real world) coordinates, *index* gives the starting index of the string in *text3d-chars* and *style* is currently unused.

text3d-chars is a list of *nb-text3d-chars* chars. Substrings are separated with the '^' character.

```
============== END OF INSERTED TEXT ==============
```

More information about GMSH can be found at its homepage:
`http://www.geuz.org/gmsh/`

Comments concerning `FFLOW20`:

- `FFLOW20` generates `.POS` file with four views: Elements' pressure, edges' pressure, interelement fluxes and complex view. First three views shows "raw data", results obtained by the solver without any interpolations, smoothing etc. The fourth view contains data processed in this way.

  Elements' pressure: Contains only *scalar-triangle-values*. Triangles are the same as the elements of the original mesh. We prescribe constant value of the pressure on the element, as it was calculated by the solver as the unknown p. Therefore, the three values on every triangle are the same.

  Edge pressure: Contains only *scalar-line-values*. The lines are the same as the edges of the elements of the original mesh. We prescribe constant value of the pressure on the edge, as it was calculated by the solver as the unknown λ. Therefore, the two values on every edge are the same.

  Interelement flux: Contains *vector-point-values* and *scalar-triangle-values*. The *scalar-triangle-values* carry no information, all values are set to 0, these are in the file only to define a shape of the elements. The points for the *vector-point-values* are midpoints of the sides of the elements. The vectors are calculated as $u\mathbf{n}$, where u is value of the flux calculated by the solver and $\mathbf{n}$ is normalized vector of outer normal of the element's side.

  Complex view: Contains *scalar-triangle-values* and *vector-point-values*. The *scalar-triangle-values* shows the shape of the pressure field. The triangles are the the same as the elements of the original mesh. Values of pressure in nodes are interpolated from ps and λs. The *vector-point-values* shows the velocity of the flow in the centres of the elements.