sedFlow
User manual

Published by the Swiss Federal Research Institute WSL
Written by Florian U. M. Heimann

June 2014
Version 1.00

The manual is subdivided in two parts: The *Quick start guide* from page 4 to page 11 gives the minimum information to run the model. The *Further specifications* from page 12 to page 55 complement the short introduction of the *Quick start guide* and gives an in depth description of the different ways to adjust simulation parameters and output formats.

The manual provides an introduction to the use of the model *sedFlow*. For an in depth description of the model please refer to Heimann et al. [3]. Examples of the application of *sedFlow* are given in Heimann et al. [4].

Contents

Part I
Quick start guide

1 General remarks

The new model *sedFlow* has been developed to provide an efficient tool for the simulation of bedload transport in mountain streams. The following elements were important for the development of *sedFlow*:

(i) provision of a sediment transport model together with its complete source code open and free of charge

(ii) consideration of state of the art approaches for the calculation of bedload transport in steep channels accounting for macro-roughness effects

(iii) individual calculation for several grain diameter fractions (fractional transport)

(iv) consideration of the effects of adverse slopes in terms of pondages e.g. due to sudden sediment deposition by debris flow inputs

(v) fast calculations for modelling entire catchments, and for automated calculation of many scenarios exploring a range of parameter space

(vi) flexibility in model development featuring an object oriented code design

(vii) flexibility in model application featuring an easy and straightforward pre- and postprocessing of simulation data.

Thus, the model *sedFlow* fills a gap in the range of existing sediment transport models for mountain streams and the mentioned aims have led to the implementation described in the following sections. This implementation represents a current state and may be easily extended and adjusted in the future.

sedFlow is a console program without graphical user interface. This format was chosen to enable automated simulation starts within batch or shell scripts and especially to allow for short calculation times. The input files are prepared in other programs. *sedFlow* reads these files and creates output files which are continuously updated in the course of a simulation. To check the progress of a simulation one can take a look at the *ElapsedSeconds* column of any output file. As some programs block files for writing, it is recommended to open copies of an output file as long as the code is still running.

As *sedFlow* is intended for fast simulations (which implies long time steps), executable versions of the code are provided, which will give a warning

if the length of the time step falls below a threshold of 1 second or 100 seconds respectively. A version which does not give a warning for short time steps is provided as well.

Most of the simulation data is fed to the model by tabulator delimited text files, which can be created and edited with regular spreadsheet applications such as *Microsoft Excel*. Each such file contains one header line, which defines the number of columns for this file together with the column names. The code selects the data based on the column names. Thus, there is no fixed order for the columns and additional extra columns will be ignored. Empty cells are ignored by the code as well. Within rows containing more elements than the header line, the first elements up to the number of header line entries are used as data entries. Any further entries are ignored. Rows containing less entries than the header line are ignored as well. By this it is easy to include comments in the spreadsheet files.

Most of the simulation parameters are fed to the model by an extensible markup language (xml) input file. For the creation and editing of the input xml file on the *Microsoft Windows* platform the use of the freeware program *Microsoft XML Editor 2007* is recommended. The code will find any information within the xml file based on the position in the node hierarchy and the node name. Additional extra nodes are simply ignored and may be used for commenting.

Any typing errors in the names of the files, columns or nodes will lead to error messages or even worse that the model will ignore the corresponding element without giving a warning at all. Blanks are usually ignored by the code.

In the tables and figures of this manual three question marks (???) indicate a required user input for which no default exists. If such a required input is not given, usually an error message will request it.

2 Starting the model

The starting point for each simulation is the main input xml file. The location of this file can be given as first argument of the model call e.g. when starting simulations within a script. If the location is not given or the code is started by a simple doubleclick, the model will ask for the location of the file. The location can be given either by an absolute path or by a relative path starting at the location of the model. It is recommended to copy the model into the same folder as the main input xml file and then simply input the name of the xml file.

At the model start the user is requested to hit enter to accept the terms of the license, under which *sedFlow* is distributed. Alternatively, the user may type *acceptLicense* as second argument of the model call, e.g. when starting simulations within a script.

3 The simulation folder

The location of the main input xml file usually defines the simulation folder which always has the same structure (Fig. 1).

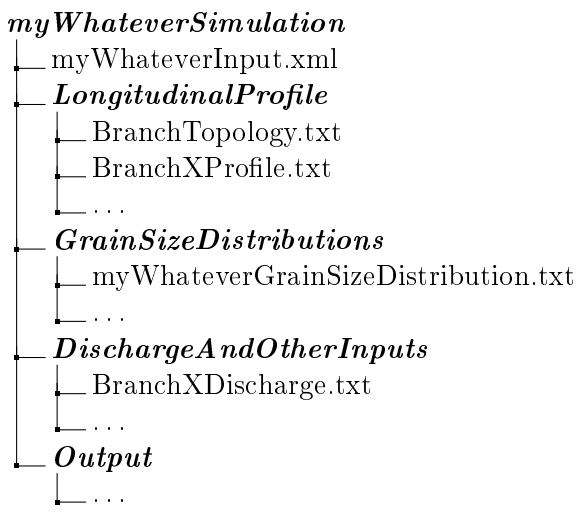

Figure 1: Minimum structure for simulation folder.

4 *BranchTopology.txt*

First, the code will go into the folder *LongitudinalProfile* and look for the file *BranchTopology.txt*. Within *sedFlow* the river network is organised in branches. A branch is a section of a river without any tributaries or confluences. Different branches may be connected by confluences. The topology of the river network is defined by the *BranchTopology.txt*. The file contains the two columns *BranchIDs* and *DownstreamBranchIDs* (Tab. 1), in which the IDs are given as integer numbers. It is required that a branch can only flow into another branch with a higher ID. Examples are given in Figure 2 and Tables 2 and 3.

Table 1: Structure of BranchTopology.txt

BranchIDs	DownstreamBranchIDs
???	???
⋮	⋮

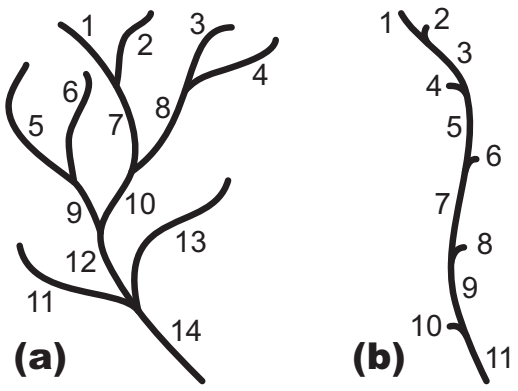

Figure 2: Examples of potential BranchTopology of (a) a river network and (b) a main channel with stub tributaries.

Table 2: Structure of BranchTopology.txt for river network example of Fig. 2a.

BranchIDs	Downstream-BranchIDs
1	7
2	7
3	8
4	8
5	9
6	9
7	10
8	10
9	12
10	12
11	14
12	14
13	14

Table 3: Structure of BranchTopology.txt for main channel with stub tributaries example of Fig. 2b.

BranchIDs	Downstream-BranchIDs
1	3
2	3
3	5
4	5
5	7
6	7
7	9
8	9
9	11
10	11

5 *BranchXProfile.txt*

For each branch defined in the *BranchTopology.txt*, the code will look for a *BranchXProfile.txt* file within the *LongitudinalProfile* folder with *X* substituted by the current branch ID. Within *sedFlow* every branch is discretised into river reaches. Every reach is described by the cross section at its upstream end. Each such cross section is defined by one row in *BranchXProfile.txt*. The file contains at least the following columns (Tab. 4):

Table 4: Minimum structure of BranchXProfile.txt

KilometrageUpstreamDirected	ElevationInM	ChannelWidthInM	StrataGrainSizeDistribution
???	???	???	???
⋮	⋮	⋮	⋮

KilometrageUpstreamDirected gives an along channel kilometrage which may be e.g. the distance to the river mouth. Within the file, reaches should be sorted from upstream to downstream i.e. the kilometrage values should decrease. It is advisable to define cross sections at roughly equal distances, which should not be smaller than about 50 m. In this context it is important to note that the lowermost cross section of a feeding branch and the uppermost cross section of the recieving branch should *not* be the same. This would define a reach of length zero and lead to kryptic artifacts. Rather, the two profiles should have a distance as described before. In the same way, the lowermost reach of the complete system should also have an adequate length, usually defined by the distance of its cross section to a kilometrage of zero.

ElevationInM defines the river bed elevation of the cross section at the upstream end of the reach. Implicitly, this also defines the slope of the reach. When using a kinematic wave routing it is important to make sure that no adverse i.e. negative slopes occur, as they would lead to kryptic artefacts. The slope of the lowermost reach of the system is set equal to the slope of the next upstream reach in front of it.

ChannelWidthInM defines the width of the infinitely deep rectangular channel.

StrataGrainSizeDistribution defines the file names (without ".txt" extension) of the grain size distributions of the alluvium to be used at the start of a simulation. Of course, individual file names may appear repeatedly within this column.

6 Grain size distributions

For each grain size distribution defined in the *BranchXProfile.txt* files, the code will try to find the corresponding file (Tab. 5) in the *GrainSizeDistributions* folder. The column *GrainDiameterInCM* usually gives the upper boundaries for the individual grain size fractions. These values need to be sorted in increasing order and need to be identical in all grain size distribution files. The column *RelativeAbundance* gives the relative abundance of the individual fractions (*not* cumulative). The abundances are normalised by the code. Therefore, they do not need to sum up to 1 or 100%.

Table 5: Structure of grain size distribution spreadsheets

GrainDiameterInCM	RelativeAbundance
???	???
⋮	⋮

7 *BranchXDischarge.txt*

The branch topology defines those branches for which no upstream branch is available. For these headwater branches the code tries to find the *BranchX Discharge.txt* files (Tab. 6) within the *DischargeAndOtherInputs* folder with X substituted by the corresponding branch ID. The column *ElapsedSeconds* defines the points in time at which a discharge value is given. Of course, these values should increase. The column *DischargeInM3PerS* defines the corresponding discharge values in $\frac{m^3}{s}$. Between the points of this discrete time series, the discharge values are interpolated linearly. Beyond the range of this time series, discharge is constantly set equal to the first or last discharge value respectively. If one wants to feed water (and sediment) at a certain point to the system, it is recommended to introduce a stub branch consisting of just one reach and give a *BranchXDischarge.txt* for it.

Table 6: Structure of BranchXDischarge.txt

ElapsedSeconds	DischargeInM3PerS
???	???
⋮	⋮

8 The main input xml

Most of the simulation data is defined in the spreadsheet files. Most of the simulation parameters will be defined in the main input xml, the location of which already has been discussed in section 2. The name of the root node of this file defines the input reader i.e. the way, in which all information is read and interpreted. For this manual the root node name is *SEDFLOW_StandardInput*. The root node contains at least the two main nodes *overallParameters* and *riverSystemMethods* (Fig. 3), which will be described in the following. An example with potential values is given in Fig. 4. For the sake of clarity, the main nodes are printed in dark red in all figures while any other node, which contains child nodes, is printed in blue.

SEDFLOW_StandardInput
- **overallParameters**
  - lowerDiameterBoundaryForFinestFractionInCM ..... ???
- **riverSystemMethods**
  - **sedimentEnergySlopeCalculationMethod**
    - realisationType ..... ???
  - **bedloadTransportEquations**
    - realisationType ..... ???

Figure 3: Minimum xml.

The *overallParameters* collect values of typically single parameters, which are used throughout the complete simulation space and time. It contains at least the node *lowerDiameterBoundaryForFinestFractionInCM*, which defines the lower boundary for the finest grain size fraction. Make sure that its value is smaller than the smallest diameter of the grain size distribution files (section 6).

The *riverSystemMethods* typically collect the procedures and algorithms which are used in the course of a simulation. The names of the nodes specify the job, which is to be fulfilled by the corresponding method. The child nodes specify the way in which the method fulfills its job. Typically, there is always one child node called *realisation* to select the general algorithm of the current method. Depending on the realisation there may be other child nodes defining further parameters, which are specific for the current realisation. Different realisations and their various parameters are described in section 13. The *riverSystemMethods* contain at least the nodes *sedimentEnergySlopeCalculationMethod* and *bedloadTransportEquations*. The node *sedimentEnergySlopeCalculationMethod* is mainly used to determine whether or not the energy slope for the determination of bedload transport capacity should be reduced in order to account for flow resistance partitioning. The node *bedloadTransportEquations* defines the way, in which the bedload trans-

port capacity is estimated. The recommended combination of realisations is *WithFlowResistancePartitioning* and *RickenmannBedloadCapacityBasedOnTheta* (Fig. 4).

SEDFLOW_StandardInput
- **overallParameters**
  - lowerDiameterBoundaryForFinestFractionInCM . . . . . `0.2`
- **riverSystemMethods**
  - **sedimentEnergySlopeCalculationMethod**
    - realisationType . . . . . `WithFlowResistancePartitioning`
  - **bedloadTransportEquations**
    - realisationType . . . `RickenmannBedloadCapacityBasedOnTheta`

Figure 4: Minimum xml with recommended values.

Please consider the cautionary notes given in section 13.2 related to the water flow routing method *UniformDischarge*, which is used by default in the setup of Figure 3.

Part II
Further specifications

The *Quick start guide* on the previous pages described the minimum amount of information which is needed to run a simulation. In such a case, many simulation parameters are set to their default values. The user can override these default values simply by adding further files to the simulation folder or further columns to the spreadsheet files or further nodes to the input xml file. (It is important to note that complete columns need to be given. Even if only one value differs from the default, the complete column needs to be input with the defaults given explicitly and the single other value. If default values are not given and the corresponding rows have therefore less entries than the header line, these rows will be ignored completely.)

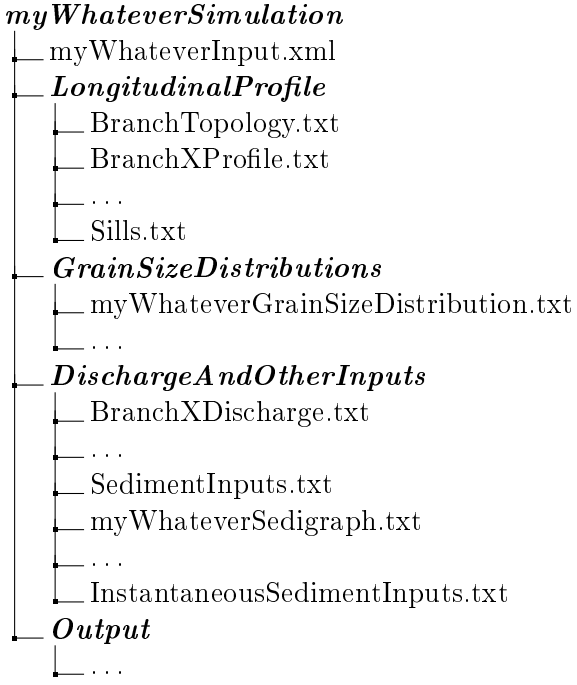

Figure 5: Complete structure for simulation folder.

The complete structure for the simulation folder including all possible input files is displayed in Figure 5. The complete structure of the *BranchXProfile.txt* spreadsheet files including all possible columns is described in section 9 and summarised in the Tables 7 & 8. Predefined sediment inputs may be included in a simulation by additional files in the *DischargeAndOtherInputs* folder as described in section 10. The influence of sills may be considered in a simulation by including an additional file in the *Longitudi-*

nalProfile folder as described in section 11. The complete structure for the input xml file is described in the section 12 and an example of the complete xml structure is summarised in the Figures 6 to 10.

9 Complete structure for BranchXProfile.txt

The complete structure of *BranchXProfile.txt* is summarised in the Tables 7 & 8. The columns *KilometrageUpstreamDirected*, *ElevationInM*, *ChannelWidthInM* and *StrataGrainSizeDistribution* have been described in section 5.

The *AlluviumThicknessInM* defines the amount of erodible material in a reach. If this column is not given the model will set its values to 4000.0 by default.

As described in section 5, the *StrataGrainSizeDistribution* defines the grain size distributions of the alluvium to be used at the start of a simulation. It is possible to define an initial grain size distribution for the surface or active layer which may differ from the rest of the alluvium. To do so, one adds the column *SurfaceLayerGrainSizeDistribution* which defines the surface layer grain size distribution file names (without the .txt extension). If this column is not given, the code will use the file names of *StrataGrainSizeDistribution* for the definition of the initial *SurfaceLayerGrainSizeDistribution*.

BedrockRoughnessEquivalentRepresentativeGrainDiameterInCM defines the flow resistance and hiding properties of a reach, when the river runs over bedrock and there is no grain size distribution left to define the properties of the reach. For example, if a flow resistance relation is based on the 84^{th} percentile grain diameter D_{84}, the values in this column will be used as D_{84} for this flow resistance relation. It is advisable to select values close to the measured grain diameters in this area. If the alluvium cover gets thin, the local grain sizes will tendentially approach the value of the bedrock roughness. If the column *BedrockRoughnessEquivalentRepresentativeGrainDiameterInCM* is not given, the code will use the D_{84} of the grain size distributions given in the *SurfaceLayerGrainSizeDistribution*.

Table 7: Complete structure of BranchXProfile.txt (part 1)

KilometrageUpstreamDirected	ElevationInM	ChannelWidthInM	AlluviumThicknessInM	$\cdots$
???	???	???	4000.0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 8: Complete structure of BranchXProfile.txt (part 2) including default values

...	Strata-GrainSizeDistribution	SurfaceLayer-GrainSizeDistribution	BedrockRoughnessEquivalent-RepresentativeGrainDiameterInCM
...	???	file name of StrataGrainSizeDistribution	D_{84} of SurfaceLayerGrainSizeDistribution
⋮	⋮	⋮	⋮

10 Sediment input

10.1 Transport capacity input

The transport capacity input is the default way to feed sediment to the system. At the start of a simulation, the reaches at the upstream ends of the simulated river system are copied to create virtual margin reaches just outside the boundaries of the simulated system. The elevation, grain size distribution and alluvium thickness of these margin cells are kept constant for the complete simulation. Within these margin cells, bedload transport capacity is calculated corresponding to the grain size distribution of its active layer, its width & slope and the current discharge. The calculated bedload tranport capacity is then fed to the simulated river system. Therefore it is recommended to define the most upstream reaches in a way to be representative for the transport system of the headwaters, which are not simulated. If no sediment is to be fed to the system, it is recommended to set the alluvium thickness to zero in the corresponding topmost reach. (A value close to zero will not do the job, as alluvium thickness is not eroded in the virtual margin reaches.)

10.2 Sedigraph input

In order to input sediment independent of the discharge, one can define sedigraphs. For this, the optional file *SedimentInputs.txt* is put into the *DischargeAndOtherInputs* folder. The file (Tabs. 9 & 10) contains at least the columns *BranchIDs*, *KilometrageUpstreamDirected*, *GrainSizeDistribution* and *SedimentInputTimeSeries*.

Table 9: Structure of SedimentInputs.txt (part 1)

BranchIDs	KilometrageUpstreamDirected	GrainSizeDistribution	...
???	???	???	...
⋮	⋮	⋮	⋮

Table 10: Structure of SedimentInputs.txt (part 2) including default values

...	SedimentInputTimeSeries	ReplacingRegularDepositionRate-InsteadOfAddingToIt	InputIncludingPoreVolume
...	???	false	true
⋮	⋮	⋮	⋮

The *BranchIDs* and *KilometrageUpstreamDirected* define the crosssection, at which the sediment is fed to the system. So if sediment enters the river channel between two crosssections, one would simply input the branch ID and kilometrage of the more downstream profile. If the code cannot match branch ID and kilometrage to an existing profile, it will simply ignore the corresponding row in the *SedimentInputs.txt*. The *GrainSizeDistribution* defines the file name (without the .txt extension) for the grain size distribution of the sediment input. The code will look for the specified files in the *GrainSizeDistributions* folder. Please note that the grain size distribution of the sediment input stays the same for the complete simulation. It will *not* become coarser at high input rates and it will *not* become finer at low input rates. To mimic such behaviour one may either use the transport capacity input as described in section 10.1 or one may feed multiple sediment inputs with different grain size distributions to the same reach. The *SedimentInputTimeSeries* define the file names of the different time series of sediment flux fed to the simulated channel. The code will look for these files in the *DischargeAndOtherInputs* folder. The structure of these time series files (Tab. 11) and the way, in which they are read by the code, correspond to the discharge time series as described in section 7.

Table 11: Structure of sedigraph spreadsheets

ElapsedSeconds	SedimentInputInM3PerS
???	???
⋮	⋮

Please note that the model reads the sediment input rate at the beginning of a time step and keeps this value for the complete time step. This means that the temporal discretisation of the sediment input time series should not be finer than the average time step of the simulation. Any input event which is shorter than the average time step will not be captured properly by this input mechanism.

The sedigraph input is usually added to the material coming from upstream. To modify this, one adds the optional column *ReplacingRegularDepositionRateInsteadOfAddingToIt*. One of the rare applications of this option

is the simulation of a retention basin with a defined downstream sediment yield (which may be zero as well). If the column *ReplacingRegularDepositionRateInsteadOfAddingToIt* is not included in the *SedimentInputs.txt*, its values are set to *false* by default.

The sediment inputs are usually considered to include pore volume. To modify this, one adds the optional column *InputIncludingPoreVolume*. If the column is not given, its values are set to *true* by default.

10.3 Instantaneous sediment pulse input

As mentioned before, sediment input events, which are shorter than the average time step, cannot be represented by sedigraph inputs as described in section 10.2. To account for such events one adds the file *InstantaneousSedimentInputs.txt* to the *DischargeAndOtherInputs* folder. This file contains the columns *BranchIDs*, *KilometrageUpstreamDirected*, *GrainSizeDistribution*, *InputVolumesInM3*, *ElapsedSeconds* and *InputIncludingPoreVolume*, in which *InputIncludingPoreVolume* can be left out to set all its values to *true*. All columns except *InputVolumesInM3* and *ElapsedSeconds* are treated in the way which is described in section 10.2. The *InputVolumesInM3* define the volume of sediment which is to be fed to a certain reach and the *ElapsedSeconds* define the point in time in which this shall happen.

Table 12: Structure of InstantaneousSedimentInputs.txt (part 1)

BranchIDs	KilometrageUpstreamDirected	GrainSizeDistribution	⋯
???	???	???	⋯
⋮	⋮	⋮	⋮

Table 13: Structure of InstantaneousSedimentInputs.txt (part 2) including default values

⋯	InputVolumesInM3	ElapsedSeconds	InputIncludingPoreVolume
⋯	???	???	true
⋮	⋮	⋮	⋮

11 Sills

The large scale effects of sills i.e. the dissipation of energy can be considered within *sedFlow* simulations. A sill is represented by the fixed elevation of the top edge of the sill and the variable elevation of the river bed just downstream

of the sill (disregarding local effects such as the stilling pool). The slope upstream of the sill is calculated using the elevation difference between the next upstream cross section and the maximum of the fixed sill top edge and the variable bed just downstream of the sill. The maximum function allows for the consideration of the burial of sills. The slope downstream of the sill is calculated using the elevation difference between the variable bed just downstream of the sill and the next downstream cross section. To simulate a series of sills with distances between them, which are smaller than the ideal distance of cross sections, it is recommended to substitute the series of sills by one single sill with an elevation drop equal to the sum of elevation drops of the complete series of sills. As only the large scale effects of sills are considered, the results will be the same and with one single sill it is possible to keep the ideal distance of cross sections. All outputs for a cross section with a sill refer to the variable bed just downstream of the sill. The evolution at the top edge of the sill is not displayed as it can be trivially derived.

To introduce sills one adds the optional *Sills.txt* to the folder *LongitudinalProfile*. The columns *BranchID* and *KilometrageUpstreamDirected* define the reach at which a sill should be introduced. If no existing reach can be found to match the branch ID and kilometrage, the corresponding sill entry is ignored. The *SillTopEdgeElevationInM* gives the absolute elevation of the top edge of the sill. It should equal the elevation of the corresponding reach plus the elevation drop of the sill. Alternatively the column *SillDropHeightInM* can be given. In this case, the elevation of the sill top edge is calculated as the sum of the drop height and the elevation of the corresponding reach. If both columns are given, the values from the *SillTopEdgeElevationInM* are used. Please note that the top edge elevation is fixed, while the drop height may change in the course of a simulation as the crosssection at the sill may experience erosion or accumulation of sediment. The sill hydraulics are calculated according to the so called Poleni equation:

$$Q = \frac{2}{3} \cdot \mu \cdot \sqrt{2 \cdot g} \cdot w \cdot h^{\frac{3}{2}} \tag{1}$$

In this, Q is discharge; g is gravity acceleration; w is flow width and h is hydraulic head at the sill. The weir coefficient μ is defined by the column *PoleniFactor*. If this column is not given, its values are set to $\sqrt{\frac{1}{3}}$ by default.

The way in which the bed elevation upstream of a sill is updated, may be defined by node *upstreamOfSillsWedgeShapedInsteadOfParallelUpdate* in the main input xml. For details see section 12.2.

12 Complete structure of the main input xml file

The structure of the main input xml as described in section 8 represents only a minimum to run a *sedFlow* simulation. The following sections describe all

Table 14: Structure of Sills.txt including default values

BranchIDs	KilometrageUpstreamDirected	SillTopEdgeElevationInM **or** SillDropHeightInM	PoleniFactor
???	???	???	$\sqrt{\frac{1}{3}}$
⋮	⋮	⋮	⋮

potential nodes and their default values which are inserted, if a node is not given. The summarising illustrations (Figs. 6 to 10) just give one possible example of a complete input xml structure including all default values.

12.1 *overallParameters*

inputUpperBoundaryInsteadOfMeanGrainDiameter is used to switch the way in which the *GrainDiameterInCM* column of the grain size distribution files is interpreted. If it is true (default value), the diameters are interpreted as upper fraction boundaries and the representative fraction diameter is calculated as the mean of the boundaries. If it is false, the diameters given in the grain size distribution files are directly used as representative fraction diameters.

The *lowerDiameterBoundaryForFinestFractionInCM*, which has been already described in section 8, is only needed if *inputUpperBoundaryInsteadOfMeanGrainDiameter* is *true*.

If *inputUpperBoundaryInsteadOfMeanGrainDiameter* is *true*, the *useArithmeticMeanInsteadOfGeometricMeanForFractionGrainDiameters* is used to switch between the geometric (default) and arithmetic mean for the representative fraction diameter.

The *densityWater* and *densitySediment* define the liquid and solid densities in $\frac{kg}{m^3}$ with their default values of 1000.0 and 2650.0 respectively.

The *poreVolumeFraction* defines the volumetric portion of pores and fine (suspension load) sediment with its default value of 0.3.

The *gravityAcceleration* given in $\frac{m}{s^2}$ has a default value of 9.80665.

The *angleOfReposeInDegree* has a default value of 36.0.

The *elapsedSeconds* and *finishSeconds* define the start and end of the simulation. By default they are set to the minimum and maximum of the *ElapsedSeconds* column of the *BranchXDischarge.txt*, with X equal to the smallest used branch ID.

The *courantFriedrichsLewyNumber* is used to determine numerically stable time step lengths. It represents the maximum potential value for the ratio $\frac{v \cdot \Delta t}{\Delta x}$, in which v is velocity and Δx & Δt are spatial & temporal discretisation. The default value for the Courant-Friedrichs-Lewy number is

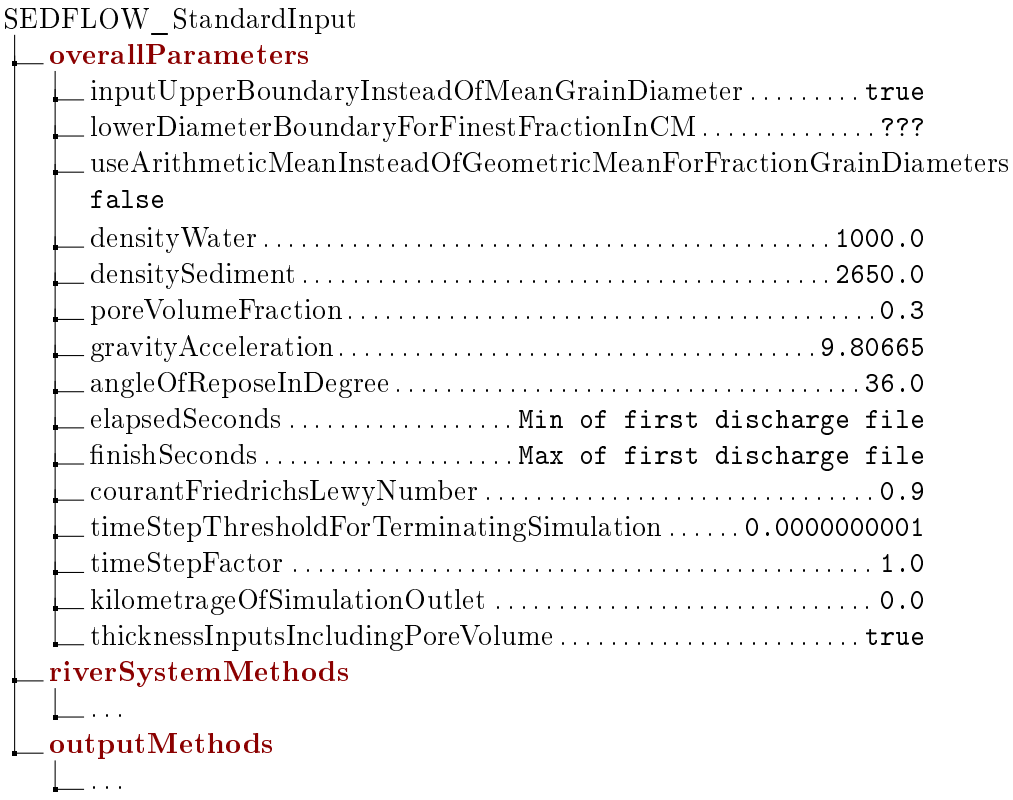

Figure 6: overallParameters including default values.

0.9.

As *sedFlow* is intended for fast simulations with long time steps, the simulation will stop whenever the time step length drops below the *timeStepThresholdForTerminatingSimulation*, as this points to some problems in the current simulation. The default value for this threshold is 0.0000000001.

The *timeStepFactor* may be used to modify the time step lengths, which the model determines based on its criteria for numeric stability. Its default value is 1.0 and it is not recommended to use the option of changing this value.

The length of the lowermost river reach is calculated as the difference between its kilometrage from the *BranchXProfile.txt* and the *kilometrageOfSimulationOutlet*. By default, the simulations start at a kilometrage of 0.0.

The switch *thicknessInputsIncludingPoreVolume* is used to define whether or not thickness inputs such as e.g. the erodible alluvium thickness are considered to include the pore volume. The default value of this node is *true*.

12.2 *riverSystemMethods*

The switch *upstreamOfSillsWedgeShapedInsteadOfParallelUpdate* determines the way, in which the bed elevation upstream of a sill (section 11) is updated. If the switch is *false*, the model assumes that erosion and deposition are equally distributed within the completete reach, which is also assumed in any other reach not affected by a sill. If the switch is *true*, the model assumes that erosion and deposition are distributed in a wedge-shaped way within the reach. This will cause the bed elevation upstream of a sill to adjust twice as fast as in reaches not affected by sills. If the node is not given, its value is set to *false* by default.

The nodes *bedSlopeCalculationMethod*, *waterEnergySlopeCalculationMethod* and *sedimentEnergySlopeCalculationMethod* define the way in which the bed slope and the energy slope for the hydraulic and sediment transport capacity calculations is determined. Different realisations are described in section 13.6. However it is not recommended to change the default values of *bedSlopeCalculationMethod* and *waterEnergySlopeCalculationMethod*. If these nodes are not given, their realisation types are set to the *SimpleDownstreamTwoCellGradient* of *elevation* (section 13.6.1) and the *ReturnBedslope* (section 13.6.4) by default. For *sedimentEnergySlopeCalculationMethod* the realisations *WithFlowResistancePartitioning* or *NoFlowResistancePartitioning* (both section 13.6) are recommended.

The *flowResistance* node defines the way in which average flow velocity and wetted cross section area are determined for a given discharge. Different realisations are described in section 13.1. Please note that for the *ImlicitKinematicWave* flow routing (section 13.2.3) only the *FixedPowerLawFlowResistance* flow resistance (section 13.1.1) can be used. However,

the *VariablePowerLawFlowResistance* (section 13.1.2) is assumed to better represent the processes in steep mountain channels. If the *flowResistance* node is not given, its realisation type is set to *FixedPowerLawFlowResistance* (if the *waterFlowRouting* realisation is *ImplicitKinematicWave*) or to *VariablePowerLawFlowResistance* (in any other case) by default.

The node *bedloadVelocityCalculationMethod* defines the way to estimate grain velocities, which are only needed and used to check for the bedload flux the Courant-Friedrichs-Lewy criterion of numeric stability. Different realisations are described in section 13.4. If the node *bedloadVelocityCalculationMethod* is not given, its realisation is set to *VelocityAsTranpsportRatePerUnitCrossSectionalArea* (section 13.4.1) by default.

The node *tauCalculationMethod* defines the way to estimate the bed shear stress. Different realisations are described in section 13.7. If the node is not given, its realisation is set to *EnergyslopeTau* (section 13.7) by default.

The node *activeWidthCalculationMethod* defines the way to estimate the width in which sediment transport takes place. At the moment *SetActiveWidthEqualFlowWidth* is the only realisation for this node. Alternative methods may be implemented in future versions of *sedFlow*.

The *waterFlowRouting* node defines the way in which water is transferred through the channel system and how the *flowResistance* relation is applied to calculate flow depths and velocities. Different realisations are described in section 13.2. The *ExplicitKinematicWave* (section 13.2.2) and *ImplicitKinematicWave* (section 13.2.3) depend on positive bed slopes. The *ImplicitKinematicWave* (section 13.2.3) further depends on the *FixedPowerLawFlowResistance* flow resistance (section 13.1.1), the approximation of the hydraulic radius by flow depth (section 13.1) and on an infinitely deep rectangular channel, which is given anyway. The *UniformDischarge* (section 13.2.1) has no dependencies and replaces the discharge routing approach by a uniform discharge approach.

The node *bedloadTransportEquations* defines the way in which the bedload transport capacity is estimated. Different realisations are described in section 13.3. Fractional transport as well as representative single grain size approaches can be selected. However, single grain size approaches do not imply an increase in calculation speed, as the model framework is optimised for fractional approaches.

The node *strataSorting* defines the interaction between the surface active layer and the subsurface alluvium. Different realisations are described in section 13.5. If the node *strataSorting* is not given, its realisation is set to *TwoLayerWithShearStressBasedUpdate* (section 13.5.3) by default.

The optional node *additionalMethods* may be added to include further methods, which are not covered by the aforementioned *riverSystemMethods* nodes. Up to now the only *additionalMethods* are concerned with the effects of gravel abrasion (section 13.11).

- SEDFLOW_StandardInput
  - **overallParameters**
    - ...
  - **riverSystemMethods**
    - upstreamOfSillsWedgeShapedInsteadOfParallelUpdate ..... `false`
    - **bedSlopeCalculationMethod**
      - realisationType ..... `SimpleDownstreamTwoCellGradient`
      - propertyOfInterest ..... `elevation`
    - **waterEnergySlopeCalculationMethod**
      - realisationType ..... `ReturnBedslope`
      - minimumSlope ..... `-inf`
    - **sedimentEnergySlopeCalculationMethod**
      - realisationType ..... `WithFlowResistancePartitioning`
      - stressPartitioningExponent ..... `1.5`
      - calculationBasedOnqInsteadOfh ..... `false`
      - maximumFroudeNumber ..... `value from flowResistance`
      - minimumInputSlope ..... `???`
      - ensureMinimumInputSlope ..... `if minimumInputSlope node exists`
    - **flowResistance**
      - startingValueForIteration ..... `0.4`
      - accuracyForTerminatingIteration ..... `0.001`
      - maximumNumberOfIterations ..... `400`
      - typeOfNumericRootFinder ..... `RiddersMethod`
      - useApproximationsForHydraulicRadius `If waterFlowRouting == ImplicitKinematicWave`
      - maximumFroudeNumber ..... `4.0`
      - minimumHydraulicSlope ..... `0.0004`
      - realisationType ..... `VariablePowerLawFlowResistance`
    - **bedloadVelocityCalculationMethod**
      - realisationType`VelocityAsTransportRatePerUnitCrossSectionalArea`
      - **estimateThicknessOfMovingSedimentLayer**
        - realisationType`ConstantThicknessOfMovingSedimentLayer`
        - constantThickness ..... `0.7`
    - **tauCalculationMethod**
      - realisationType ..... `EnergyslopeTau`
      - correctionForBedloadWeightAtSteepSlopes ..... `true`
    - **activeWidthCalculationMethod**
      - realisationType ..... `SetActiveWidthEqualFlowWidth`
    - ...
  - **outputMethods**
    - ...

Figure 7: riverSystemMethods part 1 including default values.

- SEDFLOW_StandardInput
  - **overallParameters**
    - ...
  - **riverSystemMethods**
    - ...
    - **waterFlowRouting**
      - realisationType ..... `UniformDischarge`
      - maximumTimeStep ..... `900`
    - **bedloadTransportEquations**
      - maximumFractionOfActiveLayerToBeEroded ..... `0.9`
      - preventZeroOrNegativeBedSlopes ... `if waterFlowRouting == some KinematicWave`
      - maximumRelativeTwoCellBedSlopeChange ..... `0.9`
      - realisationType ... `RickenmannBedloadCapacityBasedOnTheta`
      - useOnePointOneAsExponentForFroudeNumber ..... `false`
      - simplifiedEquation ..... `true`
      - thetaCriticalBasedOnConstantSred ..... `false`
      - **thresholdCalculationMethod**
        - realisationType ..... `LambEtAlCriticalTheta`
        - minimumCriticalDimensionlessShearStress ..... `0.03`
        - correctionForBedloadWeightAtSteepCounterSlopes .... `true`
      - **hidingFactorsCalculationMethod**
        - realisationType ..... `PowerLawHidingFunction`
        - referenceDiameterPercentile ..... `50.0`
        - exponent ..... `-0.8`
    - **strataSorting**
      - realisationType ..... `TwoLayerWithShearStressBasedUpdate`
      - dynamicLayerThickness ..... `false`
      - useInitialGrainSizesForConstantLayerThickness ..... `true`
      - layerThicknessFactor ..... `1.75`
      - referenceGrainSizePercentile ..... `84.0`
      - layerThickness ..... `only needed if neither dynamicLayerThickness nor useInitialGrainSizesForConstantLayerThickness`
      - dynamicBreakUpConditions ..... `false`
      - usePredefinedBreakUpConditions ..... `false`
      - referenceMedianDiameter ..... `only needed if usePredefinedBreakUpConditions`
      - thetaCriticalForActiveLayer ..... `only needed if usePredefinedBreakUpConditions`
      - thetaCriticalForSublayer ..... `only needed if usePredefinedBreakUpConditions`
      - **thresholdCalculationMethod** ..... `method from bedloadTransportEquations or default displayed there`
    - **additionalMethods** ..... `optional`
  - **outputMethods**
    - ...

Figure 8: riverSystemMethods part 2 including default values.

12.3 *outputMethods*

12.3.1 Standard and regular outputs

General remarks Within *sedFlow* the format of the output files can be defined globally by the standard output properties. Alternatively single output files can be defined individually by the regular output nodes. In both cases the nodes are the same. For the standard output properties the term "Standard" is added to the node name and the nodes are direct child nodes of the *OutputMethods* node. For individual files the node *regularOutput* or the nodes *regularOutputX* may be added to the *OutputMethods*. In this, *X* is replaced by an integer number starting at 1 and going up to the desired number of files. If there is a break in the sequence of *X* values, the following nodes will be ignored. The nodes for the formatting of output files are described in the following:

Output file formatting The node *forVisualInterpretation* is used to switch between the usual output format for numeric postprocessing using other programmes and a version for a more visual and by-hand postprocessing workflow. The usual format has a single line header and no additional formatting symbols. The *forVisualInterpretation* version has a header which consists of several rows and has extra columns with formatting symbols, which separate the individual properties and reaches. This format is useful, when different properties for only a few reaches are written to a single file, which shall be evaluated in a by-hand postprocessing.

The *precisionForOutput* defines the precision for the output of floating point numbers. Please note that floating point numbers are output in scientific format convention. For long simulation times, make sure that the output precision is sufficient to discriminate the different *ElapsedSeconds* values.

By *outputTimeStepLength* an additional column with the current time step length in seconds may be added to the output file.

By *outputInitialValues* an output row with the values at the start of the simulation is added. This makes sense for some state properties like elevation or grain diameter percentiles. For other properties like e.g. the transport rate, which are calculated in the course of a simulation, dummy values will be printed.

As described in section 10.1, the code internally creates virtual margin reaches at the upstream ends and at the downstream end of the simulated system. These margin reaches are used to feed inputs to the system and to recieve the outflux at the downstream end. The *printUpstreamMargins* and the *printDownstreamMargin* switches may be used to add these virtual margin reaches to the output file. This option is mainly intended for debugging.

Output timing The *explicitTimesForOutput* may contain a series of child nodes, which define certain points in time (expressed as elapsed seconds), at which an output row shall be written. The names of the child nodes are ignored.

The *outputInterval* defines the interval in seconds, at which output rows shall be written.

The *writeLineEachTimeStep* may be used to enforce the output of every simulated time step. This option is intended for debugging and not recommended for daily use.

In *sedFlow* it is possible to use a secondary output interval, which is applied, when some property like e.g. discharge exceeds a certain threshold. This may be used to e.g. increase output frequency during floods. For the standard, the secondary interval is used, when one of the four nodes *referenceCellIDStandard*, *referencePropertyStandard*, *thresholdToBeExceededStandard* or *secondaryOutputIntervalStandard* is given. For individual outputs the secondary interval is used, when the node *SecondaryOutputInterval* is given, which contains the four mentioned nodes. The node *referenceProperty* defines the property, for which it is checked whether it exceeds some threshold. The node *referenceCellID* defines the ID of the reach, at which it is checked whether the reference property exceeds a threshold. The node *thresholdToBeExceeded* defines the threshold, which needs to be exceeded for the application of the secondary output interval. Finally, the *secondaryOutputInterval* defines the output interval, which is applied whenever the threshold is exceeded.

Selecting output reaches and properties The node *reachIDsForOutput* is used to select the reaches, for which output should be written to files. This node contains child nodes with the possible names *reachID* and *branchID*. The branch IDs are used as defined in the *branchTopology.txt*. The reach IDs are used as defined in the *reachIDExplanations.txt*, which is written to the *Output* folder by the model. If the node *reachIDsForOutput* is not given, the model will produce outputs for all reaches by default.

The node *regularRiverReachPropertiesForOutput* defines for which properties the values shall be written to the output files. The node contains child nodes, the names of which define the output properties. With the standard outputs, the code will create one file for each property, if the switch *createStandardOutputs* is *true* (default). For the individual outputs all properties will be written into one file. Potential values for the *regularRiverReachPropertiesForOutput* are the following:

- *elevation*
- *sillOccurence*
- *discharge*
- *length*
- *sillTopEdgeElevation*
- *flowVelocity*

- *maximumWaterdepth*
- *waterLevel*
- *hydraulicHead*
- *kineticComponentOfHydraulicHead*
- *bedShearStress*
- *activeWidth*
- *strataPerUnitBedSurface*
- *strataPerUnitBedSurfaceIncludingPoreVolume*
- *alluviumThicknessIncludingPoreVolume*
- *bedrockElevation*
- *bedslope*
- *waterEnergyslope*
- *sedimentEnergyslope*
- *unreducedSedimentEnergyslope*
- *waterVolumeChangeRate*
- *waterVolumeChange*

- *activeLayerPerUnitBedSurface*
- *activeLayerPerUnitBedSurfaceIncludingPoreVolume*
- *activeLayerPerUnitBedSurfaceOverallVolume*
- *activeLayerPerUnitBedSurfaceOverallVolumeIncludingPoreVolume*
- *activeLayerPerUnitBedSurfaceMedianGrainDiameter*
- *activeLayerPerUnitBedSurfaceArithmeticMeanGrainDiameter*
- *activeLayerPerUnitBedSurfaceGeometricMeanGrainDiameter*
- *activeLayerPerUnitBedSurfaceD20*
- *activeLayerPerUnitBedSurfaceD30*
- *activeLayerPerUnitBedSurfaceD50*
- *activeLayerPerUnitBedSurfaceD84*
- *activeLayerPerUnitBedSurfaceD90*

- *erosionRate*
- *erosionRateIncludingPoreVolume*
- *erosionRateOverallVolume*
- *erosionRateOverallVolumeIncludingPoreVolume*
- *erosionRateMedianGrainDiameter*
- *erosionRateArithmeticMeanGrainDiameter*
- *erosionRateGeometricMeanGrainDiameter*
- *erosionRateD20*
- *erosionRateD30*
- *erosionRateD50*
- *erosionRateD84*
- *erosionRateD90*

- *depositionRate*
- *depositionRateIncludingPoreVolume*
- *depositionRateOverallVolume*
- *depositionRateOverallVolume-IncludingPoreVolume*
- *depositionRateMedianGrainDiameter*
- *depositionRateArithmeticMean-GrainDiameter*
- *depositionRateGeometricMean-GrainDiameter*
- *depositionRateD20*
- *depositionRateD30*
- *depositionRateD50*
- *depositionRateD84*
- *depositionRateD90*

- *erosion*
- *erosionIncludingPoreVolume*
- *erosionOverallVolume*
- *erosionOverallVolume-IncludingPoreVolume*
- *erosionMedianGrainDiameter*
- *erosionArithmeticMean-GrainDiameter*
- *erosionGeometricMean-GrainDiameter*
- *erosionD20*
- *erosionD30*
- *erosionD50*
- *erosionD84*
- *erosionD90*

- *deposition*
- *depositionIncludingPoreVolume*
- *depositionOverallVolume*
- *depositionOverallVolume-IncludingPoreVolume*
- *depositionMedianGrainDiameter*
- *depositionArithmeticMean-GrainDiameter*
- *depositionGeometricMean-GrainDiameter*
- *depositionD20*
- *depositionD30*
- *depositionD50*
- *depositionD84*
- *depositionD90*

- *erosionPerUnitBedSurface*
- *erosionPerUnitBedSurface-IncludingPoreVolume*
- *erosionPerUnitBedSurface-*

OverallVolume

- *erosionPerUnitBedSurface-OverallVolumeIncludingPoreVolume*
- *erosionPerUnitBedSurface-MedianGrainDiameter*
- *erosionPerUnitBedSurface-ArithmeticMeanGrainDiameter*
- *erosionPerUnitBedSurface-GeometricMeanGrainDiameter*
- *erosionPerUnitBedSurfaceD20*
- *erosionPerUnitBedSurfaceD30*
- *erosionPerUnitBedSurfaceD50*
- *erosionPerUnitBedSurfaceD84*
- *erosionPerUnitBedSurfaceD90*

- *depositionPerUnitBedSurface*
- *depositionPerUnitBedSurface-IncludingPoreVolume*
- *depositionPerUnitBedSurface-OverallVolume*
- *depositionPerUnitBedSurface-OverallVolumeIncludingPoreVolume*
- *depositionPerUnitBedSurface-MedianGrainDiameter*
- *depositionPerUnitBedSurface-ArithmeticMeanGrainDiameter*
- *depositionPerUnitBedSurface-GeometricMeanGrainDiameter*
- *depositionPerUnitBedSurfaceD20*
- *depositionPerUnitBedSurfaceD30*
- *depositionPerUnitBedSurfaceD50*
- *depositionPerUnitBedSurfaceD84*
- *depositionPerUnitBedSurfaceD90*

The *erosionRate* properties display the potential local transport capacity in $\frac{m^3}{s}$ not considering supply limitations. The *erosion* properties display the actual local transport in m^3 considering supply limitations. For the actual local transport rate, the *erosion* properties have to be divided by the current time step length. The *erosionPerUnitBedSurface* properties display the *erosion* properties normalised to a column of $1m^2$ base area. The *depositionRate* and *deposition* properties are usually equal to the *erosionRate* and *erosion* properties of the upstream reach. If there are several upstream neighbours, the *depositionRate* and *deposition* properties display the sum of the upstream values. Local *depositionRate* or *deposition* may differ from the upstream *erosionRate* or *erosion*, if the effects of gravel abrasion (section 13.11) are simulated.

Output file naming If an output file contains only one property, the file will have the name of this property by default. If a file contains several properties, it will be named regularOutput or regularOutputX by default

corresponding to the name of its node. These default names may be overridden by the node *name*. Please note that if a file name (default or user defined) is used several times, several output routines will write to the same file creating cryptic artefacts.

outputAccumulatedBedloadTransport Within the regular and standard output files, all values represent a snapshot of the property at a specified point in time. That is, the values are *not* integrated over an output interval or the complete simulation time. However, the important descriptive variable of the accumulated bedload transport is defined as the temporal integral of the local transport rates. Therefore, the *regularOutputX* nodes are complemented by the *outputAccumulatedBedloadTransportX* nodes, which may be used to output the temporal integral of the bedload transport rates. (Within the *regularRiverReachPropertiesForOutputStandard* the pseudo property *accumulatedBedloadTransport* may be used to create an output file corresponding to *outputAccumulatedBedloadTransportX*.) The *outputAccumulatedBedloadTransportX* contains the following child nodes, which are used as described for *regularOutputX*: *explicitTimesForOutput*, *outputInterval*, *precisionForOutput*, *reachIDsForOutput* and *writeLineEachTimeStep*. The switch *outputIncludingPoreVolume* defines whether the displayed volumes shall include pore volume or just represent the solid material. The default for *outputIncludingPoreVolume* is *true*, as any volumetric data recorded in the field commonly includes the pore volume as well. By default, the *outputAccumulatedBedloadTransport* will create a file with the overall transported volumes. The switch *outputDetailedFractional* is used to create an additional file, in which the transported volumes are specified for each grain size fraction separately. To discriminate these two types of files, the suffixes "ABT-OverallVolume" or "ABT-DetailedFractional" are appended to the (by default empty) file name, which is specified by the node *name*.

12.3.2 *backupXML*

For very long simulations, it is possible to create backups of the current state of a running simulation. To do so, one adds the node *backupXML* to the *outputMethods*. The child nodes *explicitTimesForOutput*, *outputInterval* and *precisionForOutput* are used as described for standard and regular outputs in section 12.3.1. By default, any new backup will replace the previous one. To change this, one sets *overwriteFiles* to *false*. In this case an ID will be appended to the file name. The number of digits of this ID can be changed using the *numberOfFileIDDigits* node. For multiple backups it may be worth to redirect these files from the standard *Output* folder, where they are stored by default, to some other location, which is given in the node *alternativePathForXMLBackupOutputs*. To restart a simulation from a backup, one simply starts *sedFlow* and imports the backup file instead of

the normal main input xml (section 8). In general, it is not recommended to create backup files, as this considerably slows down a simulation.

12.3.3 *outputSimulationSetup*

By default the model creates an easy to read summary of the current simulation setup. To supress this output, one sets the node *notOutputSimulationSetup* to *true*. The format of the output can be defined within the *outputSimulationSetup* node. The child node *precisionForOutput* is used as described for standard and regular outputs in section 12.3.1. The selection and order of the displayed setup properties is defined in the *setupPropertiesForOutput*. The default values are *CalcBedloadCapacity*, *FlowResistance*, *CalcGradient* and *StrataSorting*. Other potential values are *CalcTau* and *CalcActiveWidth*. The file name can be changed using the node *name*. The *simulationID* and *simulationName* can be used to include any user defined text to the output file. The switch *printStartingTime* defines whether the starting time of the simulation shall be included to the file and the switch *printModelVersion* selects whether the compilation date of the used model binary shall be included to the file as well. By default the value of both switches is *true*.

SEDFLOW_StandardInput

- **overallParameters**
  - ...
- **riverSystemMethods**
  - ...
- **outputMethods**
  - createStandardOutputs ..... `true`
  - forVisualInterpretationStandard ..... `false`
  - explicitTimesForOutputStandard ..... `empty`
  - outputIntervalStandard ..... `3600.0`
  - precisionForOutputStandard ..... `4`
  - writeLineEachTimeStepStandard ..... `false`
  - outputTimeStepLengthStandard ..... `false`
  - outputInitialValuesStandard ..... `true`
  - printUpstreamMarginsStandard ..... `false`
  - printDownstreamMarginStandard ..... `false`
  - **reachIDsForOutputStandard** ..... `all reach IDs`
  - useSecondaryOutputInterval .... `if one of the following four nodes exists`
  - referenceCellIDStandard ..... `???`
  - referencePropertyStandard ..... `???`
  - thresholdToBeExceededStandard ..... `???`
  - secondaryOutputIntervalStandard ..... `???`
  - **regularRiverReachPropertiesForOutputStandard**
    - elevation
    - activeLayerPerUnitBedSurfaceD50
    - activeLayerPerUnitBedSurfaceD84
    - bedslope
    - maximumWaterdepth
    - accumulatedBedloadTransport
  - notOutputSimulationSetup ..... `false`
  - **outputSimulationSetup**
    - name ..... `SimulationSetup.txt`
    - precisionForOutput ..... `standard value`
    - simulationID ..... `empty`
    - simulationName ..... `empty`
    - printStartingTime ..... `true`
    - printModelVersion ..... `true`
    - **setupPropertiesForOutput**
      - CalcBedloadCapacity
      - FlowResistance
      - CalcGradient
      - StrataSorting
  - ...

Figure 9: outputMethods standard including default values.

SEDFLOW_StandardInput
- **overallParameters**
  - ...
- **riverSystemMethods**
  - ...
- **outputMethods**
  - ...
  - **regularOutputX**
    - forVisualInterpretation ..... `standard value`
    - **regularRiverReachPropertiesForOutput**
      - ...
    - name ..... `name of PropertyForOutput or regularOutputX`
    - explicitTimesForOutput ..... `standard value`
    - outputInterval ..... `standard value`
    - precisionForOutput ..... `standard value`
    - **reachIDsForOutput** ..... `standard value`
    - writeLineEachTimeStep ..... `standard value`
    - outputTimeStepLength ..... `standard value`
    - outputInitialValues ..... `standard value`
    - printUpstreamMargins ..... `standard value`
    - printDownstreamMargin ..... `standard value`
    - **SecondaryOutputInterval**
      - referenceCellID ..... `standard value`
      - referenceProperty ..... `standard value`
      - thresholdToBeExceeded ..... `standard value`
      - secondaryOutputInterval ..... `standard value`
  - **outputAccumulatedBedloadTransportX**
    - name ..... `empty`
    - explicitTimesForOutput ..... `standard value`
    - outputInterval ..... `standard value`
    - precisionForOutput ..... `standard value`
    - reachIDsForOutput ..... `standard value`
    - writeLineEachTimeStep ..... `standard value`
    - outputIncludingPoreVolume ..... `true`
    - outputDetailedFractional ..... `false`
  - **backupXML**
    - alternativePathForXMLBackupOutputs ..... `empty`
    - explicitTimesForOutput ..... `standard value`
    - outputInterval ..... `standard value`
    - precisionForOutput ..... `standard value`
    - overwriteFiles ..... `true`
    - numberOfFileIDDigits ..... `4`

Figure 10: outputMethods regular including default values.

13 Realisations

13.1 *flowResistance* realisations

flowResistance methods are intended to determine the $\sqrt{\frac{8}{f}}$ in the following equation:

$$\sqrt{\frac{8}{f}} = \frac{v}{\sqrt{g \cdot r_h \cdot S_f}} \tag{2}$$

In this, f is the Darcy-Weisbach friction factor; v is flow velocity; g is gravitational acceleration; r_h is hydraulic radius and S_f is friction slope.

To deal with numeric issues all *flowResistance* realisations (Fig. 11) have the following optional nodes: *startingValueForIteration*, *accuracyForTerminatingIteration*, *maximumNumberOfIterations* and *typeOfNumericRootFinder*. Potential values for *typeOfNumericRootFinder* are *BisectionMethod*, *SecantMethod*, *FalsePositionMethod* and *RiddersMethod*, among which the *RiddersMethod* is recommended.

The switch *useApproximationsForHydraulicRadius* defines whether the hydraulic radius should be approximated by flow depth. By default, it is *true*, if *ImplicitKinematicWave* is selected as *waterFlowRouting*, and *false* in any other case.

If *UniformDischarge* is selected as *waterFlowRouting*, the code will ensure that the Froude number will not exceed a maximum and that the hydraulic slope will not fall below a minimum. The respective values are defined in the nodes *maximumFroudeNumber* and *minimumHydraulicSlope*.

flowResistance
- startingValueForIteration ..... `0.4`
- accuracyForTerminatingIteration ..... `0.001`
- maximumNumberOfIterations ..... `400`
- typeOfNumericRootFinder ..... `RiddersMethod`
- useApproximationsForHydraulicRadius ... `If waterFlowRouting == ImplicitKinematicWave`
- maximumFroudeNumber ..... `4.0`
- minimumHydraulicSlope ..... `0.0004`
- ...

Figure 11: flowResistance including default values.

13.1.1 *FixedPowerLawFlowResistance*

The *FixedPowerLawFlowResistance* (Fig. 12) has four optional nodes, which can be derived from the following equation:

$$\sqrt{\frac{8}{f}} = j \cdot \left(\frac{r_h}{k \cdot D_x}\right)^l \tag{3}$$

The value of j is defined by the node *factor* with its default value of 6.5. The value of k is defined by the node *grainsFactor* with its default value of 1.0. D_x is the x'th percentile grain diameter with the value of x defined by the node *grainsPercentile* with its default value of 84.0. The value of l is defined by the node *exponent* with its default value of 0.166666667.

flowResistance
- ...
- realisationType ..... `FixedPowerLawFlowResistance`
- factor ..... `6.5`
- grainsFactor ..... `1.0`
- grainsPercentile ..... `84.0`
- exponent ..... `0.166666667`

Figure 12: FixedPowerLawFlowResistance including default values.

If one uses the default value of $\sim \frac{1}{6}$ for the *exponent* l, this method corresponds to a Manning-Strickler flow resistance relation with:

$$k_{st} = \frac{1}{n} = \frac{j \cdot \sqrt{g}}{(k \cdot D_x)^{\frac{1}{6}}}, \tag{4}$$

in which k_{st} is the Strickler roughness coefficient and n is the Manning roughness coefficient.

If one uses all default values, this method corresponds to a Manning-Strickler flow resistance relation with:

$$k_{st} = \frac{1}{n} = \frac{6.5 \cdot \sqrt{g}}{D_{84}^{\frac{1}{6}}} \tag{5}$$

13.1.2 *VariablePowerLawFlowResistance*

The *VariablePowerLawFlowResistance* (Fig. 13) implements the following equation from Ferguson [2] with the parameter values by Rickenmann and Recking [14] and has no specific additional nodes.

$$\sqrt{\frac{8}{f}} = \frac{6.5 \cdot 2.5 \cdot \frac{r_h}{D_{84}}}{\sqrt{6.5^2 + 2.5^2 \cdot \left(\frac{r_h}{D_{84}}\right)^{\frac{5}{3}}}} \tag{6}$$

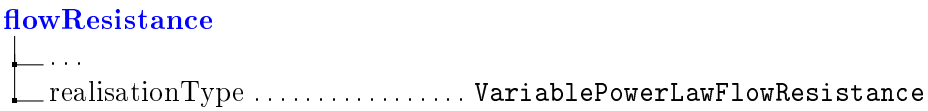

Figure 13: VariablePowerLawFlowResistance.

13.2 *waterFlowRouting* realisations

It has to be noted that the *UniformDischarge* approach (section 13.2.1) in its default combination with some sediment energy slope based on *hydraulicHead* (section 13.6) is based on the assumption that the simulated system only consists of the two extreme cases of pondages (friction slope approximately zero) on the one hand and situations of parallel slopes (friction slope approximately equal bedslope) on the other. (For details please refer to Heimann et al. [3].) It will produce large errors when intermediate cases of moderate backwater effects are part of the simulated system. In such systems, a *KinematicWave* approach, which uses bedslope both as friction slope for the hydraulic and as energy slope for the sediment transport calculations, will produce better estimates of the transported sediment volumes, but requires the absence of adverse channel gradients.

13.2.1 *UniformDischarge*

The only optional node of *UniformDischarge* (Fig. 14) is *maximumTimeStep* with its default value of 900, which defines the maximum time step length that will not be exceeded during this simulation. It is recommended that this value should be smaller than the time step length of the input time series (e.g. discharge).

Figure 14: UniformDischarge including default values.

13.2.2 *ExplicitKinematicWave*

The *ExplicitKinematicWave* (Fig. 13.2.2) has no specific nodes.

Figure 15: ExplicitKinematicWave.

13.2.3 *ImplicitKinematicWave*

The *ImplicitKinematicWave* (Fig. 16) performs an implicit kinematic wave routing using the algorithms of Liu and Todini [8]. Just like *UniformDischarge*, the *ImplicitKinematicWave* has the optional node *maximumTimeStep* with its default value of 900. Additionally it has the optional node *checkForCourantFriedrichsLewy* with its default value *false*, which can be used to switch on a test for the Courant-Friedrichs-Lewy criterion based on the water flow velocity.

waterFlowRouting
- realisationType ..... `ImplicitKinematicWave`
- maximumTimeStep ..... `900`
- checkForCourantFriedrichsLewy ..... `false`

Figure 16: ImplicitKinematicWave including default values.

13.3 *bedloadTransportEquations* realisations

All *bedloadTransportEquations* realisations estimate the dimensionless bedload transport rate Φ_b defined by the following equation:

$$\Phi_b = \frac{q_b}{\sqrt{\left(\frac{\rho_s}{\rho} - 1\right) \cdot g \cdot D^3}}, \tag{7}$$

in which q_b is bedload flux per unit flow width; ρ is fluid density; ρ_s is sediment density; g is gravity acceleration and D is grain diameter.

For all *bedloadTransportEquations* (Fig. 17), the node *maximumFractionOfActiveLayerToBeEroded* with its default value of 0.9 defines some sort of vertical Courant-Friedrichs-Lewy criterion for erosion.

If a kinematic wave flow routing is selected, the code will prevent zero or negative bed slopes. The switch *preventZeroOrNegativeBedSlopes* is used to force this option on or off. If zero or negative bed slopes are to be prevented, the code will only allow changes of the bed slope which are a fraction of its current value. This fraction is defined in the node *maximumRelativeTwoCellBedSlopeChange* with its default value of 0.9. Please note that the prevention of zero or negative bed slopes will extremely slow down simulations with bed slopes close to zero.

bedloadTransportEquations

- maximumFractionOfActiveLayerToBeEroded ..... `0.9`
- preventZeroOrNegativeBedSlopes ... `if waterFlowRouting == someKinematicWave`
- maximumRelativeTwoCellBedSlopeChange ..... `0.9`
- ...

Figure 17: bedloadTransportEquations including default values.

13.3.1 Rickenmann bedload capacity

For the bedload transport equation of Rickenmann, there exists a version based on dimensionless shear stress θ [12]:

$$\Phi_b = 3.1 \cdot \left(\frac{D_{90}}{D_{30}}\right)^{0.2} \cdot \sqrt{\theta} \cdot (\theta - \theta_c) \cdot Fr \cdot \frac{1}{\sqrt{\frac{\rho_s}{\rho} - 1}} \tag{8}$$

... and a version based on discharge per unit flow width q [12]:

$$q_b = 3.1 \cdot \left(\frac{\rho_s}{\rho} - 1\right)^{-1.5} \cdot \left(\frac{D_{90}}{D_{30}}\right)^{0.2} \cdot (q - q_c) \cdot S^{1.5}; \tag{9a}$$

$$q_c = 0.065 \cdot \left(\frac{\rho_s}{\rho} - 1\right)^{1.67} \cdot \sqrt{g} \cdot D_{50}^{1.5} \cdot S^{-1.12}; \tag{9b}$$

In these equations Fr is the Froude number; q is the discharge per unit flow width; q_c is the discharge per unit flow width threshold for the initiation of bedload motion; S is slope and D_{50} is the median diameter of the local grain size distribution.

Additionally, non fractional versions of these equations are implemented. In these versions, transport capacity is estimated based on the median diameter D_{50} and the local grain size distributions stay constant in the course of a simulation. Please note that *sedFlow* is optimised for fractional transport. Therefore, the use of the non fractional version does not speed up simulations.

So in total, there are four implementations of the Rickenmann equation: *RickenmannBedloadCapacityBasedOnTheta*, *RickenmannBedloadCapacityBasedOnThetaNonFractional*, *RickenmannBedloadCapacityBasedOnq*, *RickenmannBedloadCapacityBasedOnqNonFractional*.

For the *BasedOnTheta* versions (Figs. 18 & 19), the node *useOnePointOneAsExponentForFroudeNumber* defines whether 1.0 (default) or 1.1 should be used as exponent for the Froude number. The value of 1.1 corresponds to the initial version of the equation, but is commonly not used any more. The node *thresholdCalculationMethod* defines the method by which the threshold for

the initiation of bedload motion θ_c is calculated. The default method is *LambEtAlCriticalTheta* (section 13.8.2). The switch *simplifiedEquation* defines whether the following simplified version of equation 8 based on Rickenmann [12] is to be used (default) or not:

$$\Phi_b = 2.5 \cdot \sqrt{\theta} \cdot (\theta - \theta_c) \cdot Fr \tag{10}$$

For consistency, the following $\theta_{c,r}$ is used in bedload transport calculations, if θ is based on S_{red} from equation 22d.

$$\theta_{c,r} = \theta_c \cdot \gamma \tag{11}$$

The optional switch *thetaCriticalBasedOnConstantSred* is used to select one of the following two alternatives for the calculation of the correction factor γ. If *thetaCriticalBasedOnConstantSred* is *false* (default), equation 12a is used. If it is *true*, equation 12b is used.

$$\gamma = \frac{S_{red}}{S} \tag{12a}$$

$$\gamma = \frac{S_c}{S} \tag{12b}$$

In the first approach (equation 12a), $\theta_{c,r}$ varies with discharge, as it depends on S_{red} (equation 22d), which in turn is a function of r_h. In the second approach (equation 12b) suggested by Nitsche et al. [9], $\theta_{c,r}$ is independent of discharge. The value of S_c is calculated using equations 22a to 22d, with the value of r_h replaced by the critical hydraulic radius $r_{h,c}$:

$$r_{h,c} = \theta_c \cdot \left(\frac{\rho_s}{\rho} - 1 \right) \cdot D_{50} \cdot \frac{1}{S} \tag{13}$$

For the *BasedOnq* versions (Figs. 20 & 21), the switch *correctionForBedloadWeightAtSteepSlopes* selects whether the following correction of energy slopes according to Rickenmann [13] is to be applied (default) or not.

$$S_k = S \cdot \frac{\sin(\phi_r)}{\sin(\phi_r - S_b)} \tag{14}$$

In this S_k is the energy slope corrected for the effects of bedload weight at steep bed slopes; S is the uncorrected slope; ϕ_r is the angle of repose and S_b is the bed slope.

For the fractional versions (Figs. 18 & 20), the node *hidingFactorsCalculationMethod* defines the hiding function to be used. The default hiding function is the *PowerLawHidingFunction* (section 13.9.1).

For the *NonFractional* versions (Figs. 19 & 21), the switch *takeArmourLayerIntoAccount* selects whether the following correction of the discharge threshold for the initiation of bedload motion according to Badoux and Rickenmann [1] should be applied or not (default).

$$q_{cA} = q_c \cdot \left(\frac{D_{90}}{D_{mArith}} \right)^{\frac{10}{9}} \tag{15}$$

In this q_c is the threshold discharge per unit flow width for the initiation of bedload motion, q_{cA} is q_c corrected for the effect of bed armouring and D_{mArith} the arithmetic mean diameter of the local grain size distribution.

bedloadTransportEquations
- ...
- realisationType ..... `RickenmannBedloadCapacityBasedOnTheta`
- useOnePointOneAsExponentForFroudeNumber ..... `false`
- simplifiedEquation ..... `true`
- **thresholdCalculationMethod**
  - realisationType ..... `LambEtAlCriticalTheta`
  - ...
- **hidingFactorsCalculationMethod**
  - realisationType ..... `PowerLawHidingFunction`
  - ...
- thetaCriticalBasedOnConstantSred ..... `false`

Figure 18: RickenmannBedloadCapacityBasedOnTheta including default values.

bedloadTransportEquations
- ...
- realisationType`RickenmannBedloadCapacityBasedOnThetaNonFractional`
- useOnePointOneAsExponentForFroudeNumber ..... `false`
- simplifiedEquation ..... `true`
- takeArmourLayerIntoAccount ..... `???`
- **thresholdCalculationMethod**
  - realisationType ..... `LambEtAlCriticalTheta`
  - ...
- thetaCriticalBasedOnConstantSred ..... `false`

Figure 19: RickenmannBedloadCapacityBasedOnThetaNonFractional including default values.

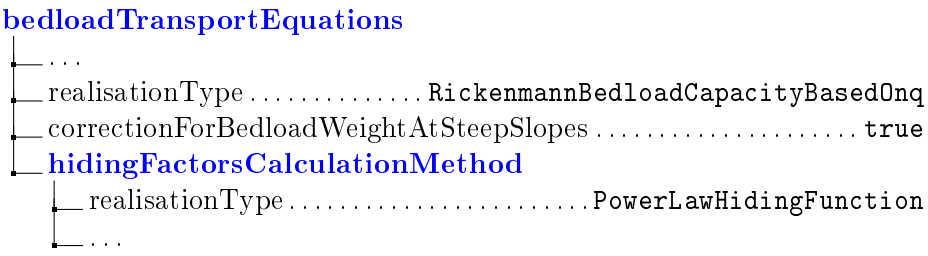

Figure 20: RickenmannBedloadCapacityBasedOnq including default values.

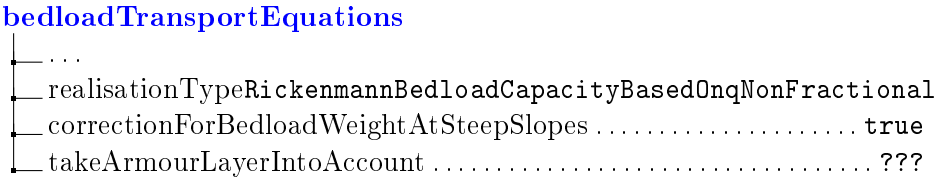

Figure 21: RickenmannBedloadCapacityBasedOnqNonFractional including default values.

13.3.2 *WilcockCroweBedloadCapacity*

The *WilcockCroweBedloadCapacity* (Fig. 22) calculates bedload transport capacity according to Wilcock and Crowe [16]. Its main optional node is *constantSandFraction*, the value of which should range between 0.0 and 1.0. If this node is given, the code will use its value for the determination of the reference dimensionless Shields stress τ^*_{rm} (see equation below). If not, the code will determine the sand fraction F_s from the local grain size distribution. However, it is recommended to have only grain size fractions larger than 2 mm, as they will be transported as bed load. For such a grain size distribution the sand fraction is always 0. Therefore it is recommended to select a *constantSandFraction* between 0.0 and 0.2.

$$\tau^*_{rm} = 0.021 + [0.015 \cdot \exp(-20F_s)] \tag{16}$$

The second optional node *useConstantSandFraction* can be used to force the determination of the sand fraction based on the constant value or based on the grain size distribution. Its default value is *true*, if the node *constantSandFraction* is given, and *false*, if not.

bedloadTransportEquations

- ...
- realisationType ..... `WilcockCroweBedloadCapacity`
- constantSandFraction ..... `???`
- useConstantSandFraction ..... `if constantSandFraction is given`

Figure 22: WilcockCroweBedloadCapacity including default values.

13.3.3 *ReckingBedloadCapacityNonFractional*

The *ReckingBedloadCapacityNonFractional* (Fig. 23) has no specific nodes. It calculates the non fractional transport rates (local grain size distributions constant within simulation) according to the following equations from Recking [11]:

$$\theta_{c84} = (1.32 \cdot S_b + 0.037) \cdot \left(\frac{D_{84}}{D_{50}}\right)^{-0.93} \tag{17a}$$

$$L = 12.53 \cdot \left(\frac{D_{84}}{D_{50}}\right)^{4.445 \cdot \sqrt{S_b}} \cdot \theta_{c84}^{1.605} \tag{17b}$$

$$\theta_{84} = \frac{r_h \cdot S_b}{\left(\frac{\rho_s}{\rho} - 1\right) \cdot D_{84}} \tag{17c}$$

$$\Phi_b = 0.0005 \cdot \left(\frac{D_{84}}{D_{50}}\right)^{-18 \cdot \sqrt{S_b}} \cdot \left(\frac{\theta_{84}}{\theta_{c84}}\right)^{6.5} \quad \text{for} \quad \theta_{84} < L \tag{17d}$$

$$\Phi_b = 14 \cdot \theta_{84}^{2.45} \quad \text{for} \quad \theta_{84} \geq L \tag{17e}$$

In this, θ_{c84} is θ_c for D_{84}; θ_{84} is θ for D_{84} and L is a process break point.

bedloadTransportEquations

- ...
- realisationType ..... `ReckingBedloadCapacityNonFractional`

Figure 23: ReckingBedloadCapacityNonFractional.

13.4 *bedloadVelocityCalculationMethod* realisations

13.4.1 *VelocityAsTranpsportRatePerUnitCrossSectionalArea*

The *VelocityAsTranpsportRatePerUnitCrossSectionalArea* (Fig. 24) estimates bedload velocity v_b according to the following simple relation, in which h_{q_b} is the thickness of the moving sediment layer:

$$v_b = \frac{q_b}{h_{q_b}}, \tag{18}$$

The method for the estimation of the thickness of the moving sediment layer is defined by the optional node *estimateThicknessOfMovingSediment-Layer* with its default realisation *ConstantThicknessOfMovingSedimentLayer* (section 13.10.1).

bedloadVelocityCalculationMethod
- realisationType `VelocityAsTranpsportRatePerUnitCrossSectionalArea`
- **estimateThicknessOfMovingSedimentLayer**
  - realisationType . . . . . `ConstantThicknessOfMovingSedimentLayer`
  - . . .

Figure 24: VelocityAsTranpsportRatePerUnitCrossSectionalArea including default values.

13.4.2 *JulienBounvilayRollingParticlesVelocity*

The *JulienBounvilayRollingParticlesVelocity* (Fig. 25) estimates bedload velocity according to the following equations from Julien and Bounvilay [6]:

$$\theta_r = \frac{\tau}{(\rho_s - \rho) \cdot g \cdot a \cdot D_x}; \tag{19a}$$

$$v_b \approx \sqrt{g \cdot r_h \cdot S} \cdot (3.3 \cdot \ln \theta_r + 17.7)\,; \tag{19b}$$

In this θ_r is θ for the roughness length; τ is the bed shear stress; r_h is the hydraulic radius; D_x is the representative roughness diameter percentile with x defined by the node *roughnessDiameterPercentile* with its default value 84.0 and a is an empiric factor defined by the node *roughnessDiameterFactor* with its default value 3.5.

bedloadVelocityCalculationMethod
- realisationType . . . . . `JulienBounvilayRollingParticlesVelocity`
- roughnessDiameterPercentile . . . . . `84.0`
- roughnessDiameterFactor . . . . . `3.5`

Figure 25: JulienBounvilayRollingParticlesVelocity including default values.

13.5 *strataSorting* realisations

13.5.1 *StratigraphyWithThresholdBasedUpdate*

The *StratigraphyWithThresholdBasedUpdate* (Fig. 26) represents the river alluvium by a pile of sediment layers. The number of layers is defined by the node *numberOfLayers*. The thickness of the topmost and lowermost layer may vary. All other layers have the same thickness which is defined by the

incrementLayerThickness. Whenever the variable thickness of the topmost or active layer exceeds an upper threshold or falls below a lower threshold, sediment is sorted down- or upward until the thickness of the active layer again has a value in the middle of the thresholds. In the simplest case, these thresholds have constant values which are defined by the nodes *valueOrFactorForLowerThresholdForActiveLayerThickness* and *valueOrFactorForUpperThresholdForActiveLayerThickness*. If the switch *useInitialGrainSizesForConstantLayerThickness* is true, the thresholds are defined as a multiple of a reference percentile grain diameter D_x of the initial local grain size distribution. If the switch *dynamicThresholds* is true, the thresholds do not stay constant in the course of a simulation, but are updated according to the changing local grain size distribution. If the thresholds are defined as a function of the grain size distribution, the node *referenceGrainSizePercentile* defines the value of x of the reference percentile diameter D_x and the nodes *valueOrFactorForLowerThresholdForActiveLayerThickness* and *valueOrFactorForUpperThresholdForActiveLayerThickness* are not used as absolut values but as factors, which are multiplied with D_x in order to calculate the threshold values. When D_x becomes small, the thresholds become small as well. However, very thin active layer thicknesses can slow down the simulations. To prevent this, the node *minimumLowerThresholdForActiveLayerThickness* defines a minimum for the lower threshold[1]. When the thresholds become small, the difference between them will become small as well. If this difference becomes smaller than the *incrementLayerThickness* it is not possible for the code to produce an active thickness which has a value between the thresholds. Therefore the node *minimumDifferenceBetweenThresholdsForActiveLayerThickness* with its default value of 1.1 times the *incrementLayerThickness* defines a minimum for the difference between the thresholds.

[1] default: $\left[\frac{\Delta_{min} \cdot low}{up - low}\right]$ with
Δ_{min}=*minimumDifferenceBetweenThresholdsForActiveLayerThickness*,
low=*valueOrFactorForLowerThresholdForActiveLayerThickness* and
up=*valueOrFactorForUpperThresholdForActiveLayerThickness*

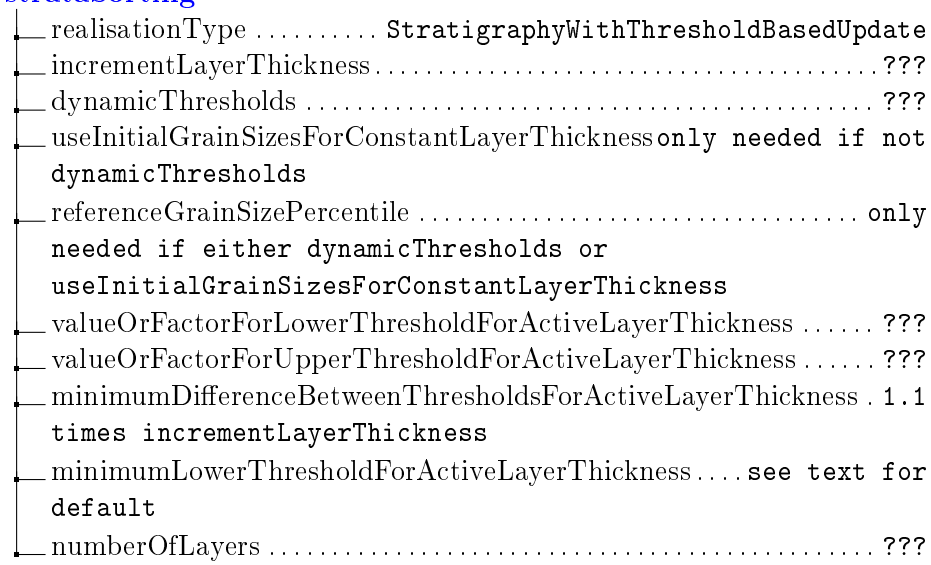

Figure 26: StratigraphyWithThresholdBasedUpdate including default values.

13.5.2 *TwoLayerWithContinuousUpdate*

Within *TwoLayerWithContinuousUpdate* (Fig. 27), the thickness of the active surface layer stays constant for the complete duration of the simulation or is dynamically adjusted according to the current local grain size distribution. If it is constant in time, either a single value can be selected for all reaches or the value for each reach is calculated according to the local grain size distribution at the beginning of the simulation.

The mandatory node *dynamicLayerThickness* selects whether the active layer thickness should be constant in time or dynamically updated. If the value of *dynamicLayerThickness* is *false*, the node *useInitialGrainSizesForConstantLayerThickness* is needed to select whether the constant thickness should be the same for all reaches or defined according to the initial local grain size distribution. If both the *dynamicLayerThickness* and the *useInitialGrainSizesForConstantLayerThickness* are *false*, the active layer thickness is defined by the node *layerThickness*. If only one of the two nodes is *true*, the local active layer thickness is defined as $(b \cdot D_x)$ with b defined by the node *layerThicknessFactor* and x defined by the node *referenceGrainSizePercentile*.

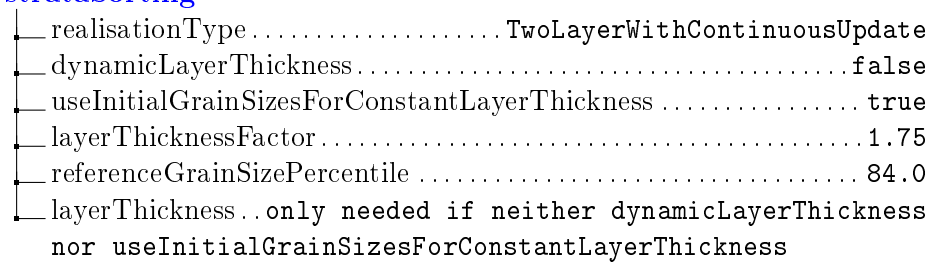

Figure 27: TwoLayerWithContinuousUpdate including default values.

13.5.3 *TwoLayerWithShearStressBasedUpdate*

The *TwoLayerWithShearStressBasedUpdate* (Fig. 28) basically works in the same way as *TwoLayerWithContinuousUpdate*. Therefore, *TwoLayerWithShearStressBasedUpdate* contains all nodes of *TwoLayerWithContinuousUpdate*. However in case of erosion, only some fraction i_s of the material added to the active layer from below, carries grain size information from the subsurface.

$$i_s = \frac{\theta - \theta_{cs}}{\theta_{ca} - \theta_{cs}} \quad \text{with} \quad 0 \leq i_s \leq 1 \tag{20}$$

and with θ_{ca} calculated according to the following relation from Jäggi [5]

$$\theta_{ca} = \theta_{cs} \cdot \left(\frac{D_{mAritha}}{D_{mAriths}} \right)^{\frac{2}{3}} \tag{21}$$

In these equations, θ_{cs} and θ_{ca} are representative θ_c values for the subsurface alluvium or the active layer respectively. Analogously, $D_{mAriths}$ and $D_{mAritha}$ are D_{mArith} values for the subsurface alluvium or the active layer respectively.

To determine i_s, the *TwoLayerWithShearStressBasedUpdate* has the following specific nodes. The method for the calculation of θ_{cs} is defined by the node *thresholdCalculationMethod* which is by default the same method as used in the *bedloadTransportEquations* (section 13.3). Usually the break up conditions including the θ_{cs} and θ_{ca} are defined using the initial local conditions and then kept constant for the complete simulations. To constantly update the break up condition according to the changing local situation, one simply sets the switch *dynamicBreakUpConditions* to *true*. If some global constant break up condition shall be used for all reaches independent of the local conditions, one simply sets the switch *usePredefinedBreakUpConditions* to *true*. If *usePredefinedBreakUpConditions* is *true*, the node *referenceMedianDiameter* defines the representative diameter for the calculation of θ, *thetaCriticalForActiveLayer* defines the value of θ_{ca} and *thetaCriticalForSublayer* defines the value of θ_{cs}.

strataSorting
- realisationType ..... `TwoLayerWithShearStressBasedUpdate`
- dynamicLayerThickness ..... `false`
- useInitialGrainSizesForConstantLayerThickness ..... `true`
- layerThicknessFactor ..... `1.75`
- referenceGrainSizePercentile ..... `84.0`
- layerThickness .. `only needed if neither dynamicLayerThickness nor useInitialGrainSizesForConstantLayerThickness`
- dynamicBreakUpConditions ..... `false`
- usePredefinedBreakUpConditions ..... `false`
- referenceMedianDiameter ..... `only needed if usePredefinedBreakUpConditions`
- thetaCriticalForActiveLayer ..... `only needed if usePredefinedBreakUpConditions`
- thetaCriticalForSublayer ..... `only needed if usePredefinedBreakUpConditions`
- **thresholdCalculationMethod** ..... `method from bedloadTransportEquations or default displayed there`

Figure 28: TwoLayerWithShearStressBasedUpdate including default values.

13.5.4 *SingleLayerNoSorting*

The only mandatory node of *SingleLayerNoSorting* (Fig. 29) is *layerThickness*. Of course, in this realisation, the real layer thickness corresponds to the current thickness of the alluvium. The value of the node *layerThickness* determines the alluvium thickness, below which the bedrock starts to influence flow resistance and hiding processes. The realisation *SingleLayerNoSorting* is intended for non-fractional *bedloadTransportEquations* (section 13.3).

strataSorting
- realisationType ..... `SingleLayerNoSorting`
- layerThickness ..... `0.4`

Figure 29: SingleLayerNoSorting including default values.

13.6 Gradient calculation method realisations

In section 8 the realisation types *NoFlowResistancePartitioning* and *WithFlowResistancePartitioning* were introduced. These pseudo realisation types are short hands for realisations, which are consistent with the current *waterFlowRouting* (section 13.2). If a kinematic wave flow routing is selected, the *NoFlowResistancePartitioning* will expand to *ReturnBedslope* (section 13.6.4)

and the *WithFlowResistancePartitioning* will expand to *ReducedWaterEnergyslope* (section 13.6.3). If *UniformDischarge* (section 13.2.1) is selected as *waterFlowRouting*, the *NoFlowResistancePartitioning* will expand to *SimpleDownstreamTwoCellGradient* (section 13.6.1) with *hydraulicHead* as *propertyOfInterest* and the *WithFlowResistancePartitioning* will expand to *ReducedWaterEnergyslopeNotUsingWaterEnergyslopeVariable* (section 13.6.3) with *SimpleDownstreamTwoCellGradient* as *simpleWaterEnergyslopeCalculationMethod* and with *hydraulicHead* as *propertyOfInterest*. The output of the simulation setup (section 12.3.3) shows in which way the short hand has been expanded.

13.6.1 *SimpleDownstreamTwoCellGradient*

The *simpleDownstreamTwoCellGradient* (Fig. 30) is the recommended standard gradient calculation method within *sedFlow*. It determines the local gradient as the difference between the local and downstream property value divided by the local reach length. The only mandatory child node *propertyOfInterest* defines for which property the gradient is calculated. Potential values are the same as for *regularRiverReachPropertiesForOutput* described in section 12.3.1.

bed- or waterEnergy- or sedimentEnergySlopeCalculationMethod

- realisationType ..... `SimpleDownstreamTwoCellGradient`
- propertyOfInterest ..... `???`

Figure 30: SimpleDownstreamTwoCellGradient.

13.6.2 *SimpleThreeCellGradient*

The *simpleThreeCellGradient* (Fig. 31) determines a local gradient as the difference between the upstream and downstream property value divided by the sum of the local and upstream reach length. The *propertyOfInterest* defines for which property the gradient is calculated. As there may be several upstream reaches, the local gradient is calculated as the weighted average of the individual gradients. The child node *weightingProperty* defines which property is used to weight the individual gradients. Potential values for *propertyOfInterest* and *weightingProperty* are the same as for *regularRiverReachPropertiesForOutput* described in section 12.3.1. The use of simpleThreeCellGradient is not recommended.

bed- or waterEnergy- or sedimentEnergySlopeCalculation-Method

- realisationType ..... `SimpleThreeCellGradient`
- propertyOfInterest ..... `???`
- weightingProperty ..... `???`

Figure 31: SimpleThreeCellGradient.

13.6.3 *ReducedWaterEnergyslope*

The *ReducedWaterEnergyslope* (Fig. 32) corrects the water energy slope according to the following equations from Rickenmann and Recking [14] in order to use it as energy slope for sediment transport calculations considering the shear stress partitioning.

$$v_{tot}\left(r_h\right) = \sqrt{g \cdot r_h \cdot S} \cdot \frac{6.5 \cdot 2.5 \cdot \frac{r_h}{D_{84}}}{\sqrt{6.5^2 + 2.5^2 \cdot \left(\frac{r_h}{D_{84}}\right)^{\frac{5}{3}}}} \tag{22a}$$

$$v_0\left(r_h\right) = \sqrt{g \cdot r_h \cdot S} \cdot 6.5 \cdot \left(\frac{r_h}{D_{84}}\right)^{\frac{1}{6}} \tag{22b}$$

$$\sqrt{\frac{f_0}{f_{tot}}} = \frac{v_{tot}\left(r_h\right)}{v_0\left(r_h\right)} \tag{22c}$$

$$S_{red} = S \cdot \left(\frac{v_{tot}\left(r_h\right)}{v_0\left(r_h\right)}\right)^e \tag{22d}$$

In this, f_0 is the base-level flow resistance and f_{tot} is the total flow resistance. v_0 and v_{tot} are virtual velocities corresponding to f_0 or f_{tot} respectively and S_{red} is the energy slope corrected for the effects of shear stress partitioning.

The value of the empiric exponent e is defined by the node *stressPartitioningExponent* with its default value of 1.5.

The code will make sure that v_{tot} will not result in a Froude number, which exceeds the threshold defined by the node *maximumFroudeNumber*. The default for this threshold is the value which is used for the *flowResistance* (section 13.1).

If the child node *minimumInputSlope* is given, the code will use this value as unreduced slope for the calculation, if the original unreduced slope S is smaller than this value.

The switch *calculationBasedOnqInsteadOfh* may be used to correct the water energy slope according to the following equations, if a *BasedOnq* version of the Rickenmann transport equation is used, in which q^{**} is the di-

mensionless unit discharge. The default value of this node is *false*.

$$q^{**} = \frac{q}{\sqrt{g \cdot S \cdot D_{84}^3}} \tag{23a}$$

$$v_{tot}(q) = \sqrt{g \cdot S \cdot D_{84}} \cdot 1.443 \cdot q^{**0.60} \cdot \left[1 + \left(\frac{q^{**}}{43.78}\right)^{0.8214}\right]^{-0.2435} \tag{23b}$$

$$v_0(q) = \sqrt{g \cdot S \cdot D_{84}} \cdot 3.074 \cdot q^{**0.4} \tag{23c}$$

$$\sqrt{\frac{f_0}{f_{tot}}} = \left(\frac{v_{tot}(q)}{v_0(q)}\right)^{1.5} \tag{23d}$$

$$S_{red} = S \cdot \left(\frac{v_{tot}(q)}{v_0(q)}\right)^{1.5 \cdot e} \tag{23e}$$

The companion realisation *ReducedWaterEnergyslopeNotUsingWaterEnergyslopeVariable* (Fig. 33) can be used in the same way as *ReducedWaterEnergyslope*. However, it does not use the water energy slope as unreduced slope for the calculation, but offers the node *simpleWaterEnergyslopeCalculationMethod* to define a new gradient calculation method to create the unreduced slope for the calculation.

bed- or waterEnergy- or sedimentEnergySlopeCalculationMethod

- realisationType ..... `ReducedWaterEnergyslope`
- stressPartitioningExponent ..... `1.5`
- calculationBasedOnqInsteadOfh ..... `false`
- maximumFroudeNumber ..... `value from flowResistance`
- minimumInputSlope ..... `???`
- ensureMinimumInputSlope .... `if minimumInputSlope node exists`

Figure 32: ReducedWaterEnergyslope including default values.

bed- or waterEnergy- or sedimentEnergySlopeCalculation-Method

- realisationType `ReducedWaterEnergyslopeNotUsingWaterEnergyslopeVariable`
- stressPartitioningExponent `1.5`
- calculationBasedOnqInsteadOfh `false`
- maximumFroudeNumber `value from flowResistance`
- minimumInputSlope `???`
- ensureMinimumInputSlope `if minimumInputSlope node exists`
- **simpleWaterEnergyslopeCalculationMethod**
  - realisationType `SimpleDownstreamTwoCellGradient`
  - propertyOfInterest `hydraulicHead`

Figure 33: ReducedWaterEnergyslopeNotUsingWaterEnergyslopeVariable including default values.

13.6.4 *ReturnBedslope* and *ReturnWaterEnergyslope*

The realisations *ReturnBedslope* (Fig. 34) and *ReturnWaterEnergyslope* (Fig. 35) do not calculate a new gradient, but use the value of the bedslope or water energy slope respectively. If the child node *minimumSlope* is given, the code will use this value, whenever the original bedslope or water energyslope are smaller than this value.

bed- or waterEnergy- or sedimentEnergySlopeCalculation-Method

- realisationType `ReturnBedslope`
- minimumSlope `???`

Figure 34: ReturnBedslope.

bed- or waterEnergy- or sedimentEnergySlopeCalculation-Method

- realisationType `ReturnWaterEnergyslope`
- minimumSlope `???`

Figure 35: ReturnWaterEnergyslope.

13.7 *tauCalculationMethod* realisations

The recommended realisation *EnergyslopeTau* (Fig. 36) calculates bed shear stress τ according to the following equation.

$$\tau = \rho \cdot g \cdot r_h \cdot S \tag{24}$$

The switch *correctionForBedloadWeightAtSteepSlopes* selects whether the following correction of energy slopes according to Rickenmann [13] should be applied (default) or not.

$$S_k = S \cdot \frac{\sin(\phi_r)}{\sin(\phi_r - S_b)} \tag{25}$$

tauCalculationMethod
- realisationType ..... `EnergyslopeTau`
- correctionForBedloadWeightAtSteepSlopes ..... `true`

Figure 36: EnergyslopeTau including default values.

The companion realisation *EnergyslopeTauBasedOnFlowDepth* (Fig. 37) uses flow depth instead of the hydraulic radius r_h for the determination of τ. It can be used in the same way as *EnergyslopeTau*.

tauCalculationMethod
- realisationType ..... `EnergyslopeTauBasedOnFlowDepth`
- correctionForBedloadWeightAtSteepSlopes ..... `true`

Figure 37: EnergyslopeTauBasedOnFlowDepth including default values.

13.8 *thresholdCalculationMethod* realisations

For all *thresholdCalculationMethod* realisations (Fig. 38), the switch *correctionForBedloadWeightAtSteepCounterSlopes* selects whether the following correction of θ_c according to Rickenmann [13] should be applied (default) or not.

$$\theta_{ck} = \theta_c \cdot \frac{\sin(\phi_r + |S_b|)}{\sin(\phi_r)} \tag{26}$$

In this, θ_{ck} is the θ_c corrected for the effects of bedload weight at steep adverse bed slopes.

thresholdCalculationMethod
- correctionForBedloadWeightAtSteepCounterSlopes ..... `true`
- ...

Figure 38: thresholdCalculationMethod including default values.

13.8.1 *ConstantThresholdForInitiationOfBedloadMotion*

The *ConstantThresholdForInitiationOfBedloadMotion* (Fig. 39) has one mandatory child node *constantThreshold*, which defines the threshold which should be used in this simulation.

thresholdCalculationMethod
- ...
- realisationType `ConstantThresholdForInitiationOfBedloadMotion`
- constantThreshold ..... `???`

Figure 39: ConstantThresholdForInitiationOfBedloadMotion.

13.8.2 *LambEtAlCriticalTheta*

The *LambEtAlCriticalTheta* (Fig. 40) calculates the threshold for the initiation of bedload motion according to the following empiric relation from Lamb et al. [7]:

$$\theta_c = 0.15 \cdot S_b^{0.25} \tag{27}$$

In this equation, gentle slopes may result in too low values of θ_c. Therefore, the optional child node *minimumCriticalDimensionlessShearStress* whith its default value of 0.03 defines a minimum value for θ_c.

thresholdCalculationMethod
- ...
- realisationType ..... `LambEtAlCriticalTheta`
- minimumCriticalDimensionlessShearStress ..... `0.03`

Figure 40: LambEtAlCriticalTheta including default values.

13.9 *hidingFactorsCalculationMethod* realisations

The *hidingFactorsCalculationMethod* modifies the threshold for the initiation of bedload motion θ_c for the different grain size fractions to account for the effects of grain exposure and hiding.

13.9.1 *PowerLawHidingFunction*

The *PowerLawHidingFunction* (Fig. 41) modifies θ_c according to the following equation (e.g. Parker [10]).

$$\theta_{ci} = \theta_c \cdot \left(\frac{D_i}{D_x}\right)^m \tag{28}$$

In this θ_{ci} is the θ_c for the grain size fraction i; D_i is the representative grain diameter of this fraction; D_x is a reference grain diameter percentile with

x defined by the optional child node *referenceDiameterPercentile* with its default value of 50.0 and m an empiric hiding exponent, which is defined by the optional child node *exponent* with its default value of -0.8.

hidingFactorsCalculationMethod
- realisationType ..... `PowerLawHidingFunction`
- referenceDiameterPercentile ..... `50.0`
- exponent ..... `-0.8`

Figure 41: PowerLawHidingFunction including default values.

13.9.2 *NoHiding*

The *NoHiding* (Fig. 42) does not modify θ_c at all. In this way, it corresponds to a *PowerLawHidingFunction* with an *exponent* of 0.0.

hidingFactorsCalculationMethod
- realisationType ..... `NoHiding`

Figure 42: NoHiding.

13.9.3 *WilcockCroweHidingFunction*

The *WilcockCroweHidingFunction* (Fig. 43) modifies θ_c according to the following equation from Wilcock and Crowe [16].

$$\theta_{ci} = \theta_c \cdot \left(\frac{D_i}{D_m}\right)^{m_{wc}} \quad \text{with} \quad m_{wc} = \frac{0.67}{1 + \exp\left(1.5 - \frac{D_i}{D_m}\right)} - 1 \tag{29}$$

In this D_m is the geometric mean diameter of the local grain size distribution and m_{wc} the hiding exponent according to Wilcock and Crowe [16].

hidingFactorsCalculationMethod
- realisationType ..... `WilcockCroweHidingFunction`

Figure 43: WilcockCroweHidingFunction.

13.10 *estimateThicknessOfMovingSedimentLayer* realisations

13.10.1 *ConstantThicknessOfMovingSedimentLayer*

The *ConstantThicknessOfMovingSedimentLayer* (Fig. 44) assumes the thickness of the moving sediment layer to be equal to the value in metres defined in the child node *constantThickness*, with its default value of 0.7. This

rather large default, has been chosen to avoid unnecessary slowing down of simulations.

estimateThicknessOfMovingSedimentLayer
- realisationType ..... `ConstantThicknessOfMovingSedimentLayer`
- constantThickness ..... `0.7`

Figure 44: ConstantThicknessOfMovingSedimentLayer including default values.

13.10.2 *MultipleDiameterOfCoarsestGrainMoved*

The *MultipleDiameterOfCoarsestGrainMoved* (Fig. 45) takes the diameter of the coarsest fraction, for which the local transport rate exceeds the threshold defined by the node *minimumTransportRateForFractionToBeConsidered-Moving* with its default value of 0.0001, and multiplies it with the value defined by the node *factor* with its default value of 1.75 in order to estimate the thickness of the moving sediment layer.

estimateThicknessOfMovingSedimentLayer
- realisationType ..... `MultipleDiameterOfCoarsestGrainMoved`
- minimumTransportRateForFractionToBeConsideredMoving .. `0.0001`
- factor ..... `1.75`

Figure 45: MultipleDiameterOfCoarsestGrainMoved including default values.

13.10.3 *MultipleReferenceGrainDiameter*

The *MultipleReferenceGrainDiameter* (Fig. 46) multiplies some reference diameter percentile D_x of the local grain size distribution with the value defined by the node *factor* with its default value of 1.25 in order to estimate the thickness of the moving sediment layer. The value of x is defined by the node *referenceDiameterPercentile* with its default value of 84.0. In cases with no alluvium cover or input from upstream it is not possible to calculate any diameter percentile. In these cases, the thickness of the moving sediment layer is assumed to be equal to the value in metres defined in the child node *default-ThicknessForCasesWithNoAlluviumOrInputFromUpstream*, with its default value of 0.7.

estimateThicknessOfMovingSedimentLayer
- realisationType ..... `MultipleReferenceGrainDiameter`
- factor ..... `1.25`
- referenceDiameterPercentile ..... `84.0`
- defaultThicknessForCasesWithNoAlluviumOrInputFromUpstream `0.7`

Figure 46: MultipleReferenceGrainDiameter including default values.

13.11 *additionalMethods* with Sternberg abrasion

To this date the only *additionalMethods* are *SternbergAbrasionWithoutFining* (Fig. 47) and *SternbergAbrasionIncludingFining* (Fig. 48). These methods correct the estimated bedload flux for gravel abrasion according to the classic equation of Sternberg [15], in which $q_{b_{abr}}$ is bedload flux per unit flow width corrected for abrasion; λ is an empiric abrasion coefficient and ΔX is the travel distance of the grains *given in kilometres*. In this, the material loss due to erosion is regarded as suspension throughput load.

$$q_{b_{abr}} = q_b \cdot \exp\left(-\lambda \cdot \Delta X\right) \tag{30}$$

The value of *lambda* is defined by the mandatory child node *sternbergAbrasionCoefficient*. The method *SternbergAbrasionWithoutFining* simply reduces the transported volumes according to equation 30. The method *SternbergAbrasionIncludingFining* reduces the transported volumes and shifts material from coarser to finer grain size fractions (assuming equally distributed grain sizes within each fraction). If both nodes are given only the *SternbergAbrasionIncludingFining* is used.

additionalMethods
- **SternbergAbrasionWithoutFining**
  - sternbergAbrasionCoefficient ..... `???`

Figure 47: SternbergAbrasionWithoutFining.

additionalMethods
- **SternbergAbrasionIncludingFining**
  - sternbergAbrasionCoefficient ..... `???`

Figure 48: SternbergAbrasionIncludingFining.

Part III
Appendix

Notation

The following symbols are used in this manual:

β = an empiric constant factor

γ = correction factor for θ_c

Δt = temporal discretisation
i.e. time step duration

Δx = spatial discretisation
i.e. reach length

ΔX = travel distance of grains

θ = dimensionless bed shear stress

θ_{84} = θ for D_{84}

θ_c = dimensionless bed shear stress threshold for initiation of bedload motion

θ_{ca} = a representative θ_c for the complete active layer

θ_{ci} = θ_c for i'th grain size fraction

θ_{ck} = θ_c corrected for bedload weight at steep adverse bed slopes

$\theta_{c,r}$ = θ_c corrected for form roughness

θ_{cs} = a representative θ_c for the complete subsurface alluvium

θ_{c84} = θ_c for D_{84}

θ_r = θ for the roughness length according to Julien and Bounvilay [6]

λ = empiric abrasion coefficient

μ = weir coefficient

ρ = fluid density

ρ_s = sediment density

τ = bed shear stress

τ^*_{rm} = reference dimensionless Shields stress for mean size of bed surface

ϕ_r = angle of repose of bedload material

Φ_b = dimensionless bedload flux

a,b = empiric constants

D = grain diameter

D_i = representative grain diameter for i'th grain size fraction

D_m = geometric mean for grain diameters

D_{mArith} = arithmetic mean for grain diameters

D_{mArith_a} = D_{mArith} for the active layer

D_{mArith_s} = D_{mArith} for the subsurface alluvium

D_x = x'th percentile for grain diameters

D_{xs} = x'th percentile for grain diameters of bed surface

D_{50} = median grain diameter

e = empiric exponent
ranging from 1 to 2

f = Darcy-Weisbach friction factor

f_0 = base-level flow resistance

f_{tot} = total flow resistance

F_s = proportion of sand fraction

Fr = Froude number

g = gravitational acceleration

h = hydraulic head

h_{q_b} = thickness of
moving sediment layer

i_s = grain size influence from the subsurface alluvium

j,k,l = empirical constants

k_{st} = Strickler roughness coefficient

L = process break point

m = empiric hiding exponent

m_{wc} = hiding exponent according to Wilcock and Crowe [16]

n = Manning roughness coefficient

q = discharge per unit flow width

q^{**} = dimensionless unit discharge

q_b = bedload flux per unit flow width

$q_{b_{abr}}$ = q_b corrected for gravel abrasion

q_c = discharge per unit flow width
threshold for initiation of
bedload motion

q_{cA} = q_c corrected for the effect of bed armouring

Q = discharge

Q_{lat} = lateral water influx

r_h = hydraulic radius

$r_{h,c}$ = r_h for $[\theta_{50} = theta_c]$

S = slope

S_c = virtual slope for the correction of θ_c based on $r_{h,c}$

S_b = bed slope

S_f = friction slope

S_k = slope corrected for
bedload weight at steep bed slopes

S_{red} = slope reduced for
form roughness

t = time

U^* = shear velocity

v = flow velocity

v_0 = virtual v corresponding to base-level flow resistance

v_b = bedload velocity

v_{tot} = virtual v corresponding to total flow resistance

V_{pore} = pore volume fraction

w = flow width

x = distance in flow direction

z = elevation of channel bed

List of Tables

List of Figures

References

[1] A. Badoux and D. Rickenmann. Berechnungen zum Geschiebetransport während der Hochwasser 1993 und 2000 im Wallis. *Wasser Energie Luft*, 100(3):217–226, 2008.

[2] R. I. Ferguson. Flow resistance equations for gravel- and boulder-bed streams. *Water Resour. Res.*, 43:W05427, 2007. doi: 10.1029/2006WR005422.

[3] F. U. M. Heimann, J. M. Turowski, D. Rickenmann, and J. W. Kirchner. sedflow - an efficient tool for simulating bedload transport, bed roughness, and longitudinal profile evolution in mountain streams. *Earth Surface Dynamics*, submitted.

[4] F. U. M. Heimann, D. Rickenmann, M. Böckli, A. Badoux, J. M. Turowski, and J. W. Kirchner. Recalculation of bedload transport observations in swiss mountain rivers using the model sedflow. *Earth Surface Dynamics*, submitted.

[5] M. N. R. Jäggi. Sedimenthaushalt und Stabilität von Flussbauten. In D. Vischer, editor, *Mitteilung der Versuchsanstalt für Wasserbau, Hydrologie und Glaziologie*, number 119, pages 1–105. ETH Zurich, Switzerland, 1992.

[6] P. Y. Julien and B. Bounvilay. Velocity of rolling bed load particles. *J. Hydr. Eng.*, 139(2):177–186, 2013. doi: 10.1061/(ASCE)HY.1943-7900.0000657.

[7] M. P. Lamb, W. E. Dietrich, and J. G. Venditti. Is the critical Shields stress for incipient sediment motion dependent on channel-bed slope? *J. Geophys. Res.*, 113:F02008, 2008. doi: 10.1029/2007JF000831.

[8] Z. Liu and E. Todini. Towards a comprehensive physically-based rainfall-runoff model. *Hydrology and Earth System Sciences*, 6(5):859–881, 2002. doi: 10.5194/hess-6-859-2002.

[9] M. Nitsche, D. Rickenmann, J. M. Turowski, A. Badoux, and J. W. Kirchner. Evaluation of bedload transport predictions using flow resistance equations to account for macro-roughness in steep mountain streams. *Water Resour. Res.*, 47:W08513, 2011. doi: 10.1029/2011WR010645.

[10] G. Parker. Transport of gravel and sediment mixtures. In *Sedimentation Engineering: Theories, Measurements, Modeling, and Practice*, volume 110 of *ASCE Manuals and Reports on Engineering Practice*, chapter 3, pages 165–252. American Society of Civil Engineers (ASCE), 2008.

[11] A. Recking. A comparison between flume and field bed load transport data and consequences for surface-based bed load transport prediction. *Water Resour. Res.*, 46:W03518, 2010. doi: 10.1029/2009WR008007.

[12] D. Rickenmann. Comparison of bed load transport in torrent and gravel bed streams. *Water Resour. Res.*, 37(12):3295–3305, 2001. doi: 10.1029/2001WR000319.

[13] D. Rickenmann. Geschiebetransport bei steilen Gefällen. In *VAW 75 JAHRE Festkolloquium 7. Oktober 2005*, pages 107–119. Versuchsanstalt für Wasserbau, Hydrologie und Glaziologie, Zurich, Switzerland, 2005.

[14] D. Rickenmann and A. Recking. Evaluation of flow resistance in gravel-bed rivers through a large field data set. *Water Resour. Res.*, 47: W07538, 2011. doi: 10.1029/2010WR009793.

[15] H. Sternberg. Untersuchungen über längen- und querprofil geschiebeführender flüsse. *Zeitschrift für Bauwesen*, 25:483–506, 1875.

[16] P. R. Wilcock and J. C. Crowe. Surface-based transport model for mixed-size sediment. *J. Hydr. Eng.*, 129(2):120–128, 2003. doi: 10.1061/(ASCE)0733-9429(2003)129:2(120).