RAVEN:
User's and Developer's Manual v2.7

the RAVEN development team

Contributions to Raven, its utilities, documentation, testing, and source code, have been made by numerous students at the University of Waterloo, including Rob Chlumsky, Susan Huang, Ayman Khedr, Konhee Lee, Stuart Pearson, Silvie Spraakman, Graham Stonebridge, Connor Werstuck, and Cloud Zhang. Andrew P. Snowdon contributed much to the library of hydrological process algorithms, and to the global numerical solver. Juliane Mai has provided Mac/Linux compilation support and NetCDF gridded data support. Martin Serrer from NRC contributed to the I/O design, code optimization, and interfacing tools. Wayne Jenkinson from NRC and Georg Jost from BC Hydro have provided debugging, benchmarking, and planning support. The base software architecture and much of the core program was developed by James R. Craig at the University of Waterloo, with support through multiple conversations with Drs. Eric Soulis and Bryan Tolson.

Contents

Chapter 1

Introduction

This document describes the design and operation of the Raven hydrological modelling framework, a software package for watershed modeling. The document is meant for both users of the software who wish to run the program and understand the multitude of model options and by new developers of the Raven software who wish to understand, customize, and/or upgrade the code (chapters and sections for developers are marked with an asterisk*).

Raven is a mixed lumped/semi-distributed model that can be used to understand the hydrological behavior of a watershed and assess the potential impacts of land use, climate, and other environmental change upon watershed properties such as flood potential, soil water availability, or groundwater recharge. The model can be used to investigate individual storm events or develop long-term water, mass, and energy balances for resource management and water quality assessment. Raven's uniqueness primarily comes from its numerical robustness and its flexibility; Raven is able to use a wide variety of algorithms to represent each individual component of the hydrological cycle and has a quite general treatment of every possible model option, from output access to numerical simulation algorithm. Because of its modular design, users have access to a number of different methods of interpolating meteorological forcing data, routing water downstream, representing evaporation, and any number of other model options. With this flexibility, a modeler can examine the wide range of possible outcomes that result from our uncertainty about a watershed model, and test hypotheses about watershed function.

In addition, Raven's flexibility and large library of user-customizable subroutines allow it to emulate (and augment) a number of existing hydrological models. Raven has achieved level 1 (near-perfect) emulation of the UBC Watershed Model (Quick, 1995), Environment Canada's version of the HBV model (Bergstrom, 1995), and GR4J (Perrin et al., 2003). Level 2 (conceptual) emulation is available for various algorithms used which are comparable to those found in (e.g.,) Brook90, SWAT, VIC, PRMS, HYMOD, and/or described within various hydrological texts, such as Dingman's *Physical Hydrology* (2002).

1.1 Model Abstraction

While much of Raven's operations are generic and flexible, they are all built up from critical assumptions about the organization and operation of a watershed. These collectively form the core structure of any Raven model, which is depicted in figure 1.1. A watershed is here assumed to be assembled from a number of subbasins, which in turn are assembled from a number of contiguous or non-contiguous hydrological response units (HRUs), defined as areas with hydrologically unique responses to precipitation events. Each HRU is defined by a single combination of land use/land type (LU/LT), vegetation cover, and terrain type and is underlain by a defined soil profile and stratified aquifer. Membership in these classification schemes, or property classes, is used to determine all or part of the physically-based properties of the response unit, such as soil conductivity or leaf area index. Each HRU is composed of a finite number of storage compartments, such as the soil, canopy, and snowpack, in which water and energy are stored (see table 1.1). Given some set of user-specified controlling hydrological processes (see table 1.2), RAVEN builds and solves the resultant zero- and 1-dimensional water and energy balance problem for each HRU, redistributing water within the HRU in response to precipitation and other atmospheric forcings. Some of this water is redistributed to surface water channels associated with the subbasin, where it is routed downstream from subbasin to subbasin. During this simulation process, diagnostics about water/mass/energy storage distribution, cumulative flux, and instantaneous fluxes may be tracked.

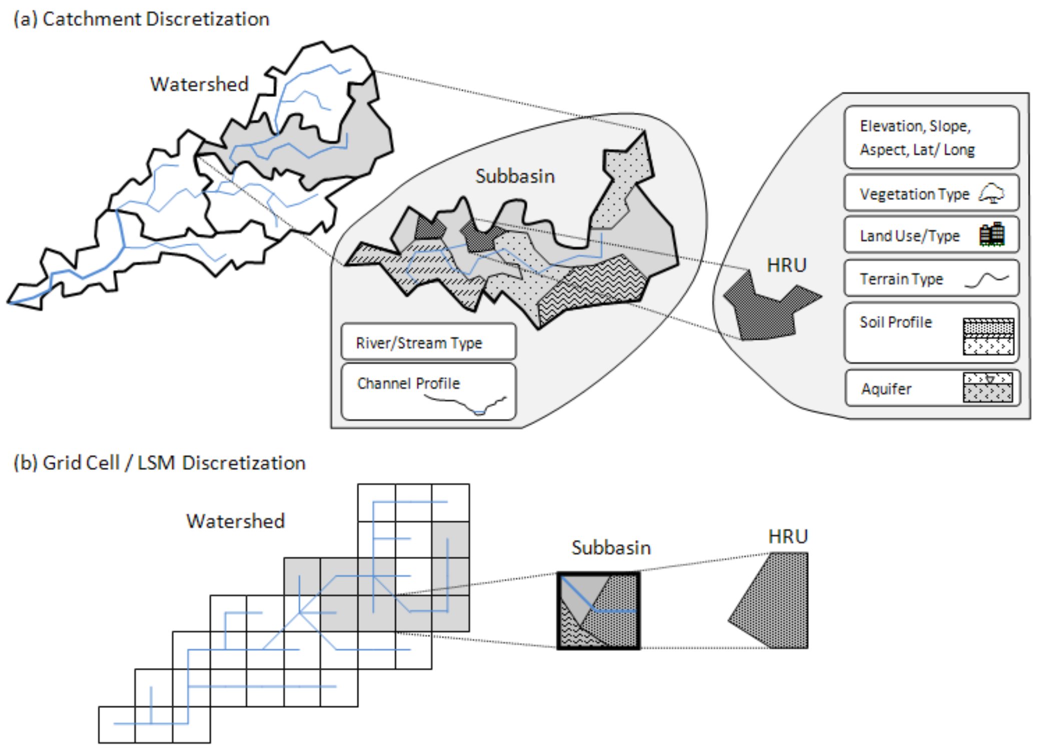


Figure 1.1: Land surface partitioning in RAVEN

surface(ponded water)	surface(lakes and streams)	atmospheric
shallow soil	deep soil	groundwater aquifer
frozen snow	liquid snow	canopy
glacial ice	glacial melt	wetlands

Table 1.1: Common storage compartments that correspond to state variables in hydrological models - each compartment can store both water and energy (a non-comprehensive list)

Each HRU is wholly defined by its geometric properties (area, latitude, longitude, parent subbasin), topographic properties (slope, aspect), subterranean soil profile, and its property class membership (land use, vegetation, terrain). Each soil horizon in the soil profile and the aquifer in turn belong to a soil property class. All individual HRU properties are assigned based upon mem-

precipitation	runoff	evaporation	transpiration
drip	trunk drainage	canopy drainage	interflow
throughfall	infiltration	recharge	capillary rise
snowmelt	sublimation		glacial melt

Table 1.2: Common hydrological processes that may be included in a RAVEN model

bership in these classes, i.e., most of the properties belong to the class, not the HRU, enabling the solution of a finely discretized model (>1000 HRUs) without generating an equally large number of unknown parameters.

As a generalization of standard methods used to represent shallow soils in hydrological models, the shallow subsurface may be represented by one or many discrete layers, which is generated from the specified soil profile, as shown in figure 1.2. The soil profile, specified for each HRU, describes the thickness and soil type of each constituent horizon. Soil parameters for the M-layer soil model (e.g., hydraulic conductivity) are then determined based upon soil class membership of each soil horizon, aggregated or disaggregated depending upon desired vertical model resolution. Alternatively, the soil layers may correspond to conceptual soil moisture stores not explicitly linked to physical soil horizon, as is done in many lumped watershed models.

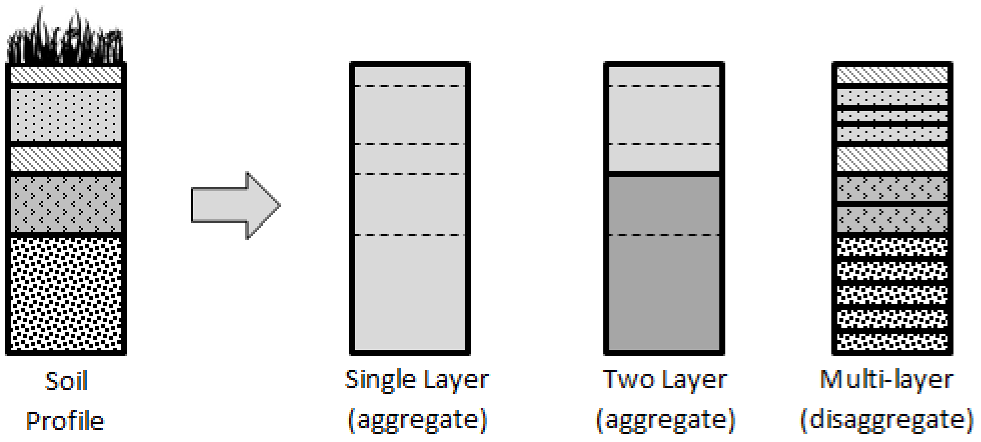


Figure 1.2: Translation of soil profiles to soil models. Properties are aggregated or disaggregated depending upon specified vertical resolution of soil model

Subbasins are similarly succinctly characterized by their channel characteristics, their topology with respect to other subbasins (i.e., their outlet basin) and their cross-sectional profile. Again, properties are linked to channel and profile types, so finely discretized distributed models may still be parsimonious in terms of parameters.

With RAVEN, unlike other models, the modeler determines the degree of model complexity. At the simplest, a watershed can be treated as a single giant HRU/subbasin where only daily precipitation and temperature are needed to provide predictions of streamflow. In the other extreme, the model could be composed of thousands of HRUs consisting of tens of individual storage compartments and forced using measured hourly longwave radiation, wind velocity, and air pressure. The complexity of the model is limited by the user or (even more sensibly) the availability of data.

While the various components of the HRU water balance are user-specified, an example schematic of the flow of water in a single HRU can be seen in figure 1.3.

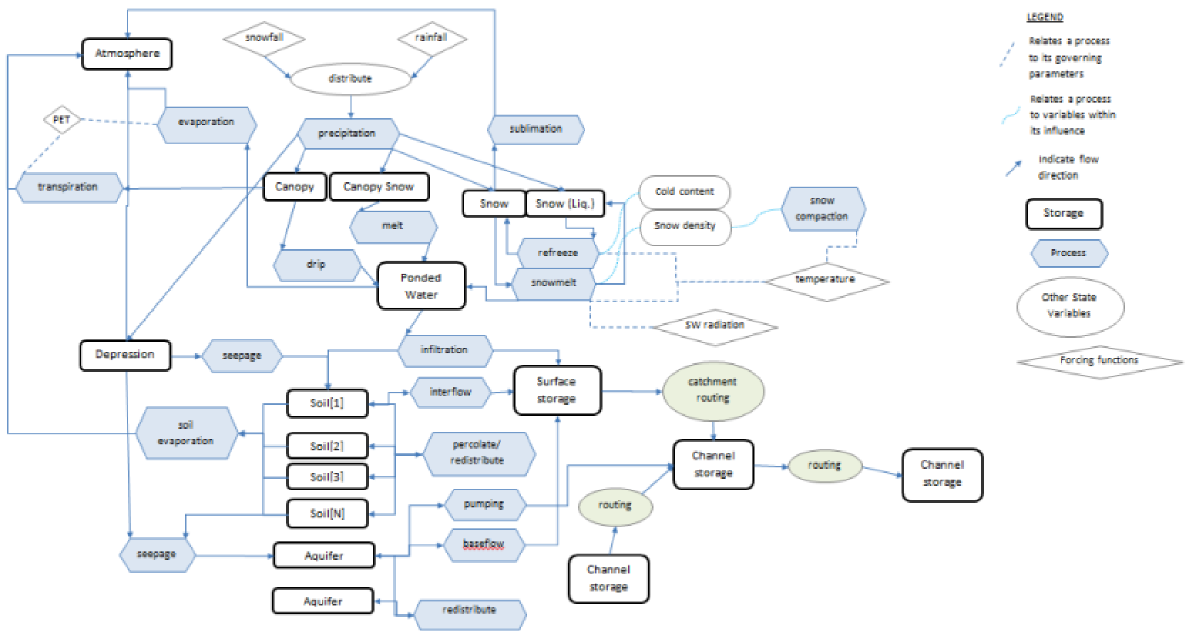


Figure 1.3: Example flowchart of the water balance in a RAVEN model. Note that individual processes and storage compartments may be added or subtracted from this schematic.

1.2 Global Numerical Algorithm

The operation of RAVEN is fundamentally simple. Starting from some initial state of the watershed, the model moves forward in time, updating the distribution of water, mass and energy both within and between HRUs in response to physical forcings such as precipitation, and laterally routing water and energy downstream to the watershed outlet. The entire system is simulated one timestep at a time. During each timestep, the following sequence of events occur:

1. The forcing functions are updated, i.e., the representative values of rain and snow precipitation, temperature, and perhaps wind velocity, longwave radiation, etc. are generated or extracted from user-specified time series at a (relatively small) number of gauge stations, then interpolated to every HRU in the model. Alternatively, these functions may be specified as a gridded model input from a regional climate or weather model.

2. All of the model parameters which change in response to the current state of the system are updated in each HRU (for example, canopy leaf area index may be updated with the seasons)

3. Using these updated forcing functions and parameters, the state of the system at the end of the timestep is determined from the state of the system at the start of the timestep by rigorously solving the coupled mass and energy balance equations in each HRU in the model. These mass and energy balances are assembled from the relevant hydrological processes occurring in the HRU, which individually redistribute water and energy between different compartments (e.g., the evaporation process may move ponded water to the atmosphere).

4. If needed, advective and dispersive mass transport of constitutents (contaminants or tracers) is simulated using the water fluxes over the time step.

5. Runoff from the HRUs (and mass/energy associated with this runoff) is routed into the surface water network in each subbasin, and concurrently routed downstream.

6. Mass/Energy balance checks are performed
7. Output is written to a number of continuous output files

The process is repeated until the model has been run for the specified duration.

1.2.1 The HRU Mass/Energy Balance

The problem being solved by Raven within each HRU is fundamentally that of a coupled system of ordinary and partial differential equations (ODEs and PDEs). These ODEs and PDEs individually describe either (1) the accumulation of mass or energy within a given storage compartment or continuum (i.e., a mass or energy balance) or (2) the temporal change in some auxiliary system property (e.g., snow density or albedo).

Here, each state variable in an HRU is subject to the influence of a number of hydrological processes. Increases or decreases in a primary state variable are simply the additive combination of influx or outflux terms (i.e., the ODE or PDE corresponding to a primary state variable is built up from mass or energy balance considerations). Increases or decreases in auxiliary variables are likewise assumed to be written as the additive combination of terms. We can therefore write an individual differential equation for the change in the j^{th} state variable, ϕ_j, as:

$$\frac{\partial \phi_j}{\partial t} = \sum_{k=1}^{NP}\sum_{i=1}^{NS} M_{ij}^k(\vec{\phi}, \vec{P}, \vec{F}) \tag{1.1}$$

where M_{ij}^k is the change in state variable j due to process k (of NP processes), which is linked to another state variable i. This linkage typically communicates flow direction, e.g., a process M_{ij}^k moves mass or energy from storage compartment i to compartment j. A process M_{ii}^k (i.e., $i = j$) represents an independent rate of change for an auxiliary variable, and does not connotate exchange of mass or energy between compartments. The fluxes or rates-of-change returned by each process are a function of the current vector of state variables ($\vec{\phi}$), system parameters ($\vec{P}$), and forcing functions $\vec{F}$. For example, the mass balance for ponded water on the land surface (depression storage, DS) may be given as:

$$\frac{\partial \phi_{DS}}{\partial t} = P - E - I - R \tag{1.2}$$

where P is the precipitation input, E is the evaporation rate, I is the infiltration rate into the soil beneath, and R is the overflow rate of the depression. Each of these processes (M^k) may be a function of a number of forcings (e.g., precipitation and temperature), current state variables (e.g., ponding depth and soil saturation), and parameters (e.g., maximum depression storage and soil hydraulic conductivity).

The full system of equations describing the influence of all processes in an HRU can be written in matrix form:

$$\frac{\partial \vec{\phi}}{\partial t} = \mathbf{M}^G(\vec{\phi}, \vec{P}, \vec{F})\{1\} \tag{1.3}$$

where $\vec{\phi}$ is the complete vector of state variables, $\mathbf{M}^G$ is a NSxNS global symmetric matrix of composite rate-of-change functions, where NS is the number of state variables, and $\{1\}$ is a column vector filled with ones. The global process matrix is the sum of contributions from each individual symmetric process matrix, i.e., $\mathbf{M}^G = \sum \mathbf{M}^k$.

The above mathematical formulation enables the complete separation of individual hydrological process algorithms, which may individually be very simple or quite complicated. It also enables the use of a variety of methods for solving the global system of equations defined by 1.3. Because of the approach used to solve this system, mass balance errors are typically on the order of machine precision.

1.2.2 Routing

Raven separately handles in-catchment routing (from the HRU to the major reach in the subbasin) and in-channel routing (in and between subbasin stream reaches). The concept is depicted in figure 1.4.

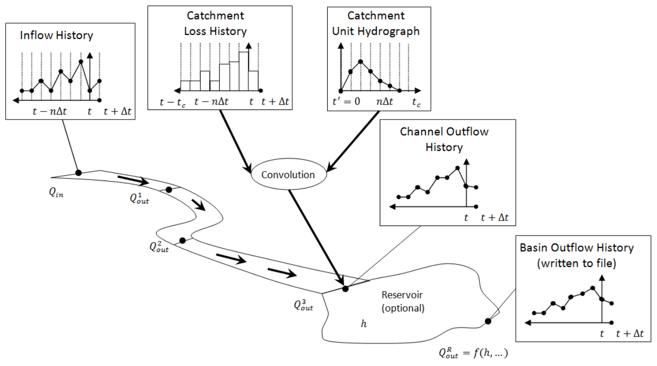


Figure 1.4: The general routing model of Raven

In-catchment routing to the primary basin channel is generally handled using a convolution or unit hydrograph (UH) approach, where the UH for each catchment is either user-specified or generated from basin characteristics. The immediate history of quickflow, interflow, and baseflow output to surface water is stored in memory as an array of time step-averaged outflow rates to off-channel tributaries, $\vec{Q}^{lat}$; the duration of this history is determined by the subbasins time of concentration, t_c. To transfer this water to either the channel segments within the subbasin or directly to the subbasin outflow, the pulse hydrograph is convolved with the unit hydrograph, represented as a piecewise linear function. Water and energy is transferred to the downstream ends of channel segments within the reach.

In-channel routing, for each time step, is assumed to be completely characterized by a finite history of upstream inflow (stored as a vector of flow values at fixed time intervals of Δt, $\vec{Q}^{in}$), and the outflow at the start of the time step; the duration of this history is determined by the minimum flood celerity and the length of the reach segment. During each time step, moving from upstream to downstream at both the watershed level (basin to basin) and subbasin level (reach segment to reach segment), a routing algorithm is used to generate the outflow from each reach based upon the time history of upstream inflows, i.e.,

$$Q_{out}^{n+1} = F_{route}(Q_{out}^{n}, \vec{Q}^{in}, \vec{P}_s) \tag{1.4}$$

where F_{route} is the routing algorithm, $\vec{P}_s$ is a vector of channel parameters, typically a number of stored channel rating curves, primary channel and bank roughness, and, if applicable, weir or reservoir relationships. This formalization supports both common lumped and distributed flow routing methods depending upon the form of $F_{route}()$, including Muskingum-Cunge, lag-and-route, transfer function/unit hydrograph convolution, and, if desired, a more complex kinematic wave or diffusive wave approach (not currently implemented). Notably, sub-time-stepping for routing is also enabled with this formulation.

Reservoir routing. At the outlet of each subbasin, the option exists to specify a reservoir which mediates the outflow from the subbasin channel. This reservoir is characterized using specified volume-stage and surface area-stage relationship, and level-pool outflow from the reservoir may be calculated using a variety of methods, including simple weir formulae to complex reservoir management rules. The mass balance within the reservoir is calculated as

$$\frac{dV(h)}{dt} = Q_{in}(t) - Q_{out}(t,h) - ET(A(h)) + P(A(h)) \tag{1.5}$$

where $V(h)$ is the stage (h) dependent volume of the reservoir, Q_{in} is the inflow to the reservoir, $Q_{out}(t,h)$ is the outflow from the reservoir (a function of stage), and ET and P are the evapotranspiration from and precipitation to the reservoir surface, both functions of surface area.

1.3 Conceptual Model

The critical feature of Raven is that it does not make any assumptions about the functioning of the watershed. That is the modelers job. There is no single system conceptualization that is forced upon the modeler, other than those imposed by the Subbasin-HRU model framework. Rather, the modeler determines what processes to use, how to parameterize the watershed, how to discretize the watershed. All the while, Raven makes this easy to do by providing reasonable defaults, an intuitive file interface, and a large library of hydrologic and algorithmic options.

Chapter 2

Running Raven

Much energy has been expended to ensure that the operation and use of Raven is as simple, convenient, intuitive, and user-friendly as possible. Model commands and file formats are in plain English, error messages are reasonably concise and explanatory, unnecessary restrictions or requirements are not forced on the user, and model input and output files can be read and understood with a minimal learning curve. There may be, however, a learning curve in familiarizing oneself with the large variety of modelling options and how they differ.

2.1 Installation

There is no formal installation package for Raven, and no special programs are libraries are required to operate Raven. Simply download the windows or linux executable `Raven.exe` and unzip to a local drive.

2.2 Input Files

In order to perform a simulation using Raven, the following five input files are required:

- `modelname.rvi` - the primary model input file
  This is where the primary functioning of the Raven model is specified. This includes all of the numerical algorithm options (simulation duration, start time, time step, routing method, etc.) and model structure (primarily, how the soil column is represented). Critically, the list of hydrological processes that redistribute water and energy between storage compartments is specified here, which define both the conceptual model of the system, the specific state variables simulated, and the parameters needed. Lastly, various options for output generation are specified.
- `modelname.rvh` - the HRU / basin definition file
  The file that specifies the number and properties of subbasins and HRUs, as well as the connectivity between subbasins and HRUs. Importantly, land use/land type, vegetation class, aquifer class, and soil classes are specified for each HRU in order to generate appropriate model parameters to represent the properties of each HRU.

- `modelname.rvt` - the time series/forcing function file
  This file specifies the temperature, precipitation, and possibly other environmental forcing functions at a set of observation points ("gauges") within the model domain. This information is interpolated to each HRU within the watershed based upon spatial location. The .rvt file typically "points" to a set of files storing information for each gauge or forcing type. If gridded forcing data is used, the details about the corresponding netCDF gridded data file and connections between the grid and landscape are specified here.

- `modelname.rvp` - the class parameters file
  This is where most of the model parameters are specified, grouped into classes. Each HRU belongs to a single vegetation class, single land use, single aquifer class, and has a unique soil profile defined by a collection of soil horizons each of a single soil class. All model parameters, on a class by class basis, are specified here. The class formalism aids in the calibration process. Note that the `:CreateRVPTemplate` command can be used to generate an empty .rvp file given the model configuration specified in the .rvi file (see appendix A.1.4 for details).

- `modelname.rvc` - the initial conditions file
  This is where the initial conditions for all state variables in all HRUs and subbasins are specified. This may be generated from the output of a previous model run. If a blank file is provided, all storage initial conditions are assumed to be zero (i.e., no snow, dry soil, etc.) and a run-up period will be warranted.

Each of these files are described in detail in appendix A. While the .rvi (setup), .rvh (watershed geometry), .rvc (initial conditions) and .rvt(forcing data) files are typically unique to a particular model, the .rvp (properties) file may ideally be ported from one model to another. Figure 2.1 depicts the base input used by and output generated from Raven, where the default/mandatory files for all simulations are indicated in light blue.

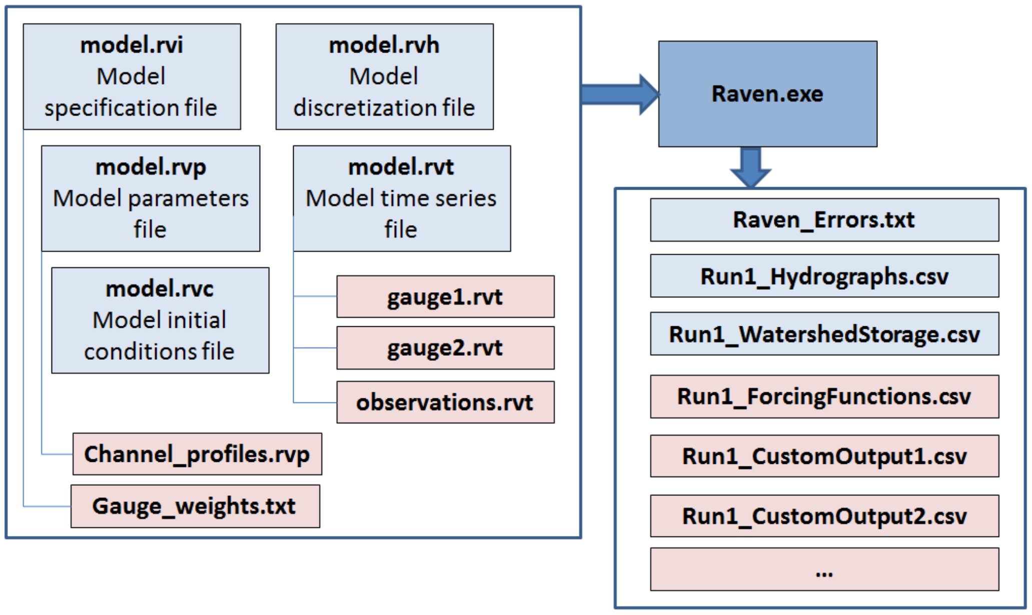


Figure 2.1: Standard input/output configuration of Raven. Light blue input files are required, light blue output files are the default output (which may be suppressed if desirable). The light red input files are files referred to by the primary input files, and are kept separate mostly for organization. The light red output files are generated only if specifically requested by the user in the .rvi file.

To prepare the input files, it is recommended to first familiarize yourself with the format and various input options. A number of pre-processors have been or are being developed to generate the .rvt file(s) from alternative formats. For instance, Environment Canada streamgauge data may be imported with utilities in the RavenR package. The .rvh file is likely best prepared with the assistance of a healthy GIS database which can be used to determine unique class combination and the topology of the watershed subbasins. Note that, if the size of .rvt or .rvh files becomes unwieldy, the `:RedirectToFile` command can be used to redirect the input from an 'extra' input file, so a model could, for instance, have a single master .rvt file that points to a number of meteorological forcing files (e.g., one or more .rvt file per gauge). A similar approach also enables the testing of multiple climate scenarios without having to overwrite data files.

2.3 Running the Model

Once all of the necessary components of the above files have been created, the model may be called from the command line, e.g.,

```
> C:\Program Files\Raven\Raven.exe C:\Temp\model_dir\modelname
```

or, if the active directory is `C:\Temp\model_dir\`

```
> C:\Program Files\Raven\Raven.exe modelname
```

where 'modelname' is the default predecessor to the .rvi, .rvh, .rvt, and .rvp extensions. There are no special flags needed, just the name of the model. The command line also supports the following flagged commands:

- `-o {output directory}` : specifies the directory for generated model output
- `-p {rvp_filename.rvp}` : specifies the rvp file location
- `-t {rvt_filename.rvt}` : specifies the rvt file location
- `-c {rvc_filename.rvc}` : specifies the rvc file location
- `-h {rvh_filename.rvh}` : specifies the rvh file location
- `-r {runname}` : specifies the run name for the simulation

Alternatively, the `:OutputDirectory` command in the .rvi file may be used to specify file output location and the `:rv*_Filename` command may be used to specify the corresponding files (see the details in appendix A.3).

A useful application of the output directory flag is to specify an output directory in the folder directly beneath the working directory, for instance:

```
> C:\Program Files\Raven\Raven.exe modelname -o .\output\
```

Raven will will create this specified output folder if it does not exist.

Note that while it is allowed that the input files from multiple models exist in a single folder, it is recommended that each model get its own output directory to avoid overwriting of outputs.

2.4 Output Files

RAVEN generates a number of customizable outputs which contain model diagnostics. By default, RAVEN generates the following files:

- `Hydrographs.csv` - the hydrograph output file
  Contains the flow rates, $Q(t)$ [m^3d^{-1}], at the outlets of specified subbasins in the watersheds (usually corresponding to those with stream gauges). Which subbasin outlets are recorded as hydrographs is specified in the .rvh file.
- `WatershedStorage.csv` - the watershed storage file
  Contains watershed-averaged water storage in all of the modeled compartments over the duration of the simulation. Also reports watershed-wide water mass balance diagnostics.
- `solution.rvc` - the solution file
  Stores the complete state of the system at the end of the simulation. This file can be used as initial conditions for a later model run. This file may also be generated at user-specified intervals during simulation as a defense against computer breakdown for massive computationally-demanding models.
- `RavenErrors.txt` - the errors file includes all of the warnings and errors for a particular model run, including when the model may be making choices on behalf of the modeler (i.e., parameter autogeneration) or when model input is somehow flawed.

The formats of these files are described in appendix B, and may be pre-appended with the runname if the `:RunName` command is used, generating (for example), `Alouette41_Hydrographs.csv` if the run name is Alouette41. `RavenErrors.txt` is never given a prefix.

In addition to the above, the following output files may be created on request:

- `WatershedMassEnergyBalance.csv` - the watershed flux diagnostics file
  Contains watershed-averaged water and energy fluxes from each hydrological process over time. (enabled using the `:WriteMassBalanceFile` command)
- `WatershedEnergyStorage.csv` - the watershed energy diagnostics file
  Contains watershed-averaged storage in all of the modeled compartments over the duration of the simulation. (enabled using the `:WriteEnergyStorage` command)
- `parameters.csv` - the parameters file
  Stores a list of specified and auto-generated parameters for all soil, land use, topography, and vegetation classes. (enabled using the `:WriteParametersFile` command)
- `forcings.csv` - the forcing functions file
  Stores the complete time series of all watershed-averaged forcing functions over the domain (i.e., rainfall, snowfall, incoming radiation, etc.) (enabled using the `:WriteForcingFunctions` command)
- `ExhaustiveMB.csv` - exhaustive mass balance file
  Stores all state variables in all HRUs over time. Given the potential size of this file, this option should be used sparingly (enabled using the `:ExhaustiveMassBalance` command.)
- `ReservoirStages.csv` - reservoir stage history file
  Stores the time history of reservoir stages for all simulated reservoirs. Requires at least one reservoir in the model.
- `diagnostics.csv` - model quality diagnostics
  reports metrics characterising of fit between the model results and any user-specified observations. This output is enabled using the `:EvaluationMetrics` command, and requires at

least one set of observation data (`:ObservationData` in the .rvt file) to be generated.

- `state files` - model intermediate state files
  similar to solution.rvc, except output at intermediate times specified using the `:OutputDump` or `:MajorOutputInterval` commands. The files are named using the output timestamp, e.g., `RunName_state_20011001.rvc`, and may be used as initial conditions for later simulation runs.

Lastly, custom output commands can be used to track and store in .csv or .tb0 flat files any parameter or state variable in the model over time. This data may be aggregated either temporally or spatially, so that the user may generate files containing, e.g., basin-averaged hydraulic conductivity of the top soil layer at the daily timescale, or monthly averaged evaporation from the canopy in the 23rd HRU. The details of this custom output are in the discussion of the `:CustomOutput` command in the .rvi file (appendix A.1.4).

Additional output files generated by the transport routines are discussed in chapter 7.

2.4.1 Alternative Output Formats

For compatibility with the Green Kenue^{TM} software interface, the option is also available to generate output in .tb0 (Green Kenue^{TM} tabular) format. Custom output will be written to a .tb0 table output file if the `:WriteEnsimFormat` parameter in the .rvi file is set to "yes" and a .csv file if set to "no" (or by default if the command is not included).

2.5 Calibration, Visualization, and Uncertainty Analysis

Unlike many hydrological modeling tools, the Raven software package intentionally does not include any methods for calibration, uncertainty analysis, plotting, or complex statistical analysis. All of these tools are best addressed using flexible and generic pre-and post-processing tools. Some recommendations:

- Green Kenue^{TM}
  An advanced data preparation, analysis, and visualization tool for hydrologic modellers, which supports some Raven features and provides useful post-processing tools for Raven output as well as direct access to Canadian hydrologic data repositories
- Ostrich
  A model-independent multi-algorithm optimization and parameter estimation tool. Ostrich can be used to calibrate Raven models, generate Monte Carlo simulations, and much, much more...
- R
  An open-source software environment for statistical computing and scientific graphics.
- RavenR
  A set of R utilities available from the Raven website. Requires the R open-source software environment.
- mc-stan.org
  An open-source software environment for Bayesian inference and maximum likelihood estimation

- WhiteBox GAT
  An open-source software (with user inteface) for geographic analysis, visualization, terrain analysis, and watershed delineation.

Note that the model quality diagnostics generated using the `:EvaluationMetrics` command may be utilised to support the calibration process.

2.6 Common Run Approaches

The following section describes suggested methods for running Raven in a mode other than straightforward simulation of a single model with a single set of inputs.

- Multiple Climate Scenarios
  For running multiple climate scenarios using a single model, it is recommended to fix the .rvc, .rvp, and .rvh files. Different .rvt files should be generated for the specific climate scenarios. Individual runs would be generated by modifying the rvt filename (using the `:rvtFilename` command in the .rvi file) and the run name (using the `:RunName` command in the .rvi file).
- Multiple Parameter Sets
  It is common to run a model using multiple parameter sets in order to assess the uncertainty or sensitivity of its predictions to changes in input (as done in, e.g., Markov Chain Monte Carlo). For such an approach, it is recommended (if not using software such as Ostrich), to generate multiple .rvp files, keeping the remainder of the data files fixed. Individual runs would be generated by modifying the rvp filename (using the `:rvpFilename` command in the .rvi file) and the run name (using the `:RunName` command in the .rvi file).

2.7 Troubleshooting Raven

While Raven will generally try to tell you when a mistake in the input files will cause problems, there are times when the interface will hang or input will be noticeably erroneous without providing a warning or error in `RavenErrors.txt` (note that Raven is designed to produce significant errors when something goes wrong rather than subtle undetectable errors). These unchecked errors are most commonly due to missing or erroneous input forcing or parameter data, though it may occasionally be due to a genuine bug in the Raven code.

The following steps may be taken to diagnose and repair issues with Raven.

1. *Check the RavenErrors.txt file* in the output directory. Often, the error messages and warnings will contain sufficient information to diagnose and repair the problem. This is always the best first step.
2. *If the model hangs prior to the beginning of simulation.* Add the command `:NoisyMode` to the .rvi file. This must be after any call to `:SilentMode` (these commands toggle the same internal switch), but ideally at the top of the input file. Running the code in noisymode generates detailed narrative output to the command prompt window, and is best for diagnosing errors in input parsing. By looking at where the code hangs, the problematic input command can often be found. See if the model runs with this command commented out. If it does, there may be (a) a missing input parameter for the chosen method/algorithm (b) erroneous input data linked to this method/algorithm that Raven is not currently able to detect.

3. *If the model runs to completion but generates clearly erroneous output* (such as NaN or -#inf in the hydrograph output).

   This type of error is likely due to (a) a missing model parameter which Raven did not detect; (b) an error in the Raven modelling library or (c) an error in input which Raven did not detect.

   (a) Step 1: Open the ForcingFiles.csv output file and look for non-sensible numerical values (e.g., negative PET or NaN radiation). These errors in Forcing Functions will propagate through the model and generate hydrograph errors. Comment out or modify the corresponding forcing function commands (catalogued in section A.1.2 of the appendix) until the faulty forcing output not generated. For instance, if the PET is consistently negative, replace the PET estimation or PET orographic correction algorithm with another method. If the errors are fixed, then this may be due to poor parameters which drive this method. If the errors remain, then data which is used to drive PET estimation may be faulty OR one of the other forcing functions which drives PET (such as shortwave radiation, temperature, etc.) is faulty. The latter would also be obvious from a cursory inspection of the ForcingFiles.csv output.

   (b) Step 2: If the forcing functions are not the culprit, then examine the WatershedStorage.csv file and check for clearly erroneous estimates of watershed-averaged water storage. If, for example, glacial storage looks faulty but everything else is OK, comment out the algorithms which operate on glacial storage in the `:HydrologicProcesses` block in the .rvi file and re-run until the glacial storage results are feasible (perhaps monotonically growing or shrinking, but not NaN or hugely negative). This narrows us down to the problematic process algorithm. Check the documentation to make sure that the proper parameters are provided for this algorithm in the .rvp file for all glacier HRUs. If you still cannot diagnose the problem, send the problematic input files with a short description to jrcraig@uwaterloo.ca.

4. *If the model is providing odd/unexpected output.*

   Sometimes generated hydrographs are not completely broken, but are at odds with our expectations. For example, outflows are 10 times larger or smaller than they should be when compared to the observed hydrographs. These issues are much thornier, as they can arise from individually reasonable (but collectively unreasonable) combinations of parameter inputs. They are also quite possible if you are building a model from scratch with Raven, and have done so improperly (e.g., Raven technically allows you the flexibility to have two evapotranspiration processes, but it is physical nonsense to implement this). There are some general approaches you may take towards debugging this kind of model issue.

   (a) Look at the `WatershedStorage.csv` file for clues. Most watersheds should have a quasi-steady state behaviour from year to year; there may be wet years and dry years, but storage in general oscillates and repeats a relatively consistent water balance from month to month. If your model is a continuous model of three or more years, you should expect this type of oscillatory behavior. If you find that one storage compartment is steadily increasing or decreasing in storage, it may be worthwhile to investigate the cause. In many cases, the inflow/outflow processes are not properly matched, e.g., a middle soil storage unit may be filled due to percolation at a much faster rate than it depletes due to baseflow losses, even at the annual scale. Another possible symptom that may be seen in the `WatershedStorage.csv` file is a storage compartment which always fills but never drains (or the opposite). Some storage units are intended to have this behavior, such as `ATMOS_PRECIP` (which is always a water source, and is a proxy for cumulative precipitation) and `ATMOSPHERE` (which is always a water sink, and is a proxy for cumulative evapotranspiration losses). Others, such as deep `GROUNDWATER`, may be used to represent external losses from the system. However, any other storage unit should have means of decreasing and increasing in storage, as determined by the hydrological process list (each storage unit should act as a To and From storage unit), and the parameter lists.

(b) Look at the ForcingFunctions.csv file for clues. Again, poor parameter choices can lead to significant underestimates or overestimates of system forcings, which propagate through to hydrographs and other model outputs. Look for reasonable values for radiative, precipitation, and temperature forcings to the watershed. What constitutes reasonable is specific to the climate and landscape, and is up to you to define.

(c) Check your stream network topology. The surface water network is fully defined by the list of `DOWNSTREAM_IDs` in the :SubBasins command. If this is improperly constructed, or if the entirety of an upstream watershed is not included in the model, you may need to either correct the stream network or add user-specified inflows to account for upstream parts of the watershed not explicitly included in the model.

(d) Check your cumulative watershed area. The area of each subbasin, and therefore also the total drainage area of each subbasin, is dependent upon the areas of its constituent HRUs. If these areas are incorrect, or if certain HRUs are not included in the model, this can lead to mass balance errors.

(e) Check the units of your forcing functions. A common mistake for subdaily flow information is to supply precipitation in mm rather than as a precipitation intensity in mm/d.

2.8 Version Notes

2.8.1 Major Changes from v2.6 to v2.7

The following features have been added:

1. Improvements to the Raven Documentation
2. Bug fixes
3. Improved QA/QC on model inputs
4. Significantly improved support for flexible reservoir simulation and calibration - time-varying reservoir curves, unevenly spaced reservoir curves,
5. Support for gridded data in netCDF format (see appendix A.4.3)
6. Improved place- and time-specific control over application of processes using the `:-->Conditional` command, `:LandUseChange` command, and `:VegetationChange` command.
7. `:CreateRVPTemplate` command can be used to generate a template .rvp file from specified .rvi model configuration
8. Added a number of new diagnostics (`LOG_NASH`, `NASH_DERIV`, `KLING_GUPTA`)
9. Addition of the GAWSER-style snow balance and consolidation routine

10. Addition of US Army Corps snowmelt model
11. Run Name can be specified from the command line

The following backwards compatibility issues were introduced:

1. None

2.8.2 Major Changes from v2.5 to v2.6

The following features have been added:

1. Significant improvements to the Raven Documentation
2. Support for additional model quality diagnostic (R2)
3. Improved support for sub-daily emulation of the UBC watershed model
4. New elevation-based gauge interpolation algorithm (`INTERP_INVERSE_DISTANCE_ELEVATION`)
5. New two-layer snow melt model (`SNOBAL_TWO_LAYER`)
6. Improved support for blank observation values and non-zero observation weights in model diagnostics

The following backwards compatibility issues were introduced:

1. The hydrograph observations file is now written in period-starting (rather than period-ending) format, meaning that the single time step correction to the start date of a continuous observation hydrograph time series is no longer needed. **ACTION**: Existing observation .rvt files will have to be amended with a simple date shift.
2. For models with more than one subbasin where the reference or initial stream discharges were not user-specified, the algorithm used to estimate basin initial and reference flows has been significantly modified. Automatic estimation of network flows now requires the specification of the :AnnualAvgRunoff command in the .rvp file. **ACTION**: Recalibration of existing models will likely be required if `Q_REFERENCE` was not user-specified for all basins and a celerity-dependent routing algorithm was used (e.g., a Muskingum variant, plug flow, or diffusive wave).
3. For models with more than one gauge and gauge-specific `:SnowCorrections` and `:RainCorrections`, the interpolation algorithm has been modified to more appropriately handle the spatial handling of these corrections. **ACTION**: Recalibration of existing models may be required.

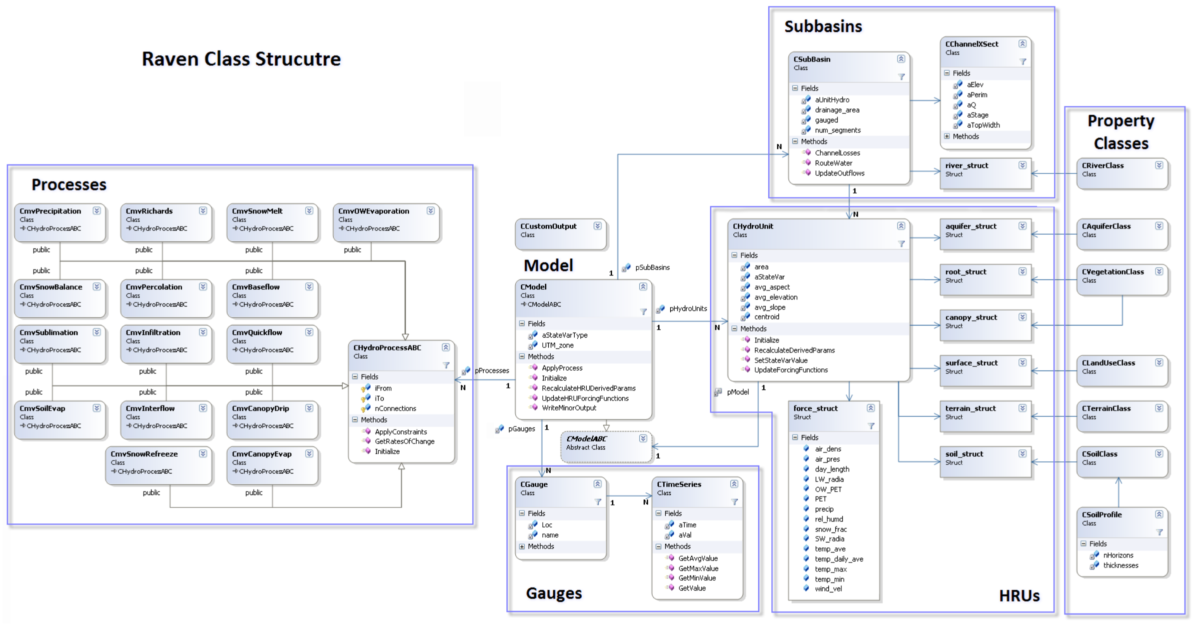
Chapter 3

Raven Code Organization*

The Raven code is fully object-oriented code designed to, as much as possible, separate the numerical solution of the coupled mass-balance and energy-balance ODEs from the evaluation of flux-storage relationships, enabling the testing of various numerical schemes without having to dig into each subroutine for each hydrological process.

3.1 Classes

The Class diagram for the Raven code is depicted in figure 3.1. The code operates by generating a single instance of the `CModel` class, which may be considered a container class for all of the model data, i.e. the arrays of basins, HRUs, land/vegetation classes, and meteorological gauges/gridded forcing data that define the entirety of the model.



Raven Class Strucutre
Processes
CmvPrecipitation Class CHydroProcessABC
CmvRichards Class CHydroProcessABC
CmvSnowMelt Class CHydroProcessABC
CmvOWEvaporation Class CHydroProcessABC
CmvSnowBalance Class CHydroProcessABC
CmvPercolation Class CHydroProcessABC
CmvBaseflow Class CHydroProcessABC
CmvSublimation Class CHydroProcessABC
CmvInfiltration Class CHydroProcessABC
CmvQuickflow Class CHydroProcessABC
CmvSoilEvap Class CHydroProcessABC
CmvInterflow Class CHydroProcessABC
CmvCanopyDrip Class CHydroProcessABC
CmvSnowRefreeze Class CHydroProcessABC
CmvCanopyEvap Class CHydroProcessABC
public
CHydroProcessABC Class
Fields iFrom iTo nConnections
Methods ApplyConstraints GetRatesOfChange Initialize
pProcesses
N
1
CCustomOutput Class
Model
CModel Class CModelABC
Fields aStateVarType UTM_zone
Methods ApplyProcess Initialize RecalculateHRUDerivedParams UpdateHRUForcingFunctions WriteMinorOutput
CModelABC Abstract Class
pSubBasins
pHydroUnits
pGauges
pModel
Subbasins
CSubBasin Class
Fields aUnitHydro drainage_area gauged num_segments
Methods ChannelLosses RouteWater UpdateOutflows
CChannelXSect Class
Fields aElev aPerim aQ aStage aTopWidth
Methods
river_struct Struct
CHydroUnit Class
Fields area aStateVar avg_aspect avg_elevation avg_slope centroid
Methods Initialize RecalculateDerivedParams SetStateVarValue UpdateForcingFunctions
force_struct Struct
Fields air_dens air_pres day_length LW_radia OW_PET PET precip rel_humd snow_frac SW_radia temp_ave temp_daily_ave temp_max temp_min wind_vel
aquifer_struct Struct
root_struct Struct
canopy_struct Struct
surface_struct Struct
terrain_struct Struct
soil_struct Struct
HRUs
Gauges
CGauge Class
Fields Loc name
Methods
CTimeSeries Class
Fields aTime aVal
Methods GetAvgValue GetMaxValue GetMinValue GetValue
Property Classes
CRiverClass Class
CAquiferClass Class
CVegetationClass Class
CLandUseClass Class
CTerrainClass Class
CSoilClass Class
CSoilProfile Class
Fields nHorizons thicknesses

Figure 3.1: Raven class diagram

3.1.1 CModel class

The `CModel` class is a container class for all of the hydrological response units (HRUs), subbasins, hydrologic processes ("HydroProcesses") and measurement gauges/gridded data. It also has global information about all of the state variables. It has a few key functions called by the solver routines:

- `Initialize()` Called before the simulation begins to initialize all variables. This also calls all Subbasin, Gauge, HRU and other initialize functions.
- `IncrementWB()`, `IncrementEB()` increment the individual cumulative HRU water and energy balances, stored within the CModel class
- `WriteMinorOutput()` Called at the end of each timestep, writes water and energy balance and watershed-scale storage information (i.e., total storage in snowpack, etc.), in addition to all custom output.
- `WriteMajorOutput()` Called at user-specified intervals, basically dumps a snapshot of all system state variables and derived parameters to an output file
- `UpdateHRUForcingFunctions()` sifts through all of the HRUs and updates precip, temperature, radiation, and other (external) atmospheric forcing functions, interpolated from gauge/measurement data or gridded forcings. These values are then stored locally within each HRU. Called at the start of each timestep.
- `ApplyProcess()` Based upon some assumed current water storage/state variable distribution, returns a prediction of the rate of water (or energy) movement from one storage unit (e.g., canopy) to another (e.g., atmosphere) during the time step. This function DOES NOT actually move the water/energy - this is done within the solver. Basically returns $\mathbf{M}^k(\{\phi\},\{P\})$ in the above discussion for specified values of $\{\phi\}$

The CModel class has an abstracted parent class, CModelABC, that ensures the model can only provide information to, but cannot be modified by, other classes aware of its existence (e.g., any hydrologic processes (`CHydroProcess`), or subbasin (`CSubBasin`), etc.)

3.1.2 CGauge class

The CGauge class stores a set of time series (of class CTimeSeries) corresponding to observations of atmospheric forcing functions (precipitation, air temperature, radiation, etc.) at a single point in the watershed. The model interpolates these forcing functions from gauge information in order to determine forcing functions for individual HRUs at any given time step.

Interpolation is performed using the most appropriate local UTM coordinate system automatically calculated from the specified lat-long centroid of the watershed.

3.1.3 CSubBasin class

A container class for HRUs - only used for routing of water, as it stores information about the connectedness of itself to other subbasins in the modeled watershed(s). Conceptualized as a subbasin.

3.1.4 CHydroUnit class

An abstraction of an HRU - a homogeneous area of land to which the zero- or one-dimensional water and energy balances are applied. It is unaware of the CModel class. It stores the state of all local HRU-specific parameters that are valid for the current timestep, the values of the HRU forcing functions (e.g., precipitation, PET, radiation) averaged over the entirety of the current timestep, and the values of the state variables (water storage, energy storage, and snow parameters) that are valid at the start of the current timestep. It also stores its membership to the landuse and vegetation cover classes via pointers to those instances, so that it may be used to access properties shared by all measures of that class.

Key routines:

- `SetStateVarValue()` updates the values of a specific state variable. Called at the end of each time step by the main Raven solver
- `UpdateForcingFunctions()` updates the values of the forcing functions (rainfall, temperature, saturated water vapor, etc.) uniformly applied to the HRU at the beginning of each time step. The HydroUnit is unaware of the source of these values, but they are interpolated from measured data.
- `RecalculateDerivedParams()` Given some set of state variables and the current time of year, updates all derived parameters (e.g., Leaf area index) stored locally within the HRU. These are used within `GetRatesOfChange` functions

3.1.5 CHydroProcessABC class

An abstraction of any hydrological process that moves water or energy from one or more storage units to another set of storage units (i.e., an abstraction of M_{ij} for one-to-one transfer of water/energy, or a summation of more than one M_{ij} that moves water through multiple compartments, as is required for PDE solution). Each CHydroProcess child class has three key subroutines:

- `Initialize()` initializes all necessary structures, etc. prior to solution
- `GetParticipatingStateVars()` returns the list of participating state variables for the model. This is used to dynamically generate the state variables used in the model. For example, snow will not be tracked in the model until a process (e.g., snowmelt) is introduced that moves snow between storage compartments.
- `GetParticipatingParameters()` returns the list of algorithm-specific parameters needed to simulate this process with the specified algorithm. This is used to dynamically ensure that all parameters needed by the model are specified by the user within each HRU.
- `GetRatesOfChange()` calculates and returns rate of loss of one set of storage units to another set, in units of mm/d (for water) or MJ/m^2/d for energy.
- `ApplyConstraints()` Corrects the rates calculated by rates of change to ensure that model constraints (e.g., state variable positivity) are met.

The CHydroProcessABC class is purely virtual - inherited classes each correspond to a single (or coupled set of) hydrologic process(es) as described in section 3.1.6

3.1.6 Hydrological Processes

All hydrological process algorithms are specified as individual child classes of `CHydroProcessABC`. Note that each HydroProcess may include multiple algorithms; distinction between classes is mostly based upon physical interpretation, i.e., baseflow and snowmelt are fundamentally different. While independent snow melt/snow balance algorithms may be very different, they are still grouped into one class.

3.2 Contributing to the Raven Framework*

Source code for RAVEN is available online, with file support for MSDN Visual C++, both 2008 and 2010 version. Users are encouraged to develop custom-made algorithms for representing hydrologic processes, estimating forcing functions from gauge data, or interpolating gauge data. If a new algorithm is tested and found useful, feel free to submit your code to the RAVEN development team to be considered for inclusion into the main RAVEN code.

3.2.1 How to Add a New Process Algorithm

1. Make sure the process algorithm is not already included in the framework with a slightly different "flavour"
2. Determine whether the algorithm requires new state variable types to be added to the master list. The complete list of state variables currently supported may be found in the `enum sv_type` definition in `RavenInclude.h`. If a new state variable is required, follow the directions in section 3.2.2.
3. Determine whether the algorithm requires new parameters, and whether these parameters will be fixed for the model duration or depend upon transient factors. The lists of existing parameters (all linked to soils, vegetation, land use, or terrain types) are found in `Properties.h`. If a new parameter is needed, follow the directions in section 3.2.3
4. Determine whether the algorithm fits within an existing `CHydroProcess` class, i.e., is it a different means of representing one of the many processes already simulated within RAVEN? If so, you will be editing the code in 6 or 7 places, all within either the `CHydroProcess` header/source files or the main input parsing routine:
   (a) Add a new algorithm type to the enumerated list of algorithms for that process. For example, if it is a new baseflow algorithm, you would add `BASE_MYALGORITHM` to the `enum baseflow_type` in `SoilWaterMovers.h`. Follow the apparent naming convention.
   (b) Edit the `CHydroProcess` constructor. Constructors should be dynamic for all routines that have fixed input and output variables. Others, such as baseflow, can have user-specified input/output pairs declared. The `CmvBaseFlow` and `CmvSnowBalance` codes are excellent templates for class construction. Edit the if-then-else statement in the constructor, specifying the `iFrom` and `iTo` state variables manipulated by the algorithm connections. For example, most infiltration algorithms move water from ponded storage to both topsoil and surface water, requiring the following specification:

```
CHydroProcessABC::DynamicSpecifyConnections(2);
iFrom[0]=pModel->GetStateVarIndex(PONDED_WATER);
```

```
iTo  [0]=pModel->GetStateVarIndex(SOIL,0);
iFrom[1]=pModel->GetStateVarIndex(PONDED_WATER);
iTo  [1]=pModel->GetStateVarIndex(SURFACE_WATER);
```

This creates two connections, one from ponded water to the topmost soil (soil 0) and one from ponded water to surface water. The corresponding rates of exchange will later be calculated in `GetRatesOfChange()` and stored in `rates[0]` and `rates[1]`. Note you shouldn't have to check for existence of state variables in the constructor - if they are later specified in `GetParticipatingStateVarList`, they will be generated in the master state variable list prior to instantiation of the class.

(c) Edit the if-then-else statement in the corresponding `GetParticipatingParamList` routine with the list of parameters needed by your new algorithm. This information is used for quality control on input data (ensuring that users specify all parameters needed to operate the model).

(d) Edit (if necessary) in `GetParticipatingStateVarList` the list of state variables required for your algorithm, within a conditional for your specific algorithm. See `CmvSnowBalance` for a good example.

(e) Add the actual flux calculation algorithm to the corresponding `GetRatesOfChange()` function for this `CHydroProcesss` class. Some key things to keep in mind:
(a) parameters may be obtained from the corresponding soil, vegetation, or land use structure via the HRU pointer, e.g.,

```
double lambda,K;
K     =pHRU->GetSoilProps(m)->max_baseflow_rate;
lambda=pHRU->GetTerrainProps()->lambda;
```

(b) the final result of the algorithm (rates of change of modeled state variables) are assigned to the `rates[]` array. The `rates[i]` array value corresponds to the flux rate of mass/water/energy from state variable `iFrom[i]` to `iTo[i]`, which you have defined in the constructor (step b).
(c) Try to follow the following code habits:

- unless required for emulation of an existing code, constraints should not be used except later in the `ApplyConstraints` routine. A good rule of thumb is that the timestep should not appear anywhere in this code
- each algorithm longer than about 20-30 lines of code should be relegated to its own private function of the class
- all unit conversions should be explicitly spelled out using the provided global constants, defined in `RavenInclude.h`
- constants that might be used in more than one process subroutine should not be hard-coded, where at all possible.
- references should be provided for all equations, where possible. The full reference should appear in the back of this manual
- all variables should be declared before, not within, algorithm code

- All returned rates should be in mmd^{-1} or MJ/m^2/d for water storage and energy storage, respectively

(f) If needed, add special state variable constraints in the `ApplyConstraints()` function, conditional on the algorithm type.

(g) Lastly, add the process algorithm option to the corresponding command in the `ParseMainInputFile(` routine within `ParseInput.cpp`.

3.2.2 How to Add a New State Variable

1. Make sure the state variable is not already included in the framework with a slightly different name. Note that proxy variables should be used cautiously. For example, right now snow (as SWE) and snow depth are included in the variable list, while snow density is not (as it may be calculated from the other two).
2. Add the state variable type to the `sv_type` enumerated type in `RavenInclude.h`
3. Edit the following routines in the `CStateVariables` class (within `StateVariables.cpp`) (revisions should be self-evident from code):
   - `GetStateVarName()`
   - `StringToSVType()`
   - `IsWaterStorage()`
   - `IsEnergyStorage()`
4. Edit the `CHydroUnit::GetStateVarMax()` routine in `HydroUnits.cpp` if there is a maximum constraint upon the variable

3.2.3 How to Add a New Parameter

1. Make sure that the parameter is not included in the framework by examining the available parameters in the `soil_struct`, `canopy_struct`, `terrain_struct` defined in `Properties.h` and the global parameters currently defined within the `global_struct` (`RavenInclude.h`). If it is not, determine whether the parameter is (and should always be) global (i.e., not spatially or temporally varying). If it is not global, determine whether the property is best tied to land use/land cover, soil type, vegetation type, or terrain type.
2. Add the new global parameter to the `global_struct` structure, non-global parameters to the corresponding `soil_`, `canopy_`, `terrain_`, or `surface_struct`. The units of the parameter should generally be consistent with those used throughout Raven, i.e., SI units, with fractions represented from 0 to 1 (not 1-100%), time units preferably in days, and energy in MJ.
3. Depending upon the type of parameter, different classes will have to be revised. As an example, if it is a soil parameter, the following code must be revised:
   - `CSoilClass::WriteParamsToFile()` (revisions evident from code)
   - `CSoilClass::AutoCalculateSoilProps(()` In most cases, the new parameter will be conceptual and therefore not autocalculable from the base parameters of soil composition. In this case, code may be replicated from other parameters (see, e.g., `VIC_zmin` code for an example.

- `CSoilClass::InitializeSoilProperties()` (revisions evident from code)
- `CSoilClass::SetSoilProperty()`(revisions evident from code)
- `CSoilClass::GetSoilProperty()`(revisions evident from code)

Similar functions exist in the alternate classes (e.g., `CVegetationClass, CGlobalParams`).
With these revisions, the parameter is now accessible via (for soils) `pHRU->GetSoilProps(0)->new_param`, where `pHRU` is a pointer to a specific instantiated HRU. New global parameters (which are not specific to an HRU) may be accessed via `CGlobalParams::GetParams->new_param`

3.2.4 How to Add a New Forcing Estimator

[UNDER CONSTRUCTION] **To do** (1)

Chapter 4

The Hydrological Process Library

The following chapter outlines the many process algorithms available for modelling the hydrological cycle in Raven.

4.1 Precipitation Partitioning

The precipitation partitioning process moves water, in the form of snow and rain, to the appropriate storage compartment. The order of application is depicted in figure 4.1. The specific distribution of rainfall and snowfall to the canopy, and ground surface (in the form of ponded water) depends upon the existence of particular storage compartments and a number of model parameters.

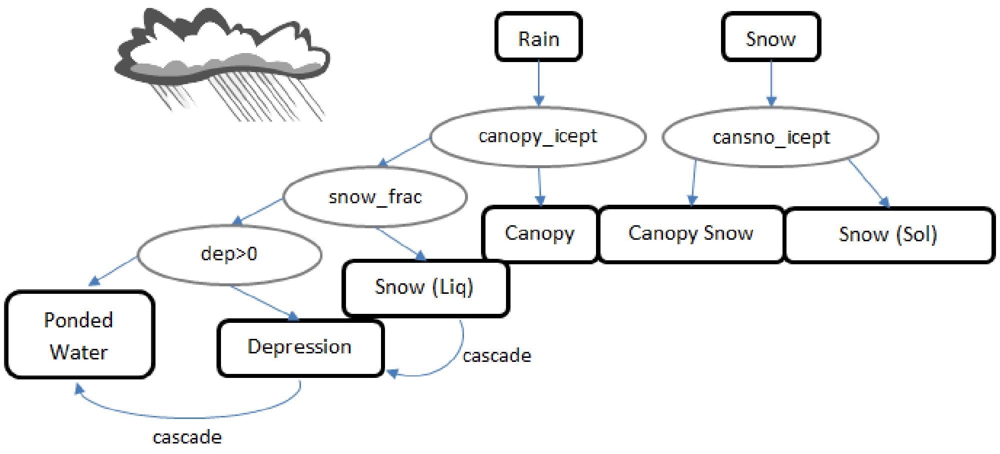


Figure 4.1: Partitioning of rainfall/snowfall to the appropriate surface storage compartments

The partitioning of precipitation proceeds as follows (for non-lake HRUs):

1. The amount of rain and snow captured by the vegetation canopy is controlled by the precipitation interception rate (calculated as described below) and the storage capacity of the canopy. If the canopy exists as a storage state variable (i.e., `CANOPY` or `CANOPY_SNOW`) are present in the model, these storage compartments are filled at the calculated interception rate until filled. The remainder (if any) is allowed to proceed onward, with a correction included

for the percent forest cover, (land use parameter `FOREST_COVER`). If canopy water/snow storage is not explicitly modeled, the amount of available canopy storage is not considered and the amount of snow and rain that would be captured by the canopy is "evaporated" to the atmosphere.

2. If there is a snow state variable in the model (determined usually by the presence of some kind of snow balance or snow melt algorithm), the snow as SWE is increased by an amount corresponding to snowfall. If rain hits the snowpack, it fills the unripe pores in the snowpack and is allowed to proceed onward. If required by the model, cold content, and snow density may also be updated. Some of the snow balance algorithms override the details of this process, instead moving all snowfall to `NEW_SNOW` and all rainfall to `PONDED_WATER` where it waits to be handled by the snow balance algorithm.

The water in the `PONDED_WATER` storage compartment, which typically also includes meltwater from snow melt, waits to be distributed to the shallow subsurface or surface water storage through subsequent application of an infiltration or abstraction algorithm.

For lake HRUs, all snow and rain are converted to liquid water and added directly to the `SURFACE_WATER` store, ready to be routed downstream via in-catchment routing. Lake HRUs are defined as those with a zero-layer soil profile whose name begins with `LAKE`.

4.1.1 Canopy Interception Algorithms

The canopy interception algorithms, specified by the model command `:PrecipIceptFract` are used to determine the percent rain or snow captured by a full forest/crop canopy. In all cases, the maximum interception rates are given as

$$\begin{aligned} R_{int} &= \theta_{rain} \cdot R \\ S_{int} &= \theta_{snow} \cdot S \end{aligned}$$

where R and S are snowfall rates, in [mm/d], R_{int} and S_{int} are interception rates, in mm/d, and $\theta_{rain}/\theta_{snow}$ are the interception percentages (values between 0 and 1). These maximum interception rates may be limited (as mentioned above) by the current amount of water stored in the canopy. Many of these rates are controlled by leaf area index, LAI, and stem area index, SAI, calculated as follows:

$$\begin{aligned} \mathrm{LAI} &= (1-s) \cdot \mathrm{LAI_{max}} \cdot \mathrm{f_{LAI}(m)} \\ \mathrm{SAI} &= (1-s) \cdot \beta \cdot h_{veg} \end{aligned}$$

where s is the land use parameter `FOREST_SPARSENESS`, LAI$_{max}$ is the maximum LAI (vegetation parameter `MAX_LAI`, $f_{LAI}(m)$ is the relative LAI correction by month m, specified by the `:SeasonalRelativeLAI` command for each vegetation type, and β is the vegetation parameter `SAI_HT_RATIO`. The height of vegetation, h_{veg} is calculated as

$$h_{veg} = h_{max} \cdot f_{veg}(m)$$

where h_{max} is the maximum vegetation height (vegetation parameter `MAX_HT`) and $f_{veg}(m)$ is the relative vegetation height correction by month m, specified using the `:SeasonalRelativeHeight` command in the .rvp file.

The following algorithms are used to determine the percentages of rain and snow that will be intercepted by the vegetative canopy:

User-specified throughfall fraction(PRECIP_ICEPT_USER)

The interception percentages are directly specified by the user θ_{rain} is the vegetation parameter `RAIN_ICEPT_PCT` and θ_{rain} is the vegetation parameter `SNOW_ICEPT_PCT`.

Linear LAI-based method (PRECIP_ICEPT_LAI)

From Dingman (2002), the interception percentages are given as a linear function of the LAI:

$$\begin{aligned}\theta_{rain} &= \alpha_{rain} \cdot (\text{LAI} + \text{SAI}) \\ \theta_{snow} &= \alpha_{snow} \cdot (\text{LAI} + \text{SAI})\end{aligned}$$

where α_{rain} and α_{snow} are the vegetation parameters `RAIN_ICEPT_FACT` and `SNOW_ICEPT_FACT`, respectively. The leaf area index LAI and stem area index SAI are calculated as indicated above.

Exponential LAI-based method (PRECIP_ICEPT_EXPLAI)

The interception percentages are given as:

$$\begin{aligned}\theta_{rain} &= 1 - \exp(-0.5(\text{LAI} + \text{SAI})) \\ \theta_{snow} &= 1 - \exp(-0.5(\text{LAI} + \text{SAI}))\end{aligned}$$

Exponential LAI-based method (PRECIP_ICEPT_HEDSTROM)

As documented in Hedstrom and Pomeroy (1998).

4.2 Infiltration

Infiltration refers to the partitioning of ponded water (the residual rainfall and/or snowmelt) between the shallow surface soil (infiltrated water) and surface water (runoff). Infiltration is typically controlled by the saturation of the soil and its hydraulic properties (e.g., hydraulic conductivity, infiltration capacity).

4.2.1 Sources/Sinks

Infiltration always moves water from `PONDED_WATER` to `SOIL[0]` (the top soil layer), and depending upon the soil structure model specified by the `:SoilModel` command, may additionally push water to lower soil moisture stores. The remaining ininfiltrated water is typically treated as runoff and moved to `SURFACE_WATER`.

4.2.2 Constraints/Notes

Infiltration is limited by the availability of soil/aquifer storage. Many of the following algorithms use the quantities of maximum soil storage (ϕ_{max}), maximum tension storage (ϕ_{tens}), and field capacity storage (ϕ_{fc}) in a layer, always calculated as:

$$\begin{aligned}\phi_{max} &= Hn(1-SF) \\ \phi_{tens} &= \phi_{max}(S_{fc}-S_{wilt}) \\ \phi_{fc} &= \phi_{max}S_{fc}\end{aligned} \tag{4.1}$$

where H is the soil layer thickness, n is the porosity (soil property `POROSITY`), SF is the stone fraction (soil property `STONE_FRAC`), S_{fc} is the saturation at field capacity (soil parameter `FIELD_CAPACITY`), and S_{wilt} is the saturation at the wilting point (soil parameter `SAT_WILT`).

4.2.3 Infiltration Algorithms

Partition Coefficient Method (`INF_PARTITION_COEFF`)

A simple linear relationship between precipitation and runoff (e.g., Chow et al. (1988)), characterized by:

$$M_{inf} = R \cdot (1 - P_c) \tag{4.2}$$

where M_{inf} is the infiltration rate [mmd^{-1}], R is the rainfall/snowmelt rate [mmd^{-1}] (alternately, the current amount of ponded water divided by the model timestep), and P_c is the partition coefficient, specified as the land use parameter `PARTITION_COEFF`. The remainder of rainfall is routed to surface water.

SCS Method (INF_SCS)

The standard Soil Conservation Society (SCS) method (Soil Conservation Service, 1986), where infiltration is a function of the local curve number:

$$M_{inf} = R \cdot \left(1 - \frac{(R - 0.2S)^2}{R + 0.8S}\right) \tag{4.3}$$

where M_{inf} is the infiltration rate [mmd^{-1}], R is the rainfall/snowmelt rate [mmd^{-1}] (alternately, the current amount of ponded water divided by the model timestep), and S [mm] is the retention parameter

$$S = 25400/CN - 254 \tag{4.4}$$

where CN is the SCS curve number (land use parameter `SCS_CN`. The curve number for moderate antecedent moisture content (condition II) is user-specified with land use parameter `SCS_CN` and corrected for dry or wet conditions based upon 5-day precipitation history and whether or not it is growing season. The SCS method should only be used for daily simulations.

Explicit Green Ampt Method (INF_GREEN_AMPT)

The explicit calculation of Green-Ampt cumulative (Green and Ampt, 1911) infiltration

$$M_{inf} = \min\left(R, k_{sat}\left(1 + \frac{|\psi_f|(\phi_{max} - \phi_{soil})}{F}\right)\right) \tag{4.5}$$

where F uses the n^{th} recursive approximation of the Lambert W_{-1} function (Barry et al., 2005). The variables ψ_f [-mm], ϕ_{max} [mm], and ϕ_{soil} [mm], are the Green-Ampt wetting front suction (soil parameter `WETTING_FRONT_PSI`), maximum soil moisture content (defined in equation 4.1), and soil moisture at the start of the time step. All parameters used are those associated with the top soil.

Simple Green Ampt Method (INF_GA_SIMPLE)

The quick-and-dirty version of the Green-Ampt (Green and Ampt, 1911) analytical solution for discrete time-stepping schemes:

$$M_{inf} = \min\left(R, k_{sat}\left(1 + \frac{|\psi_f|(\phi_{max} - \phi_{soil})}{F}\right)\right) \tag{4.6}$$

where F, the cumulative infiltration, is accumulated as a state variable during simulation, and reverts to zero after prolonged periods without precipitation. The variables ψ_f [-mm], ϕ_{max} [mm], and ϕ_{soil} [mm], are the Green-Ampt wetting front suction (soil parameter `WETTING_FRONT_PSI`), maximum soil moisture content (defined in equation 4.1), and soil moisture at the start of the time step. All parameters used are those associated with the top soil.

VIC Method (INF_VIC)

From the variable infiltration capacity model (Wood et al., 1992):

$$M_{inf} = R \cdot \left(K_1 \left(\gamma \alpha z_{max} + z_{min} - \frac{\phi_{soil}}{\phi_{max}} \right)^{\gamma} \left(1 - \frac{\phi_{crit}}{\phi_{max}} \right)^{-\gamma} \right) \tag{4.7}$$

where $\gamma = 1/(\alpha+1)$, α is the soil parameter VIC_ALPHA, z_{min} and z_{max} are the soil parameters VIC_ZMIN and VIC_ZMAX, and K_1 is given by:

$$K_1 = ((z_{max} - z_{min})\alpha\gamma)^{-\gamma} \tag{4.8}$$

VIC/ARNO Method (INF_VIC_ARNO)

The VIC/ARNO model as interpreted by (Clark et al., 2008).

$$M_{inf} = R \cdot \left(1 - \left(1 - \frac{\phi_{soil}}{\phi_{max}} \right)^{b} \right) \tag{4.9}$$

where b is the soil parameter B_EXP, ϕ_{soil} is the top soil layer water content [mm], and ϕ_{max} is the maximum topsoil storage [mm].

HBV Method (INF_HBV)

The standard HBV model approach (Bergstrom, 1995).

$$M_{inf} = R \cdot \left(1 - \left(\frac{\phi_{soil}}{\phi_{max}} \right)^{\beta} \right) \tag{4.10}$$

where β is the soil parameter HBV_BETA, ϕ_{soil} is the soil layer water content [mm], and ϕ_{max} is the maximum soil storage [mm].

PRMS Method (INF_PRMS)

The PRMS model Leavesley and Stannard (1995) as interpreted by (Clark et al., 2008):

$$M_{inf} = R \cdot \left(1 - F_{sat}^{max} \min \left(\frac{\phi_{soil}}{\phi_{tens}}, 1 \right) \right) \tag{4.11}$$

where ϕ_{soil} is the soil layer water content [mm], ϕ_{tens} is the maximum tension storage [mm], and F_{sat}^{max} is the maximum saturated area fraction (land use parameter MAX_SAT_AREA_FRAC).

UBC Watershed Model Approach (INF_UBC)

As documented in Quick (2003), the UBCWM infiltration algorithm partitions ponded water to surface water, interflow, and two groundwater stores. The infiltration rate into the shallow soil is calculated as

$$M_{inf} = \min \left(\frac{\phi_{max} - \phi_{soil}}{\Delta t}, R \right) \cdot (1 - b_2) \tag{4.12}$$

where, b_2, the effective impermeable area percentage, is calculated using a deficit-based estimate corrected with a special term for flash floods (corresponding to higher rainfall/melt rates):

$$b_2 = b_1 + (1 - b_1) \cdot FF \tag{4.13}$$

here b_1, the unmodified effective impermeable area percentage, calculated as

$$b_1 = F_{imp} \cdot 10^{\left(-\frac{\phi_{max}-\phi_{soil}}{P0AGEN}\right)} \tag{4.14}$$

where ϕ_{soil} and ϕ_{max} are as defined in equation 4.1 and FF, the flash factor (which is constrained to vary between 0 and 1) is calculated as:

$$FF = \cdot \left(1 + \log\left(\frac{\phi_{pond}}{V0FLAX}\right) / \log\left(\frac{V0FLAX}{1800}\right)\right) \tag{4.15}$$

here, F_{imp} [-] is the land use parameter `IMPERMEABLE_FRAC`, $V0FLAS$ [mm] is the global ponding parameter `UBC_FLASH_PONDING`, and $P0AGEN$ [mm] is the soil property `UBC_INFIL_SOIL_DEF`, the reference soil deficit used at which 10 percent of the soil surface generates runoff.

The remaining ponded water is distributed to groundwater (at rate M_{perc}), interflow (at rate M_{int}, and runoff M_{run} using the following expressions

$$M_{perc} = \min\left(M_{max}^{perc}, R - M_{inf}\right) \cdot (1 - b_2) \tag{4.16}$$

$$M_{int} = (R - M_{inf} - M_{perc}) \cdot (1 - b_2) \tag{4.17}$$

$$M_{run} = b_2 \cdot R \tag{4.18}$$

i.e., a percentage b_2 of the rainfall/snowmelt runs off directly. The remainder first infiltrates into the shallow soil, until the deficit is filled. Any remaining water then percolates into the groundwater at a maximum rate M_{max}^{perc} [mmd^{-1}], specified using the `MAX_PERC_RATE` parameter of the groundwater soil layers. This component will be partitioned such that a certain percentage, `UBC_GW_SPLIT` [i], a global parameter specified using the `:UBCGroundwaterSplit` command, goes to the lower groundwater storage, whereas the remainder goes to upper groundwater storage The final remaining water (if any) goes to interflow storage, where it will be routed to the surface water network.

GR4J Infiltration Method (`INF_GR4J`)

From the GR4J model Perrin et al. (2003):

$$M_{inf} = \phi_{max} \cdot \left(\frac{\alpha \cdot (1 - \left(\frac{\phi_{soil}}{\phi_{max}}\right)^2}{1 + \alpha\phi_{soil}}\phi_{max}\right) \tag{4.19}$$

where $\alpha = \tanh(\phi_{pond}/\phi_{max})$, ϕ_{pond} [mm] is the ponded water storage, ϕ_{soil} is the top soil layer water content [mm], and ϕ_{max} is the maximum topsoil storage [mm].

4.3 Baseflow

Baseflow refers to the flow of water from an aquifer or deeper soil horizon to surface water, typically due to a head gradient between fully saturated soil and stream. It may be considered the sum of the contribution of deep groundwater exchange with a river and delayed storage in the streambank.

4.3.1 Sources/Sinks

Baseflow moves water from either `SOIL[m]` or `AQUIFER` state variables, depending upon the soil structure model specified by the `:SoilModel` command. The water is always moved to `SURFACE_WATER`.

4.3.2 Constraints/Notes

Baseflow is rate-limited by the availability of soil/aquifer storage.

4.3.3 Available Algorithms

Constant Baseflow (`BASE_CONSTANT`)

A constant, specified rate of baseflow:

$$M_{base} = M_{max} \tag{4.20}$$

where M_{max} [mm/d] is the maximum baseflow rate, soil parameter `MAX_BASEFLOW_RATE`.

Linear Storage (`BASE_LINEAR_STORAGE` or `BASE_LINEAR_ANALYTIC`)

A very common approach used in a variety of conceptual models. The baseflow rate is linearly proportional to storage:

$$M_{base} = k\phi_{soil} \tag{4.21}$$

Where k [1/d] is the baseflow coefficient (soil parameter `BASEFLOW_COEFF`), and ϕ_{soil} is the water storage [mm] in the soil or aquifer layer. An alternate version, `BASE_LINEAR_ANALYTIC` may be used to simulate the same condition, except using a closed-form expression for integrated flux over the timestep:

$$M_{base} = \phi_{soil} \cdot (1 - \exp(-k\Delta t))/\Delta t \tag{4.22}$$

The two methods are effectively equivalent for sufficiently small timesteps, but the second is preferred for large values of k.

Non-Linear Storage (`BASE_POWER_LAW`)

A very common approach used in a variety of conceptual models, including HBV Bergstrom (1995). The baseflow rate is non-linearly proportional to storage:

$$M_{base} = k\phi_{soil}^{n} \tag{4.23}$$

Where k [1/d] is the baseflow coefficient (soil parameter `BASEFLOW_COEFF`), and ϕ_{soil} is the water storage [mm] in the soil or aquifer layer, and n is the user-specified soil parameter `BASEFLOW_N`.

VIC Baseflow Method (`BASE_VIC`)

From the VIC model Wood et al. (1992) as interpreted by (Clark et al., 2008):

$$M_{base} = M_{max} \left(\frac{\phi_{soil}}{\phi_{max}} \right)^{n} \tag{4.24}$$

where M_{max} [mm/d] is the maximum baseflow rate at saturation (soil parameter `MAX_BASEFLOW_RATE`), ϕ_{soil} is the water storage [mm] in the soil or aquifer layer, ϕ_{max} is the maximum soil storage capacity , and n is the user-specified soil parameter `BASEFLOW_N`.

GR4J Baseflow Method (`BASE_GR4J`)

From the GR4J model Perrin et al. (2003):

$$M_{base} = \frac{\phi_{soil}}{\Delta t} \cdot \left(1 - \left(\left(\frac{\phi_{soil}}{\phi_{ref}} \right)^{4} \right)^{\frac{1}{4}} \right) \tag{4.25}$$

where ϕ_{ref} [mm] is the reference soil storage, the user-specified soil parameter `GR4J_X3`, ϕ_{soil} is the water storage [mm] in the soil or aquifer layer..

Threshold-based Baseflow Method (`BASE_THRESH_POWER`)

Here, baseflow doesn't commence until a threshold saturation of the soil layer is met. Above the threshold, the outflow rate is controlled by saturation up to a maximum rate.

$$M_{base} = M_{max} \cdot \left(\frac{\frac{\phi_{soil}}{\phi_{max}} - S_{th}}{1 - S_{th}} \right)^{n} \tag{4.26}$$

where S_{th} [-] is the threshold saturation at which baseflow begins (soil parameter `BASEFLOW_THRESH`, M_{max} is the soil parameter `MAX_BASEFLOW_RATE` [mm/d], and the power law coefficient n is the soil parameter `BASEFLOW_N`.

4.4 Percolation

Percolation refers to the net downward flow of water from one soil/aquifer unit to another. This process is physically driven by a moisture gradient, but this is often simplified in conceptual percolation models.

4.4.1 Sources/Sinks

Percolation moves water between `SOIL[m]` or `AQUIFER` units, depending upon the soil structure model specified by the `:SoilModel` command. The user typically has to specify both the 'from' and 'to' storage compartments.

4.4.2 Constraints/Notes

Percolation is rate-limited by the availability of soil/aquifer storage and by the capacity of the receptor 'to' compartment.

4.4.3 Available Algorithms

Constant Percolation (`PERC_CONSTANT`)

A constant, specified rate of percolation from one soil layer to the next:

$$M_{perc} = M_{max} \tag{4.27}$$

where M_{max} is the soil parameter `MAX_PERC_RATE` of the 'from' soil compartment.

Constant Percolation (`PERC_GAWSER`)

As used in the GAWSER hydrological model, (Schroeter, 1989).

$$M_{perc} = M_{max} \left(\frac{\phi_{soil} - \phi_{fc}}{\phi_{max} - \phi_{fc}} \right) \tag{4.28}$$

where M_{max} is the soil parameter `MAX_PERC_RATE` and the moisture contents are defined in equation 4.1. All parameters refer to that of the 'from' soil compartment.

Power Law Percolation (`PERC_POWER_LAW`)

Percolation is proportional to soil saturation to a power:

$$M_{perc} = M_{max} \left(\frac{\phi_{soil}}{\phi_{max}} \right)^{n} \tag{4.29}$$

where M_{max} is the soil parameter `MAX_PERC_RATE`, n is the soil parameter `PERC_N` and ϕ_{soil} and ϕ_{max} are defined in equation 4.1. All parameters refer to that of the 'from' soil compartment.

Power Law Percolation (PERC_PRMS)

Percolation is proportional to drainable soil saturation to a power, as done in the PRMS model (Leavesley and Stannard, 1995):

$$M_{perc} = M_{max} \left(\frac{\phi_{soil} - \phi_{tens}}{\phi_{max} - \phi_{tens}} \right)^n \tag{4.30}$$

where M_{max} is the soil parameter `MAX_PERC_RATE`, n is the soil parameter `PERC_N` and ϕ_{soil}, ϕ_{tens}, and ϕ_{max} are defined in equation 4.1. All parameters refer to that of the 'from' soil compartment.

Sacramento Model Percolation (PERC_SACRAMENTO)

Percolation is proportional to drainable soil saturation to a power, as done in the PRMS model (Leavesley and Stannard, 1995):

$$M_{perc} = M_{max}^{base} \left(1 + \alpha \left(1 - \frac{\phi_{soil}^{to}}{\phi_{max}^{to}} \right)^{\psi} \right) \left(\frac{\phi_{soil} - \phi_{tens}}{\phi_{max} - \phi_{tens}} \right) \tag{4.31}$$

where M_{max}^{base} is the saturated baseflow rate (soil parameter `MAX_BASEFLOW_RATE`), and ϕ_{soil} and ϕ_{max} are defined in equation 4.1. All parameters refer to that of the 'from' soil compartment, unless they have the ^{to} superscript.

GR4J Model Percolation (PERC_GR4JEXCH and PERC_GR4JEXCH2)

Percolation (really here exchange between a conceptual soil store and a groundwater store) is calculated as consistent with the original GR4J model (**?**):

$$M_{perc} = -x2 * pow(min(\phi_{soil}/x3, 1.0), 3.5) \tag{4.32}$$

where $x2$ is the soil parameter `GR4J_X2` and $x3$ is the soil parameter `GR4J_X3` (both properties of the soil from which the water is percolating). In the case of `PERC_GR4JEXCH2`, the soil water content ϕ_{soil} refers to the topsoil storage (in SOIL[0]) rather than the soil from which percolation is being taken.

4.5 Interflow

Interflow refers to subsurface flow moving laterally through a shallow unsaturated soil horizon until it enters a stream channel.

4.5.1 Sources/Sinks

Interflow moves water between SOIL and SURFACE_WATER units, and is typically used in conjunction with a (slower) baseflow algorithm. The user typically has to specify the 'from' storage compartment (i.e. a specific soil layer); the 'to' storage compartment is always SURFACE_WATER.

4.5.2 Constraints/Notes

Interflow is rate-limited by the availability of soil/aquifer storage.

4.5.3 Available Algorithms

PRMS model Percolation (INTERFLOW_PRMS)

Interflow is proportional to drainable soil saturation to a power, as done in the PRMS model (Leavesley and Stannard, 1995):

$$M_{inter} = M_{max} \cdot \left(\frac{\phi_{soil} - \phi_{tens}}{\phi_{max} - \phi_{tens}} \right) \tag{4.33}$$

where M_{max} is the maximum interflow rate (soil parameter MAX_INTERFLOW_RATE), and ϕ_{soil}, ϕ_{tens}, and ϕ_{max} are defined in equation 4.1. All parameters refer to that of the 'from' soil compartment.

4.6 Soil Evaporation

Soil evaporation involves converting water from the soil layers to water vapour in the atmosphere. The rate of evaporation depends on soil moisture, plant type, stage of plant development and weather conditions such as solar radiation, wind speed, humidity and temperature.

4.6.1 Sources/Sinks

Soil evaporation always moves water between `SOIL[m]` and `ATMOSPHERE` units. Which soil layers are subjected to evaporation depend on the soil structure model specified by the `:SoilModel` command and the particular evaporation algorithm.

4.6.2 Constraints/Notes

Soil evaporation is rate-limited by the availability of soil/aquifer storage and by the capacity of the atmosphere to absorb water vapour.

4.6.3 Available Algorithms

VIC Soil Evaporation Algorithm (`SOILEVAP_VIC`)

Soil ET is proportional to the topsoil saturation to a power, as done in the VIC model (Wood et al., 1992):

$$M_{evap} = \text{PET} \cdot \left(1 - \left(1 - \frac{\phi_{soil}}{\phi_{max}}\right)^{\gamma}\right) \tag{4.34}$$

where PET is the potential evapotranspiration rate, γ is the soil parameter `VIC_EVAP_GAMMA`, and ϕ_{soil}, and ϕ_{max} are defined in equation 4.1.

Linear Evaporation (`SOILEVAP_HBV` or `SOILEVAP_TOPMODEL`)

Soil ET is at PET if storage exceeds the tension storage, then is linearly proportional to the soil saturation:

$$M_{evap} = \text{PET} \cdot \min\left(\frac{\phi_{soil}}{\phi_{tens}}, 1\right) \tag{4.35}$$

where PET is the potential evapotranspiration rate [mmd^{-1}], and ϕ_{soil} [mm] and ϕ_{tens} [mm] are defined in equation 4.1. The HBV model uses an additional snow correction, such that ET is zero in non-forested areas if snow depth is non-zero.

Root-distributed 2-layer Evaporation (`SOILEVAP_ROOT`)

Soil ET [mmd^{-1}] is linearly proportional to the soil saturation, but distributed by root fraction, ξ_m. Soil ET is at $\xi_m\cdot$ PET if storage exceeds the tension storage.

$$M^U_{evap} = \text{PET} \cdot \xi_U \cdot \min\left(\tfrac{\phi^U_{soil}}{\phi^U_{tens}}, 1\right) \tag{4.36}$$

$$M^L_{evap} = \text{PET} \cdot \xi_L \cdot \min\left(\tfrac{\phi^L_{soil}}{\phi^L_{tens}}, 1\right) \tag{4.37}$$

where U and L refer to the upper and lower layers, respectively, and ϕ_{soil} [mm] and ϕ_{tens} [mm] are defined in equation 4.1. Currently, ξ_L and ξ_U are hardcoded as 0.3 and 0.7, respectively.

Sequential 2-layer Evaporation (`SOILEVAP_SEQUEN`)

Daily soil ET [mmd^{-1}] is linearly proportional to the soil saturation; the top layer storage is exhausted first, then ET can be withdrawn from the lower layer.

$$M^U_{evap} = \text{PET} \cdot \left(\frac{\phi^U_{soil}}{\phi^U_{tens}}\right) \tag{4.38}$$

$$M^L_{evap} = (\text{PET} - M^U_{evap}) \cdot \left(\frac{\phi^L_{soil}}{\phi^L_{tens}}\right) \tag{4.39}$$

where U and L refer to the upper and lower layers, respectively, and ϕ_{soil} [mm] and ϕ_{tens} [mm] are defined in equation 4.1.

UBC Watershed Model Approach (`SOILEVAP_UBC`)

Evaporation is controlled by the soil moisture deficit, $\phi_{max} - \phi_{soil}$, where ϕ_{max} is defined in equation 4.1, and is corrected for effective saturated area.

$$M_{evap} = \text{PET} \cdot (1 - \beta_{fast})10^{\left(-\frac{\phi_{max}-\phi_{soil}}{\gamma_e}\right)} \tag{4.40}$$

where γ_e is the soil parameter `UBC_EVAP_SOIL_DEF` (the soil deficit at which the actual ET depletes to 0.1 PET), and β_{fast}, a proxy for the effective impermeable fraction is calculated as

$$\beta_{fast} = F_{imp} \cdot 10^{\left(-\frac{\phi_{max}-\phi_{soil}}{\gamma_a}\right)} \tag{4.41}$$

where F_{imp} is the impermeable fraction (land use parameter `IMPERMEABLE_FRAC`) and γ_a is the soil parameter `UBC_INFIL_SOIL_DEF`.

GR4J Soil Evaporation Method (`SOILEVAP_GR4J`)

From the GR4J model Perrin et al. (2003):

$$M_{evap} = \alpha\phi_{soil}\frac{2.0 - \frac{\phi_{soil}}{\phi_{max}}}{1.0 + \alpha\left(1.0 - \frac{\phi_{soil}}{\phi_{max}}\right)} \tag{4.42}$$

where $\alpha = \tanh(\text{PET}'/\phi_{max})$, PET$'$ is the PET remaining after ponded water storage is depleted, ϕ_{soil} is the water storage [mm] in the topsoil, ϕ_{max} is the maximum storage in the top soil.

4.7 Capillary Rise

Capillary rise is the rise of groundwater above the water table due to surface tension. The capillary zone extends up from the water table to the limit of capillary rise, and varies based on pore size and surface tension. In conceptual watershed models, the capillary rise term often refers to a process that moves water from lower to higher soil water stores.

4.7.1 Sources/Sinks

Capillary rise occurs between `SOIL` and `AQUIFER` units, depending upon the soil structure model specified by the `:SoilModel` command. The user typically has to specify the 'to' and 'from' storage compartments.

4.7.2 Constraints/Notes

Capillary rise is rate-limited by the availability of soil/aquifer storage and by the capacity of the receptor 'to' compartment.

4.7.3 Available Algorithms

HBV model Capillary Rise (`CRISE_HBV`)

Capillary rise rate is linearly proportional to soil saturation of the recipient soil, as done in the HBV model (Bergstrom, 1995):

$$M_{crise} = M_{max}^{cr}\left(1 - \frac{\phi_{soil}}{\phi_{max}}\right) \tag{4.43}$$

where M_{max}^{cr} is the maximum interflow rate (soil parameter `MAX_CAP_RISE_RATE`), and ϕ_{soil} and ϕ_{max} are defined in equation 4.1. All parameters refer to that of the 'to' soil compartment.

4.8 Canopy Evaporation

Canopy evaporation converts water from the vegetated canopy to water vapour in the atmosphere. The rate of evaporation depends on plant type, stage of plant development and weather conditions such as solar radiation, wind speed, humidity and temperature.

4.8.1 Sources/Sinks

Canopy evaporation always occurs between `CANOPY` and `ATMOSPHERE` units.

4.8.2 Constraints/Notes

Canopy evaporation is rate-limited by the availability of canopy storage.

4.8.3 Available Algorithms

Maximum Canopy Evaporation (`CANEVAP_MAXIMUM`)

Moisture on the canopy evaporates at the potential ET rate, provided storage is available.

$$M_{evap} = \text{PET} \cdot F_c \tag{4.44}$$

where PET is the potential evapotranspiration rate, F_c is the forest cover of the HRU (land use parameter `FOREST_COVERAGE`, and F_{sparse} is the vegetation sparseness factor (land use parameter `FOREST_SPARSENESS`.

Complete Canopy Evaporation (`CANEVAP_ALL`)

All moisture on the canopy evaporates instantaneously, i.e., all intercepted precipitation is sent back to the atmosphere.

Rutter Canopy Evaporation (`CANEVAP_RUTTER`)

From (Rutter et al., 1971):

$$M_{evap} = \text{PET} \cdot F_c \cdot (1 - F_t) \left(\frac{\phi_{can}}{\phi_{cap}} \right) \tag{4.45}$$

where PET is the potential evapotranspiration rate, F_c is the forest cover of the HRU (land use parameter `FOREST_COVERAGE`), F_t is the trunk fraction (vegetation parameter `TRUNK_FRACTION`), ϕ_{can} [mm] is the storage in the canopy over the forested region, ϕ_{cap} [mm] is the storage capacity of the canopy over the forested region.

4.9 Canopy Drip

Canopy drip is the loss of water from canopy to land surface.

4.9.1 Sources/Sinks

Canopy drip always occurs between `CANOPY` and `PONDED_WATER` units.

4.9.2 Constraints/Notes

Canopy drip is rate-limited by the availability of canopy storage.

4.9.3 Available Algorithms

Rutter Canopy Evaporation (`CANDRIP_RUTTER`)

Moisture on the canopy which exceeds storage falls instantaneously to the ground.

Slowdrain Canopy Evaporation (`CANDRIP_SLOWDRAIN`)

Moisture on the canopy which exceeds storage falls instantaneously to the ground, but the remaining drip is proportional to storage:

$$M_{drip} = \alpha \cdot \left(\frac{\phi_{can}}{\phi_{cap}} \right) \tag{4.46}$$

where α is the vegetation parameter `DRIP_PROPORTION`, and ϕ_{can} [mm] and ϕ_{cap} [mm] are the canopy storage and capacity in the forested region, respectively. Drip only occurs in the forested region.

4.10 Abstraction

Abstraction refers to the redirection of rainfall to surface impoundments, such as swales, ponds, and puddles. In Raven, these are collectively referred to as `DEPRESSION` storage.

4.10.1 Sources/Sinks

Abstraction always moves water from the `PONDED_WATER` state variable to the `DEPRESSION` storage state variable.

4.10.2 Constraints/Notes

none

4.10.3 Available Algorithms

SCS Method (`ABST_SCS`)

The abstraction rate is determined from the Soil Conservation Service method based upon SCS curve number.

$$M_{abst} = \frac{1}{\Delta t} \max\left(f_{SCS} \cdot 25.4 \left(\frac{1000}{\text{CN}} - 10\right), \phi_{pond}\right)$$

Where CN is the curve number corrected for antecedent precipitation conditions, where the type II (moderate wetness) curve number is given by the land use parameter `SCS_CN`. The fraction f_{SCS} is the land use parameter `SCS_IA_FRACTION`, and is 0.2 for the standard SCS approach (i.e., $I_a = 0.2S$)

Percentage Method (`ABST_PERCENTAGE`)

The abstraction rate is a given fraction of the ponded water accumulation rate,

$$M_{abst} = \alpha M_{pond}$$

where α is the land use parameter `ABST_PERCENT`

Fill Method (`ABST_FILL`)

In this approach, all ponded water (the cumulative contribution of rainfall and snowmelt) is redirected to depression storage until it is filled, then the remainder is available for infiltration/runoff.

The maximum depression storage amount is given by land use parameter `DEP_MAX`

4.11 Depression Storage Overflow

Depression overflow refers to water lost from ponds and wetlands to the main surface water network.

4.11.1 Sources/Sinks

Depression overflow moves water from the `DEPRESSION` storage variable and is always moved to `SURFACE_WATER`.

4.11.2 Constraints/Notes

Depression overflow is rate-limited by the availability of water in depression storage.

4.11.3 Available Algorithms

Power-law threshold (`DFLOW_THRESHPOW`)

The overflow to surface water is controlled by the amount of water in depression storage past a certain threshold:

$$M_{dflow} = M_{max} \cdot \left(\frac{\phi_{dep} - \phi_{th}}{\phi_{max} - \phi_{th}} \right)^n \tag{4.47}$$

where M_{max} [mm/d] is the maximum overflow rate, landuse parameter `DEP_MAX_FLOW`, ϕ_{dep} is the current depression storage [mm], ϕ_{th} is the given threshold storage level [mm] (landuse parameter `DEP_THRESHHOLD`, ϕ_{max} is the maximum depression storage `DEP_MAX` [mm], and n is the landuse parameter `DEP_N` (unitless).

4.12 Snow Balance

Snow balance algorithms are used to simulate the strongly coupled mass and energy balance equations controlling melting and refreezing of snow pack and the liquid phase in the snow pores.

4.12.1 Sources/Sinks

Most snow balance algorithms consists of multiple coupled equations, and there are also many 'to' and 'from' compartments, depending on which algorithm is selected. 'From' compartments include `SNOW` (as SWE), `SNOW_LIQ` and `SNOW_DEPTH`. 'To' compartments include `SNOW`, `ATMOSPHERE`, `SNOW_LIQ`, `SNOW_DEPTH` and `SURFACE_WATER`.

4.12.2 Constraints/Notes

Snow balance is rate-limited by the storage in 'from' and 'to' compartments.

4.12.3 Available Algorithms

Simple Melt (`SNOBAL_SIMPLE_MELT`)

The melt rate (in [mm/d]) is simply calculated by applying the potential melt rate to the snowpack until it is gone.

$$M_{melt} = M'_{melt} \tag{4.48}$$

where M'_{melt} [mm/d] is calculated using one of the methods described in section **??**. This is the same as using `:SnowMelt MELT_POTENTIAL`.

HBV Snow balance (`SNOBAL_HBV`)

Potential melt and refreeze rates are calculating using a degree day method, with the melt factor M_a corrected for forest cover and aspect. Meltwater fills the snow porespace first, then is allowed to overflow. (Bergstrom, 1995)

$$\begin{aligned} M_{melt} &= M_a \cdot \max(T - T_f, 0) \\ M_{refreeze} &= K_a \cdot \max(T_f - T, 0) \end{aligned} \tag{4.49}$$

where K_a is the land use parameter `REFREEZE_FACTOR` [mm/d/ °C], M_a is the land use parameter `MELT_FACTOR` [mm/d/ °C], which is corrected seasonally using the land use parameters `MIN_MELT_FACTOR`, `HBV_MELT_ASP_CORR` and `HBV_MELT_FOR_CORR`.

UBCWM Snow balance (`SNOBAL_UBCWM`)

As described in the UBC Watershed model documentation (Quick, 1995). Potential melt is typically calculated using the `POTMELT_UBC` method described in section 6.8.1. If the land use/land type parameter `SNOWPATCH_LIMIT` is zero, the method is relatively straightforward - SWE is melted at a rate equivalent to the potential melt, with some of the water melted first filling up the Liquid holding capacity of the snow, the remainder becoming ponded water. During melt of ripened snowpack, the liquid water is released along with the corresponding SWE melted. The user is referred to the UBCWM documentation for the full description of the snowmelt algorithm with snow patching.

Cema Niege Snow balance (`SNOBAL_CEMA_NIEGE`)

Often used with the GR4J model configuration, the Cema Niege snow balance uses the potential melt rate calculated using the methods of section 6.8.1, but corrected with a snow cover factor,

$$M_{melt} = \left(0.1 + 0.9 \cdot \min\left(\frac{\phi_{SWE}}{S_{Ann}}, 1\right)\right) \cdot M' \tag{4.50}$$

where M' is the potential melt rate, ϕ_{SWE} is the snow amount as snow water equivalent, S_{Ann} is the average annual snow amount, specified as the global parameter `AVERAGE_ANNUAL_SNOW`.

Two-layer Snow balance (`SNOBAL_TWOLAYER`)

[documentation required]

4.13 Snow Sublimation

Sublimation is the process of snow transforming to water vapour without passing through the intermediate liquid phase. It can be a significant part of the snow balance at high elevations, windy regions, and when atmospheric water conent is low.

4.13.1 Sources/Sinks

Sublimation always occurs between `SNOW` and `ATMOSPHERE` units.

4.13.2 Constraints/Notes

Sublimation is limited by the availability of snow.

4.13.3 Available Algorithms

Kuzmin (1957) method (`SUBLIM_KUZMIN`)

The sublimation rate (in [mm/d]) is calculated using the following empirical relationship Kuzmin (1957):

$$M_{subl} = 0.18 + 0.098 \cdot v_{ave} \cdot (P_{sat} - P_{vap}) \tag{4.51}$$

where v_{ave} [m/s] is the wind velocity at 10m, P_{sat} and P_{ave} [mb] are the saturated vapour pressure and vapour pressure, respectively.

Central Sierra method (`SUBLIM_CENTRAL_SIERRA`)

The sublimation rate (in [mm/d]) is calculated using the following empirical relationship U.S. Dept. of Commerce (1956):

$$M_{subl} = 0.0063 \cdot (h_w \cdot h_v)^{-\frac{1}{6}} \cdot (P_{sat} - P_{vap}) \cdot v_{ave} \tag{4.52}$$

where v_{ave} [m/s] is the wind velocity at reference height h_w [ft], P_{sat} and P_{ave} [mb] are the saturated vapour pressure and vapour pressure, respectively, and h_v is the elevation of the vapour pressure reference height [ft].

4.14 Snow Melt

Snow melt algorithms are used if the full `:SnowBalance` algorithms are not applied, and simply convert `SNOW` to `SNOW_LIQ` or `PONDED_WATER`

4.14.1 Sources/Sinks

Snow melt always occurs between `SNOW` and a user-specified target unit, typically `SNOW_LIQ` (with a cascade), `PONDED_WATER`, or `SURFACE_WATER`—

4.14.2 Constraints/Notes

Snow melt is limited by the availability of snow in the snowpack. Melt rates must be positive.

4.14.3 Available Algorithms

Potential Melt (`MELT_POTENTIAL`)

The melt rate (in [mm/d]) is simply calculated by applying the potential melt rate to the snowpack until it is gone.

$$M_{melt} = M'_{melt} \tag{4.53}$$

where M'_{melt} [mm/d] is calculated using one of the methods described in section **??**. This is the same as using `:SnowBalance SNOBAL_SIMPLE`.

4.15 Snow Refreeze

Snow refreeze algorithms are used if the full `:SnowBalance` algorithms are not applied, and simply convert `SNOW_LIQ` to `SNOW`

4.15.1 Sources/Sinks

Snow refreeze always occurs between `SNOW_LIQ` and `SNOW` units.

4.15.2 Constraints/Notes

Snow refreeze is limited by the availability of liquid water in the snowpack. Refreeze rates must be positive

4.15.3 Available Algorithms

Degree Day Approach (`FREEZE_DEGREE_DAY`)

The refreeze rate (in [mm/d]) is calculated using the following degree-day relationship (much like the degree-day melt approaches for calculating potential melt):

$$M_{frz} = K_f \cdot \min(T_f - T_a, 0) \tag{4.54}$$

where K_f [mm/d/ °C] is the refreeze parameter (land use parameter `REFREEZE_FACTOR`, T_f is the freezing temperature (0 °C) and T_a is the air temperature.

4.16 Snow Albedo Evolution

Snow albedo evolution is the process through which snow albedo changes due to snow compaction, snowpack aging, or fresh snow accumulation.

4.16.1 Sources/Sinks

The snow albedo evolution algorithms have no sources or sinks, it simply models the rate of change of albedo over time.

4.16.2 Constraints/Notes

Snow albedo is constrained to be in the range 0-1.

4.16.3 Available Algorithms

UBC Watershed Model Approach (`SNOALB_UBC`)

The albedo, α, increases with accumulating snow and decreases as the season progresses. It is bounded by the global parameters `MIN_SNOW_ALBEDO` `MAX_SNOW_ALBEDO`, defined in the `:UBCSnowParams` command in the .rvp file.

$$M_{snalb} = -\alpha \cdot \frac{1-K}{\Delta t} + \frac{(\alpha_{max} - \alpha)}{\Delta t} \min\left(\frac{SN}{SN_{alb}}, 1\right) \qquad \text{if } \alpha > \alpha_b \quad (4.55)$$

$$M_{snalb} = -\alpha_b \exp\left(-\frac{S_{cum}}{S_{max}}\right) \frac{dS_{cum}}{dt} + \frac{(\alpha_{max} - \alpha)}{\Delta t} \min\left(\frac{SN}{SN_{alb}}, 1\right) \qquad \text{if } \alpha < \alpha_b \quad (4.56)$$

where α_{max} is the global parameter `MAX_SNOW_ALBEDO`, α_b is a threshold albedo value (`ALBASE`), SN [mm/d] is the daily snowfall, SN_{alb} [mm/d] is the total daily snowfall required to bring albedo to that of new snow (global param `ALBSNW`), K is the global parameter `ALBREC` (a recession constant), S_{cum} is the cumulative snow deposited in the current winter season and S_{max} is an estimate of the maximum cumulative snowfall in a year (`MAX_CUM_MELT`). All of these global parameters are specified using the command `:UBCSnowParams` in the .rvp file.

4.17 Glacial Melt

Glacial melt refers to the process of melting of glacier ice. It is typically only applied to those HRUs treated as glaciers.

4.17.1 Sources/Sinks

Glacial melt algorithms move water from `GLACIER_ICE` to either `GLACIER` (liquid water storage in or on the glacier itself) or `SURFACE_WATER`. They may also modify the cold content of the glacier, `GLACIER_CC`.

4.17.2 Constraints/Notes

Glacial melt is not limited by the available glacier ice, which is assumed to be abundant.

4.17.3 Available Algorithms

Simple Melt Approach (`GMELT_SIMPLE_MELT`)

The melt rate is equal to the potential melt rate, calculated using the methods described in section 6.8.1.

HBV Approach (`GMELT_SIMPLE_MELT`)

> The melt rate is equal to the potential melt rate, calculated using the methods described in section 6.8.1. A glacial melt correction factor may be used to modify the melt rate (land use parameter `HBV_MELT_GLACIER_CORR`), which is 1 by default. No glacial melt occurs if there is any snow cover, i.e., the snow must melt first.

UBC Watershed Model Approach (`GMELT_UBC`)

> The potential melt rate is applied to melt the glacier, but modified by the snow cover (i.e., no glacial melt occurs if there is 100% snow cover).

4.18 Glacier Release

Glacial release refers to the release of meltwater from the glacier to surface water.

4.18.1 Sources/Sinks

Glacial release algorithms move water from `GLACIER` to `SURFACE_WATER`.

4.18.2 Constraints/Notes

Glacial release is limited by the available glacier liquid water storage.

4.18.3 Available Algorithms

Linear Storage (`GRELEASE_LINEAR_STORAGE`)

A simple linear storage coefficient approach:

$$M_{grelease} = -K\phi_{glac}$$

where ϕ_{glac} [mm] is the total glacial storage, and K [1/d] is a linear storage coefficient (land use parameter `GLAC_STORAGE_COEFF`)

Linear Storage (Analytical) (`GRELEASE_LINEAR_ANALYTIC`)

A simple linear storage coefficient approach, but analytically solved for and integrated over the timestep:

$$M_{grelease} = \frac{\phi_{glac}}{\Delta t}(1 - exp(-K\Delta t))$$

where ϕ_{glac} [mm] is the total glacial storage,Δt is the model time step and K [1/d] is a linear storage coefficient (land use parameter `GLAC_STORAGE_COEFF`)

HBV-EC approach (`GRELEASE_HBV_EC`)

A simple linear storage coefficient approach:

$$M_{grelease} = -K^{*}\phi_{glac}$$

where ϕ_{glac} [mm] is the total glacial storage, and K^{*} [1/d] is a linear storage coefficient which is corrected for snow cover, such that the glacier releases more water at times of less snow cover, calculated as:

$$K^{*} = K_{min} + (K - K_{min})\exp(-AG(SN + SN_{liq}))$$

where K_{min} [1/d] is a linear storage coefficient (land use parameter `HBV_GLACIER_KMIN`), K [1/d] is a linear storage coefficient (land use parameter `GLAC_STORAGE_COEFF`), AG [1/mm] is the land use parameter `HBV_GLACIER_AG`, and SN and SN_{liq} [mm] are the SWE and liquid snow content of the snowpack on top of the glacier, respectively.

4.19 Crop Heat Unit Evolution

Crop heat units (CHUs) are used by some municipalities in Ontario, Canada in order to assess soil evaporation. ET is maximized when CHUs meet their maturity level. To be used in conjunction with the soil evaporation algorithm `SOILEVAP_CHU`. The crop heat units grow in magnitude over the course of a growing season based upon the daily temperature profiles.

4.19.1 Sources/Sinks

Crop heat unit evolution algorithm does not move water between storage compartments. The method only revises the magnitude of the `CROP_HEAT_UNITS` state variable.

4.19.2 Constraints/Notes

Crop heat units are zero outside of the growing season.

4.19.3 Available Algorithms

Ontario method (`CHU_ONTARIO`)

The growing season is determined to begin when the minimum temperature over a 3-day period is 12.8 °C, at which time the crop heat units are set to zero. It ends when the temperature dips below -2 °Cor after September 30th. During the growing season, CHUs are incremented using the following expressions Brown and Bootsma (1993):

$$\begin{aligned} \text{CHU}_d &= 3.33 \cdot (T_{max} - 10) - 0.084 \cdot (T_{max} - 10)^2 \\ \text{CHU}_n &= 1.8 \cdot (T_{min} - 4.4) \end{aligned}$$

$$\text{CHU}_{new} = \text{CHU}_{old} + 0.5 \cdot (\text{CHU}_d + \text{CHU}_n)$$

where T_{min} and T_{max} are the minimum and maximum daily temperatures

4.20 Special Processes

The flush, split, and overflow processes are used in conceptual models to represent the 'instantaneous' movement of water from one water storage compartment to another. The convolution process allows for a time lag of storage. As these are wholly conceptual in nature, they are most often included in order to emulate the functioning of existing hydrologic models. These processes may not work as intended when using a numerical method other than the ordered series approach.

- The Flush process instantaneously moves all of the water storage from one storage to another.
- The Overflow process moves the excess water storage (more than the maximum capacity of the water storage unit) to another compartment.
- The Split process instantaneously moves all of the water storage from one storage compartment into two, with the proportion specified in the input command.
- The convolution process temporarily stores water in a convolution storage compartment, to be released using a transfer function approach. The output fluxes from a convolution process are typically an attenuated and delayed version of the input fluxes.

4.20.1 Sources/Sinks

The flush, overflow, and split processes may move water from any water storage compartment to any other. The convolution process (`:Convolve` command in the input) releases water added to a convolution storage structure buy any other process to any storage compartment.

4.20.2 Constraints/Notes

Since convolution methods store the time history of inputs to convolution storage of a duration consistent with the longest time delay in the convolution, it is not suggested to use convolution with a time constant in days with an hourly time step. Typically the order of the time delay should be on the order of the model time step.

4.20.3 Available Algorithms

The below convolution methods are available. All of them perform a discrete version of the following convolution:

$$M_{conv} = \int_0^\infty UH(\tau) I(t-\tau) d\tau$$

where $I(t)$ is the input flux history (in mm/d) to the convolution storage unit and $UH(t)$ is the transfer function; the area under the transfer function is always equal to one to ensure mass balance.

GR4J Transfer function 1 (CONVOL_GR4J_1)

The transfer function used is

$$UH(t) = \begin{cases} \frac{5}{2x_4}\left(\frac{t}{x_4}\right)^{\frac{3}{2}} & \text{for } t \leq x_4 \\ 0 & \text{for } t > x_4 \end{cases}$$

where x_4 is the land use parameter `GR4J_X4`.

GR4J Transfer function 1 (CONVOL_GR4J_2)

The transfer function used is

$$UH(t) = \begin{cases} \frac{5}{4x_4}\left(\frac{t}{x_4}\right)^{\frac{3}{2}} & \text{for } t \leq x_4 \\ \frac{5}{4x_4}\left(2-\frac{t}{x_4}\right)^{\frac{3}{2}} & \text{for } x_4 < t \leq 2x_4 \\ 0 & \text{for } t > 2x_4 \end{cases}$$

where x_4 is the land use parameter `GR4J_X4`.

To do (2)

Chapter 5

Routing

The following chapter outlines the routing algorithms available for modelling the downstream migration of water through a terrain/channel network in Raven. As briefly summarized in section 1.2.2, the routing process in Raven has two components: at the sub-basin level, rainfall and snowmelt from all HRUs is released to surface water via overland runoff, interflow, and base flow. There is some delay and/or redistribution of the timing of the release of this water to the sub-basin river reach, then again a delay before the water reaches the outlet. This delay is handled in Raven typically using a linear transfer function (e.g., Unit Hydrograph) approach, and is termed in-catchment routing. The second form of routing is the hydraulic/hydrologic routing between subbasins and within the main channel of each subbasin. This is referred to as in-channel routing. The distinction between the two is shown in figure 1.4. In addition to in-catchment and in-channel routing, a separate routine is used to route waters through reservoirs/lakes at the end of subbasins.

5.1 In-Catchment Routing

5.1.1 Overview

It is important to note that the rate of release of water from storage within an HRU is treated as constant over a given time step. This is the most appropriate, since water storage state variables are stored as snapshots in time (at the end of each time step). However, in the channel, the state variable is no longer storage, but flow rates, as is consistent with the majority of routing algorithms developed in the literature. Therefore, in addition to impacting the timing of the flows, in-catchment routing is used to map flow rates which are constant over a time step (losses from the HRU) to those which are varying linearly over a time step (in-channel flows).

In all cases, in catchment routing is treated using a discrete transfer function approach, i.e.,

$$Q(t+\Delta t) = \sum_{n=0}^{N} Q_{lat}(t-n\Delta t) \cdot UH_n \tag{5.1}$$

where $Q(t)$ [$\mathrm{m^3d^{-1}}$] is the flow rate into the channel from the subbasin, $Q_{lat}(t)$ [$\mathrm{m^3d^{-1}}$] is the lateral release flow rate from the HRU surface over the timestep from t to $t+\Delta t$, and $\vec{UH}$ is a unitless vector which describes the distribution of arrival times to the channel. The sum of values of the $\vec{UH}$ vector equal 1, and the magnitude if UH_n may be interpreted as the percentage of the flow appearing in the channel n time steps after its release from the HRU. This is the discrete generalization of a convolution:

$$Q(t) = \int_0^\infty Q_{lat}(t-\tau) \cdot UH(\tau) d\tau \tag{5.2}$$

Either of these may be interpreted as providing a distributed delay between when water is released from the HRU and when it appears in the channel.

5.1.2 Algorithms

The following algorithms may be used for in-catchment routing. The sole difference between the various catchment routing algorithms is the shape of the unit hydrograph used.

Dump Method (`ROUTE_DUMP`)

In the "dump" method of catchment routing, all of the water released from the HRUs to surface water over a time step appears in the channel at the end of the time step. This is generally valid for small subbasins (those with small times of concentration) or large time steps. This is equivalent to $\vec{UH} = \{1, 0, 0, 0, ...\}$, and is an approximation of

$$UH(t) = \delta(t)$$

where δ is the Dirac delta function.

Gamma Unit Hydrograph (`ROUTE_GAMMA_CONVOLUTION`)

Here, a Gamma distribution is used to represent the unit hydrograph, i.e.,

$$UH(t) = \frac{t^{a-1}}{t_p^a \Gamma(a)} \exp(-t/t_p)$$

where Γ is the Gamma function, t_p is the time to peak, specified as the subbasin property `TIME_TO_PEAK`. In Raven, a is fixed at $a = 3$.

Triangular Unit Hydrograph (`ROUTE_TRI_CONVOLUTION`)

A triangular unit hydrograph is used with a peak time of t_p, specified as the subbasin property `TIME_TO_PEAK` and total duration specified by the time of concentration, t_c, specified using the subbasin property `TIME_CONC`. Note that variations in the time of concentration smaller than the model time step will have no impact on model solution.

$$UH(t) = \begin{cases} \frac{2}{t_c}\frac{t}{t_p} & \text{for } t < t_p \\ \frac{2}{t_c}\left(\frac{t_c - t}{t_c - t_p}\right) & \text{for } t \geq t_p \end{cases}$$

Nash Unit Hydrograph (`ROUTE_RESERVOIR_SERIES`)

The Nash unit hydrograph is used with a linear reservoir constant (k) specified using the subbasin property `RES_CONSTANT` and the number of reservoirs (N) equal to `NUM_RESERVOIRS`.

$$UH(t) = t^{N-1} k^N e^{-kt}$$

5.2 In-Channel Routing

5.2.1 Overview

In Raven, in-channel routing is the only means by which water, mass, and energy are exchanged laterally between subbasins. It is assumed that this movement is unidirectional, i.e., water moves downstream only through a one-dimensional branching stream network fully described by the succession of subbasins defined in the .rvh file. Each subbasin can have a single outlet and is conceptualized as having a single primary channel running through it, which may or may not have a reservoir at the end of the channel. Headwater subbasins (those without an upstream subbasin) are assumed to have no corresponding channel, but may have a reservoir which is fed purely via in-catchment routing and releases water to the next downstream basin.

This routing formalization leads to some implicit guidelines for subbasin discretization.

- Subbasin outlets should typically occur at stream network junctions.
- Surface water reservoirs should be located at the outlet of a subbasin
- All stream gauges used for calibration or model evaluation should be located at the outlet of a subbasin
- For lumped (single subbasin) models, channel routing is usually disabled entirely.

In-Channel routing may be treated by a number of algorithms. However, as indicated in section 1.2.2, all of these algorithms may be generalized as

$$Q_{out}^{n+1} = F_{route}(Q_{out}^{n}, \vec{Q}^{in}, \vec{P}_s) \tag{5.3}$$

Where F_{route} is the routing algorithm, $\vec{Q}^{in}$ is the recent time history of upstream (and upbasin) inflows to the channel, $\vec{P}_s$ is a vector of channel parameters, typically a number of stored channel rating curves, primary channel and bank roughness, and weir or reservoir relationships. Figure 1.4 indicates the meaning of these major parameters. The descriptions of the channel inputs are detailed in section A.2.2 of the appendix, and specified using the `:ChannelProfile` command.

5.2.2 Algorithms

While more rigorous hydraulic routing algorithms (which handle backwater effects, etc.) may be implemented in future incarnations of Raven, for the most part, the algorithms currently in Raven are considered hydrologic routing methods based upon simple storage relationships, rather than complete solution of the Saint-Venant equations for momentum and mass conservation. They fall roughly into two categories: convolution approaches, which function in a manner identical to that of the unit hydrograph approach used for in-catchment routing, and mass-balance approaches, which solve for outflow through a discrete form of the mass balance equation. Both sets of approaches are mass-conservative.

As with the in-catchment methods, the convolution-based methods (`ROUTE_DIFFUSIVE_WAVE`) and (`ROUTE_PLUG_FLOW`), use a discrete transfer-function approach:

$$Q_{out}^{n+1} = \sum_{i=0}^{N} Q_{in}^{n-i+1} \cdot UH_i' \tag{5.4}$$

where Q_{out}^{n+1} [m^{3}d^{-1}] is the flow rate from the subbasin at the end of the time step, Q_{in}^{n} [m^{3}d^{-1}] is the inflow rate from upstream sources at the end of time step n, and $\vec{UH'}$ is a unitless vector which describes the distribution of arrival times to the channel. The sum of values of the $\vec{UH'}$ vector equal 1, and the magnitude if UH_i' may be interpreted as the percentage of the flow leaving from the channel i time steps after its arrival in the channel from upstream sources.

Many of the in-channel routing routines require the reference celerity for the channel reach:

$$c_{ref} = \left.\frac{dQ}{dA}\right|_{Q_{ref}} \tag{5.5}$$

c_{ref} is the reference celerity for the reach, the velocity corresponding to the reference flow, Q_{ref} [m^{3}d^{-1}] in the reach, usually specified as the bank full flow using the subbasin parameter `Q_REFERENCE`. The slope of the Q vs. A relationship at Q_{ref} is interpolated from that generated for the specific channel.

No Routing (`ROUTE_NONE`)

All inflows (both lateral and upstream), are instantly routed to the channel outlet, i.e.,

$$Q_{out}^{n+1} = Q_{in}^{n+1} + Q_{lat}^{n+1}$$

This option is mostly used for single subbasin models.

Simple Plug Flow (`ROUTE_PLUG_FLOW`)

Here, there is a delay between water entering and exiting the channel dictated by the celerity of the channel reach, but there is no smearing out of the hydrograph as it migrates along the channel.

$$UH'(t) = \delta\left(t - \frac{L}{c_{ref}}\right)$$

where $\delta(t)$ is the Dirac delta function, L is the reach length within the subbasin (specified from the subbasin property `REACH_LENGTH`, and c is the reference celerity of the channel, as determined from the channel profile characteristics and the subbasin's reference flow rate, Q_{ref} specified as the subbasin parameter `Q_REFERENCE`. The reference celerity c_{ref} is calculated using 5.5.

Diffusive Wave Model (`ROUTE_DIFFUSIVE_WAVE`)

Here, an analytical solution to the diffusive wave equation is used to smear out the flood wave as it propagates through the reach. As with the simple plug flow approach, the reference celerity is used to determine the mean travel time of the wave, and the channel diffusivity, D [m^{2}d^{-1}] controls the smearing out of the wave signal prior to exiting the reach.

$$UH'(t) = \frac{1}{2\sqrt{\pi D t}} \exp\left(-\frac{(L - c_{ref}t)^2}{4Dt}\right)$$

where L [m] is the channel reach length, c_{ref} is calculated using 5.5, and the channel diffusivity, D, is estimated from the channel reference flow Q_{ref} (subbasin parameter `Q_REFERENCE`) using the following relationship **?**:

$$D = \frac{Q_{ref}}{2S \cdot d(Q_{ref})}$$

where S is the channel bedslope and $d(Q)$ is the relationship between flow depth, d and flow rate, Q, in the channel, determined from the channel geometry.

Storage Coefficient Method (`ROUTE_STORAGE_COEFF`)

The storage coefficient method evaluates outflow using a discrete approximation of the water balance for the channel over the time step **?**:

$$Q_{out}^{n+1} = c_1 \cdot Q_{in}^{n+1} + c_2 \cdot Q_{in}^{n} + c_3 \cdot Q_{out}^{n} \tag{5.6}$$

here, the weights c_1, c_2, and c_3 are calculated from the storage coefficient, k, given as:

$$k = \min\left(\frac{1}{\frac{K}{\Delta t} + 0.5}, 1\right) \tag{5.7}$$

where K is the representative travel time for the reach (also the Muskingum K parameter, calculated as $\Delta x / c_{ref}$ where Δx is the reach segment length). Here, $c_1 = k/2$, $c_2 = k/2$, and $c_3 = 1 - k$.

Caution should be used with this method on long reaches without finely discretizing the reach, as water will arrive at the outlet immediately after entering, even with a large representative travel time in the reach.

Muskingum-Cunge Method (`ROUTE_MUSKINGUM`)

The standard Muskingum-Cunge approach also evaluates outflow using a discrete approximation of the water balance for the channel over the time step:

$$Q_{out}^{n+1} = c_1 \cdot Q_{in}^{n+1} + c_2 \cdot Q_{in}^{n} + c_3 \cdot Q_{out}^{n} \tag{5.8}$$

here, the weights c_1, c_2, and c_3 are calculated from the Muskingum X and K parameters as

$$\begin{aligned} c_1 &= \frac{\Delta t - 2KX}{2K(1-X) + \Delta t} \\ c_2 &= \frac{\Delta t + 2KX}{2K(1-X) + \Delta t} \\ c_3 &= \frac{-\Delta t + 2KX}{2K(1-X) + \Delta t} \end{aligned}$$

The Muskingum algorithm is well-documented in the literature. The Muskingum parameters X and K are calculated using the following relations:

$$K = \frac{\Delta x}{c_{ref}}$$

$$X = \frac{1}{2}\left(1 - \frac{Q_{ref}}{Sw_{ref}c_{ref}\Delta x}\right)$$

where c_{ref} is the reference celerity for the reach (calculated using equation 5.5), S is the channel bedslope, w_{ref} is the channel width at the reference flow Q_{ref} (basin parameter `Q_REFERENCE`, and Δx is the reach segment length (or reach length, L, if only one segment is used per reach). Care must be taken to ensure that X and K fall within a reasonable range of values, notably that $2KX < \Delta t < 2K(1-X)$. If the time step is too large, Raven automatically employs local time stepping. However, the case where the time step is too small (a warning will be thrown to RavenErrors.txt) must be handled via user intervention, by increasing the number of segments in the reach.

Iterative Hydrologic Routing Approach (`ROUTE_HYDROLOGIC`)

Here, the routing is performed using an iterative application of Newton's root-finding algorithm to the following discretization of the storage relationship for the reach,

$$\frac{V(Q_{out}^{n+1}) - V(Q_{out}^{n})}{\Delta t} = \frac{1}{2}(Q_{in}^{n} + Q_{in}^{n+1}) - \frac{1}{2}(Q_{out}^{n} + Q_{out}^{n+1})$$

Given that the channel volume, $V(Q)$ may be written as a function of outflow from the reach if a level-pool assumption is used, this may be expressed as a root-finding problem for Q_{out}^{n+1}. This method is very stable, fast, accurate, and mass-conserving. It avoids the numerical pitfalls of the non-iterative Muskingum algorithm. Right now, it can only be applied to reaches which constitute a single reach segment.

5.3 Reservoir Routing

5.3.1 Overview

Lakes or reservoirs may be specified using a `:Reservoir-:EndReservoir` command in the .rvh file (see appendix A.3), and are always located a the outlet of a subbasin, i.e., a reservoir linked to a given subbasin receives its water from that basin's in-channel routing routine, then releases it downstream.

Iterative Reservoir Routing Approach

Only one algorithmic option is available for routing water in a reservoir. In this approach, a Newton solver is used to iteratively calculate the reservoir stage using the following time discretization of the reservoir level-pool mass balance:

$$\frac{V(h^{n+1}) - V(h^{n})}{\Delta t} = \frac{1}{2}(Q_{in}^{n} + Q_{in}^{n+1}) - \frac{1}{2}\left(Q(h^{n}) + Q(h^{n+1})\right) - \frac{E}{2}\left(A(h^{n+1}) + A(h^{n})\right)$$

where h is the stage and $Q(h)$, $V(h)$, and $A(h)$ are the stage-discharge, stage-volume, and stage-area relations defined in the `:Reservoir` command; E is the open-water evaporation rate for the reservoir. Note that the reservoir should be included as an HRU with the average reservoir area. All precipitation falling on this HRU gets added to the Q_in component, where evaporation from the surface of the reservoir is only included in the above expression. If no HRU is linked to the reservoir in the reservoir command, evaporation is considered negligible and not included in the mass balance.

It is critical that the entire range of likely stage elevations are included when specifying the $Q(h)$ and $V(h)$ curves.

For large reservoirs (especially those with multiple subbasins draining into them), it is suggested to treat the reservoir as a single-HRU subbasin with zero reach length.

Known inflows or outflows from the reservoir may be considered in the above mass balance using the `:ReservoirExtraction` command in the .rvt file. These may also be used to override the reservoir outflow with known discharge rates (in this case $Q(h)$ must be set to zero for all h). If reservoirs are present in the model, the file *runname*_`ReservoirStages.csv` is automatically created.

Chapter 6

Forcing Functions

In Raven, forcing functions, such as rainfall or incident radiation, are calculated from meteorological information specified at gauge stations in the watershed or, alternatively, from gridded weather/climate data. This information is interpolated between gauge stations or grid cells to each hydrological unit (HRU), where it may be corrected for orographic or other effects. Forcing functions are calculated at the beginning of each computational time step, and are always constant over individual time steps.

Note that the basic data from which forcing functions are generated (often daily precipitation, minimum/maximum daily temperature, etc.) must be reported in terms of rates (e.g., mm/d or MJ/m^2/d) for precipitation and radiation data, not total quantities for the time period. For example, if hourly rainfall information is stored in mm, it must be converted to mm/d prior to simulation. Missing data in the gauge information is currently not allowed. The time periods of available forcing data must fully overlap the simulation duration, but they do not have to be identical.

The minimum required forcing data for fueling a Raven simulation is daily precipitation and daily maximum and minimum temperature. From this, Raven can partition precipitation into snowfall and rainfall, estimate subdaily temperatures and PET, and provide estimates of incoming shortwave and longwave radiation. Alternately, these parameters may be specified if available. Raven has the ability to estimate the following forcings from simple records of total precipitation and daily min/max temperatures:

- Snowfall/Rainfall
- Potential ET (or reference ET)
- Shortwave and longwave net radiation
- Cloud cover
- Potential melt
- Wind speed, relative humidity, and air pressure
- Orographic corrections to temperature, precip, and potential ET rates
- Sub-daily corrections to daily ET, SW radiation, and potential melt rates

Refer to section A.1.2 of Appendix A for more details about each of the available processes that will be discussed in this chapter.

6.1 Spatial Interpolation

Spatial interpolation of forcing functions from gauge stations to HRUs is based upon the lat-long locations of the gauges and HRUs as specified in the .rvt and .rvh files, respectively. These coordinates are converted into the most appropriate local Universal Transverse Mercator (UTM) coordinate system (as determined by RAVEN) to calculate distances between points. RAVEN currently supports nearest neighbor and inverse distance weighting interpolation, as documented under the `:Interpolation` command in appendix A.3. It also supports the provision of a user-specified gauge weighting file, such that gauges may be assigned specifically to individual HRUs or alternate interpolation schemes may be used external to the program.

In general, any interpolated field value (e.g., temperature), is calculated for each HRU using a relatively general weighted averaging scheme:

$$V_k = \sum_{g=1}^{NG} w_{kg} \cdot V_g$$

i.e., any value V_k for HRU k is generated by weighting the values from all gauges V_g, using an HRU-specific weighting factor w_{kg}. Note that $\sum_{g=1}^{NG} w_{kg} = 1$ is required. Different interpolation schemes differ only in the means by which they generate the weights, usually based upon the relative geographic position of the HRUs and gauges.

For gridded data, the contributions to each HRU to each grid cell is similarly specified using a weighting scheme, though in this case the most intuitive weighting scheme is to use an area-weighted average of the cells that overlap each HRU, i.e.,

$$V_k = \sum_{g=1}^{NC} w_{kg} \cdot V_g$$

where NC is the number of cells and $w_{kg} = (A_g \bigcap A_k)/A_k$, i.e., the weighting is determined by the area of intersection ($\bigcap$) between the grid cell (A_g) and HRU (A_k). The user must define these weights, typically using GIS.

6.2 Temperature

Daily average, sub-daily, and daily minimum and maximum temperatures are required for many hydrological simulation algorithms. This forcing data is often used for partitioning of precipitation into rainfall and snowfall components, estimating potential and actual evapotranspiration, driving snow melt and refreezing, as a proxy for cloud cover, etc., etc. In Raven, one of three temperature data sets are needed at each gauge or grid cell. Ideally, sub-daily (typically hourly) data is specified, and daily minimum, maximum, and average temperatures are easily calculated. If daily minimum and maximum temperature data are provided, daily averages are calculated as the average of the two, and sub-daily temperatures (if needed) are specified using the approach dictated by the `:TemperatureDownscaling` command. Lastly, if only daily average temperature is provided, the daily min, max, and sub-daily temperatures are also generated using the approach specified in the `:TemperatureDownscaling` command, but with the max and min calculated from the constant `:TemperatureSwing` parameter associated with each gauge.

6.2.1 Orographic Temperature Effects

Orographic effects may be applied to correct temperature estimates at each HRU based on the specified elevation of the HRU. The options available for orographic temperature adjustment are described below. The orographic temperature effect is set in the RVI file using the `:OroTempCorrect` keyword. Orographic corrections are typically only applied to gauged (not gridded) input data.

Simple Method (`OROCORR_SIMPLELAPSE`)

The simple method for orographic temperature correction estimates the HRU through the application of a lapse rate correction to the associated gauge temperature:

$$T = T_g - \alpha(z - z_g) \tag{6.1}$$

where T is the estimated HRU temperature, T_g is the measured gauge temperature, z and z_g are the elevation of the HRU and gauge respectively, and α is the specified adiabatic lapse rate. Equation 6.1 is applied to all temperature forcing variable time series, including: daily average, minimum and maximum; and monthly average, minimum and maximum. The adiabatic lapse rate is set with the `:AdiabaticLapseRate` keyword in the RVP file.

HBV Method (`OROCORR_HBV`)

The HBV model method from Bergstrom (1995) employs the simple orographic temperature correction method described above employing Equation 6.1, except that the monthly average temperatures are not lapsed to be consistent with their treatment in the standard HBV model.

UBC Method 1 (`OROCORR_UBC`)

The UBC watershed model orographic temperature correction method 1 employs a series of lapse rates and inflection points describing the orographic correction profile. The UBC method 1 calculates four temperature lapse rates: above and below 2000 m elevation for both daily maximum and daily minimum temperatures. The parameters are set in the RVP file using the following keword and parameter sequence: `:UBCTempLapseRates A0TLXM A0TLNM A0TLXH A0TLNH P0TEDL P0TEDU` The parameters listed above are described in Table 6.1.

Table 6.1: UBC Watershed Model temperature lapse rate parameters

Parameter	**Description**	**Units**
A0TLNH	Lapse rate for minimum temperatures when the station elevation is greater than 2000 m	C / 1000 m
A0TLNM	Lapse rate for minimum temperatures when the station elevation is less than 2000 m	C / 1000 m
A0TLXH	Lapse rate for maximum temperatures when the station elevation is greater than 2000 m	C / 1000 m
A0TLXM	Lapse rate for maximum temperatures when the station elevation is less than 2000 m	C / 1000 m
P0TEDL	Lapse rate of maximum temperature range for elevations below 2000 m	C / 1000 m
P0TEDU	Lapse rate of maximum temperature range for elevations above 2000 m	C / 1000 m

$$V = \begin{cases} \min\left(\frac{P}{A0PPTP}, 1.0\right), & \text{if } A0PPTP > 0 \\ 0, & \text{if } A0PPTP \leq 0 \end{cases} \tag{6.2}$$

where P is the precipitation rate, $A0PPTP$ is the threshold precipitation for temperature lapse rate in mm and V is a rainfall correction factor that transition a lapse rate from a dry to wet adiabatic lapse rate based on current precipitation rate. A corrected adiabatic lapse α_c is determined by providing a weighted average between the specified dry adiabatic lapse rate α_d and the wet adiabatic lapse rate α_w as shown in Equation 6.3. The wet and dry adiabatic lapse rates are specified in the RVP file using the `:WetAdiabaticLapse` and `:AdiabaticLapseRate` respectively.

$$\alpha_c = V\alpha_w + (1 - V)\alpha_d \tag{6.3}$$

A daily temperature range factor w_t is calculated as the current daily temperature range divided by the maximum temperature range parameter A0TERM shown in Equation 6.4.

$$w_t = \frac{T_{max} - T_{min}}{A0TERM} \tag{6.4}$$

The final equation for the maximum daily temperature lapse rate α_{max} and the minimum daily temperature lapse rate α_{min} are shown in Equations 6.5 and 6.6 respectively. The lapse rates have an inflection point at 2000 m in all cases, and as the daily temperature range approaches zero the lapse rates approach the corrected adiabatic lapse rate.

$$\alpha_{max} = \begin{cases} w_t A0TLXM + (1 - w_t)\alpha_c, & \text{if elevation} \geq 2000 \text{ m} \\ w_t A0TLXH + (1 - w_t)\alpha_c, & \text{if elevation} < 2000 \text{ m} \end{cases} \tag{6.5}$$

$$\alpha_{min} = \begin{cases} w_t A0TLNM + (1 - w_t)\alpha_c, & \text{if elevation} \geq 2000 \text{ m} \\ w_t A0TLNH + (1 - w_t)\alpha_c, & \text{if elevation} < 2000 \text{ m} \end{cases} \tag{6.6}$$

To do (3)

UBC Method 2 (`OROCORR_UBC2`)

The UBC Watershed Model method 2 for estimating orographic temperature effects is to dynamically derive the lapse rate from the measured temperature data collected at the meteorological gauges. This routine uses only the first two meteorological gauges (the first two listed in the RVT file) to derive the lapse rate relationships. The relationship for the maximum daily temperature lapse rate is shown in Equation 6.7 and the relationship for the minimum daily temperature lapse rate is shown in Equation 6.8.

$$\alpha_{max} = \frac{T_{max2} - T_{max1}}{z_2 - z_1} \tag{6.7}$$

$$\alpha_{min} = \frac{T_{min2} - T_{min1}}{z_2 - z_1} \tag{6.8}$$

where T_{min1} and T_{min2} are the minimum daily temperatures at stations 1 and 2 respectively, T_{max1} and T_{max2} are the maximum daily temperatures at stations 1 and 2 respectively, and z_1 and z_2 are the elevations at stations one and two respectively.

This method requires two stations configured in the RVT file and subsequent stations are ignored in the calculations.

6.3 Precipitation

Precipitation properties are interpolated directly from gauges or gridded data. At the very minimum, total daily precipitation and daily average temperature is required to generate required time series of rainfall and snowfall everywhere in the watershed.

Measured total precipitation, snow precipitation, or rain precipitation may be corrected on a gauge-by-gauge basis by using gauge-dependent rainfall and snowfall corrections to correct for observation bias. This is handled using the `:RainCorrection` and `:SnowCorrection` commands outlined in appendix A.4.1.

6.3.1 Snow-Rain Partitioning

If only total precipitation is specified at a gauge station or grid cell, then this total precipitation is partitioned into rain and snow, based upon the approach specified in the `:RainSnowPartitioning` command. The following algorithms are available:

Temperature Range Approach (`RAINSNOW_DINGMAN`)

In the temperature range approach, the snow fraction, α, is calculated from the maximum and minimum daily temperatures:

$$\alpha = \frac{T_{trans} - T_{min}}{T_{max} - T_{min}} \tag{6.9}$$

where T_{trans} is the rain/snow transition temperature (specified in the `:RainSnowTransition` command) [default: 0 °C], and T_{min} and T_{max} are the min and max daily temperatures. If T_{trans} is outside of this temperature range, the precipitation is either all snow or all rain, accordingly. This snow fraction is applied for the entire day.

Linear Approaches (`RAINSNOW_UBC` or `RAINSNOW_HBV`)

In these approaches, a linear transition between all snow and all rain is determined from the average daily temperature:

$$\alpha = 0.5 + \frac{T_{trans} - T_{ave}}{\Delta T} \tag{6.10}$$

in the range from $T_{trans} - \Delta T/2$ to $T_{trans} + \Delta T/2$, where T_{trans} and ΔT are specified in the `:RainSnowTransition` command. If T_{ave} is outside of this temperature range, the precipitation is either all snow or all rain, accordingly. This snow fraction is applied for the entire day.

Interpolate From Data (`RAINSNOW_DATA`)

To be used if snowfall (or the snow fraction) is explicitly reported in the gauge/gridded data.

6.3.2 Orographic Precipitation Effects

Orographic effects may be applied to correct total interpolated precipitation at each HRU based upon HRU elevation. The fraction of precipitation in the form of snow or rain is not modified by these corrections.

HBV Method (`OROCORR_HBV`)

From the HBV model Bergstrom (1995):

$$P = P_g \cdot (1.0 + \alpha(z - z_g)) \tag{6.11}$$

where P is the total precipitation rate, P_g is the measured gauge precipitation, z and z_g are the elevation of the HRU and gauge, respectively, and α, the precipitation correction lapse rate, is 0.00008 m^{-1} below 5000 masl, 0 above this elevation.

UBC Method 1(`OROCORR_UBC`

The UBC Watershed Model method 1 for orographic correction of precipitation estimates employs a temperature-corrected lapse rate with two inflection points (Quick, 2003). The base orographic correction equation is shown in Equation 6.12:

$$P = P_g \cdot (1 + \alpha F_t)^{\frac{z-z_g}{100}} \tag{6.12}$$

where P is the total applied precipitation rate, P_g is the measured gauge precipitation, z and z_g are the elevation of the HRU and gauge, respectively, and α, the precipitation correction lapse rate. F_t is a temperature correction factor shown in equation 6.13:

$$F_t = \begin{cases} 1, & \text{if } t_{band} \leq 0 \text{ C} \\ 1 - \mathit{A0STAB}\,(t_{band})^2\,, & \text{if } t_{band} > 0 \text{ C} \end{cases} \tag{6.13}$$

where $A0STAB$ is the precipitation gradient modification factor, and t_{band} is the temperature at the first listed elevation band in the model. F_t is constrained between 0 and 1.

Simple Method (`OROCORR_SIMPLELAPSE`)

The simple precipitation lapse rate method employs a simple linear adiabatic method as outlined in Equation 6.14 below:

$$P = P_g + \alpha(z - z_g) \tag{6.14}$$

where P is the total precipitation rate, P_g is the measured gauge precipitation, z and z_g are the HRU and gauge elevations respectively and α is the precipitation correction lapse rate specified using the `:PrecipitationLapseRate` key word in the RVP file. Checks azre included to ensure positivity of the precipitation rate

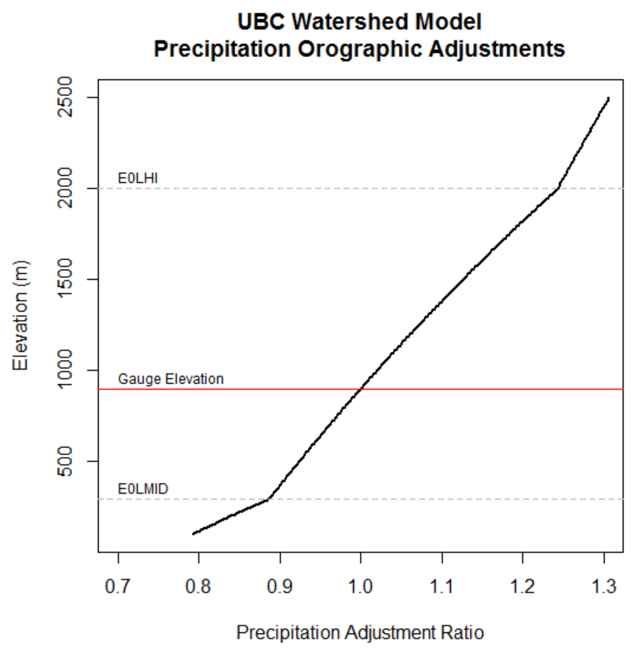


Figure 6.1: UBC Watershed Model Orographic Correction

6.4 Potential Evapotranspiration (PET)

A variety of potential evapotranspiration (PET) estimation algorithms of varying complexity are available for calculating PET within an HRU. These PET algorithms use many of the same relationships, including those for the saturated vapor pressure as a function of temperature,

$$e_s(T) = 0.6108 \cdot \exp\left(\frac{17.23T}{T + 237.3}\right) \tag{6.15}$$

and the slope of this curve, $\Delta(T) = de_s/dT$,

$$\Delta = \frac{4098}{(T + 237.3)} \cdot e_s(T) \tag{6.16}$$

where T is in °C. The latent heat of vaporization of water, λ_v, is estimable by:

$$\lambda_v = 2.495 - 0.002361 * T \tag{6.17}$$

and the psychrometric constant, γ is here treated as varying with atmospheric pressure, P,

$$\gamma = \frac{c_a}{0.622 \cdot \lambda_v} P \tag{6.18}$$

where C_a is the specific heat of air, equal to 1.012x10^{-3} MJ/kg/K.

Note that all of the algorithms below estimate daily PET. Methods are required to downscale these daily estimates to sub-daily timesteps, as discussed in 6.10.

6.4.1 PET Estimation

Constant PET (PET_CONSTANT)

The daily PET value used is constant and uniform rate of 3 mmd^{-1}.

From file (PET_DATA)

The daily PET is explicitly specified at each gauge or grid cell (see section A.4 for details) and interpolated in-between. This enables any measured ET or user-specified means of calculating PET to be used.

From Monthly (PET_FROMMONTHLY)

Used in the HBV Model Bergstrom (1995). Monthly PET and temperature norms are provided at the gauge using the `:MonthlyAveEvaporation` and `:MonthlyAveTemperature` commands. These estimates are assumed not to vary year-to-year. Daily estimates of PET may then be obtained from:

$$\text{PET} = \text{PET}_{mon} \cdot \min((1 + \tfrac{1}{2}(T_{ave} - T_{mon}), 2) \tag{6.19}$$

where PET$_{mon}$ and T_{mon} are the daily PET [mm/d] and temperature norms for the current month, and T_{ave} is the average daily temperature. Checks are used to ensure PET is positive and doesn't exceed twice the average representative monthly PET.

Penman Monteith (PET_PENMAN_MONTEITH)

From Monteith (1965). The standard Penman-Monteith equation estimates daily reference evapotranspiration over a reference vegetation,

$$\text{PET} = \frac{1}{\lambda_v \rho_w} \cdot \left[\frac{\Delta}{\Delta + \gamma^*} R_n + \frac{\rho_a C_a c_a}{\Delta + \gamma^*}(e_s - e)\right] \tag{6.20}$$

where λ_v [MJ/kg] is the latent heat of vaporization of water, ρ_w [kgm^{-3}] is the density of water, $\Delta = de_s/dT$ is the slope of the saturated vapor pressure curve, R_n [MJm^{-2}d^{-1}] is the net radiation to the system, ρ_a is the air density, C_a [MJ/kg] is the specific heat of air, c_{atm} [md^{-1}] is the atmospheric conductance, e is the vapor pressure of the atmosphere, $e_s(T)$ [kPa] is the current saturated vapor pressure of the atmosphere, a function of temperature, and γ^* [kPa/ °C] is the corrected psychrometric constant,

$$\gamma^* = \left(1 + \frac{c_a}{c_{can}}\right)\gamma \tag{6.21}$$

where c_{can} [m/d] is the canopy conductance, and γ [kPa/ °C] is calculated using 6.18. The final expression is converted from m/d to mm/d. The atmospheric conductance is calculated using the following relationships Dingman (2002):

$$c_{atm} = v \cdot \frac{VK^2}{\ln\left(\frac{z_{ref}-z_0}{z_{rough}}\right)\ln\left(\frac{z_{ref}-z_0}{z_{vap}}\right)} \tag{6.22}$$

where VK is the Von Karman Constant (0.42), z_{ref} is the reference height [m] at which the wind velocity v [m/d] is reported, z_0 [m] is the zero-plane displacement height, z_{rough} is the roughness height [m], and z_{vap} is the vapour roughness height [m]. These parameters are predominantly calculated from the ground roughness and canopy heights. The canopy conductance is calculated as a function of vegetative leaf area index Dingman (2002):

$$c_{can} = 0.5 \cdot c_{leaf} \cdot \mathrm{LAI} \tag{6.23}$$

where c_{leaf} is the leaf conductance [m/d], calculated using the expressions detailed in Dingman (2002).

Penman Combination (PET_PENMAN_COMBINATION)

From Penman (1948). A similar expression to the Penman Monteith equation, daily reference ET is calculated from the following equation:

$$\mathrm{PET} = \frac{1}{\lambda_v \rho_w} \cdot \left[\frac{\Delta}{\Delta+\gamma} R_n\right] + \left[\frac{\gamma \epsilon_v v}{\Delta+\gamma}(e_s - e)\right] \tag{6.24}$$

i.e., here the deficit-driven evapotranspiration (the second term) is treated using the wind velocity, v [m/s] and a vertical transport efficiency factor, ϵ_v, calculated as

$$\epsilon_v = \frac{0.622\rho_a}{6.25 \cdot e\rho_w} \cdot \left(\ln\left(fracz_{ref} - z_0 z_{rough}\right)^{-2}\right) \tag{6.25}$$

terms are defined as defined above in the description of the PET_PENMAN_MONTEITH algorithm.

Priestley-Taylor (PET_PRIESTLEY_TAYLOR)

From Priestley and Taylor (1972). A simplified version of the Penman-Monteith approach where only net radiation explicitly drives daily ET, with an additional correction factor for the unmodeled ET driven by vapor deficit. The Priestley-Taylor equation is given by:

$$\mathrm{PET} = 1.26 \cdot \frac{1}{\rho_w \lambda_v} \cdot \left[\frac{\Delta}{\Delta+\gamma} R_n\right] \tag{6.26}$$

where R_n is the net radiation [MJ/m^2/d], and other terms are defined as above in the description of the PET_PENMAN_MONTEITH algorithm. The factor of 1.26 is used to scale the radiation-driven ET to account for the unmodeled vapor-driven ET.

Hargreaves (PET_HARGREAVES)

From Hargreaves and Samani (1982).

$$\mathrm{PET} = \frac{1}{\rho_w \lambda_v} \cdot S_{ET} \cdot 0.000938 \cdot \sqrt{T^{mon}_{max,F} - T^{mon}_{min,F}} T_{ave,F} \tag{6.27}$$

where S_ET [MJ/m^2/d] is the extraterrestrial shortwave radiation, the temperatures $T^{mon}_{max,F}$ and $T^{mon}_{min,F}$ are the maximum and minimum monthly temperatures in Farenheit, and $T_{ave,F}$ is the daily temperature in Farenheit (converted internally within the code). The temperature factors attend to the impact of cloud cover and atmospheric interference with the extraterrestrial radiation.

Hargreaves 1985 (PET_HARGREAVES_1985)

From Hargreaves and Samani (1985). The 1985 Hargreaves equation, an empirical approach based solely on temperature and incoming solar radiation. Similar to PET_HARGREAVES, but it metric units.

$$\text{PET} = \frac{1}{\rho_w \lambda_v} \cdot S_{ET} \cdot 0.0023 \cdot \sqrt{T_{max} - T_{min}}\,(T_{ave} + 17.8) \tag{6.28}$$

where T_{ave}, T_{max}, and T_{min} are the average, maximum, and minimum daily air temperature, and S_ET [MJ/m^2/d] is the extraterrestrial shortwave radiation.

UBC (PET_MONTHLY_FACTOR)

Method used in the UBC Watershed Model (Quick, 1995). PET is calculated using the following formula:

$$\text{PET} = E_{mon} \cdot \max(T_{ave}, 0) \cdot \delta_{forest}$$

where E_{mon} [mm/d/K] is a monthly PET factor (specified using the `:MonthlyEvapFactor` command in the .rvt file), T_{ave} is the daily average temperature and δ_{forest} is the land use parameter FOREST_PET_CORR), applied only to forested regions.

Hamon (PET_HAMON)

From Hamon (1961). PET is calculated using the following relationship:

$$\text{PET} = 1115 \cdot \frac{e_{sat} L_d^2}{T_{ave}}$$

where e_{sat} is the saturated vapor pressure [kPa], T_{ave} is the average daily temperature [K], L_d is the day length [d], and the PET is in mm/d. The constant 1115 includes both units conversion factors and an approximate relationship to convert saturated vapor pressure and temperature to absolute humidity.

Turc 1961 (PET_TURC_1961)

From Turc (1961) as reported in Liu et al. (2005). This empirical PET estimation algorithm has no additional parameters required.

$$\text{PET} = \begin{cases} 0.013\left(\frac{T_{ave}}{T_{ave}+15}\right)(23.88 * S_n + 50)\left(1 + \frac{50-RH}{70}\right) & \text{for RH<50\%} \\ 0.013\left(\frac{T_{ave}}{T_{ave}+15}\right)(23.88 * S_n + 50) & \text{for RH}\geq\text{50\%} \end{cases}$$

where the PET is in mm/d, T_{ave} is the average daily temperature [°C], S_n is the daily net shortwave radiation [MJ/m^2/d], and RH is the relative humidity expressed as a percentage.

Makkink 1957 (PET_MAKKINK_1957)

From Makkink (1957) as reported in Liu et al. (2005).

$$\text{PET} = 14.57\left(\frac{\Delta}{\Delta+\gamma}\right)\frac{S_n}{58.5} - 0.12$$

where Δ is the slope of the saturation vapor pressure-temperature curve [kPa/ °C], γ is the psychrometric constant, and S_n is the net incoming solar radiation [MJ/m^2/d].

6.4.2 PET Orographic Effects

Orographic effects are calculated using the following algorithms, specified using the `:OroCorrPET` command in the .rvi file.

HBV Method (OROCORR_HBV)

From the HBV model (Bergstrom, 1995):

$$\text{PET} = \text{PET}_g \cdot \alpha\,(1-\beta)\,(z-z_g) \tag{6.29}$$

where α is the global PET correction factor (`GLOBAL_PET_CORR`), β is the HBV precip correction factor (`HBV_PRECIP_CORR`), and z and z_g are the HRU elevation and the gauge elevation, respectively.

PRMS Method (OROCORR_PRMS)

To do (4) This orographic correction factor is described in the users's manual of the PRMS model (**?**). It uses the maximum saturated vapor pressure, e_{sat}^{max} [kPa] (calculated from the average August temperature) and the minimum saturated vapor pressure e_{sat}^{min} [kPa] (calculated from the average February temperature).

$$\text{PET} = \text{PET}_g \cdot \frac{1}{68 - 0.0118z + \frac{650}{e_{sat}^{max}-e_{sat}^{min}}}) \tag{6.30}$$

where z is the HRU elevation [masl] and PET$_g$ is the estimated PET at the gauge (presumed to be calculated at an elevation of zero). Note that because this algorithm implicitly includes orographic temperature effects, it must be used with care in combination with orographic temperature corrections.

6.5 Shortwave Radiation

Solar radiation contributes to the earth surface's energy balance, and is important for estimating snow melt and evapotranspiration, amongst other things. Since solar radiation is not directly measured in many places, here the standard routines documented in (Dingman, 2002) are used to estimate critical terms needed to estimate extraterrestrial shortwave radiation. This can then be corrected using information about cloud cover and/or optical air mass. Used in many of these calculations is the day angle, Γ [rad], and the solar declination, δ [rad]:

$$\Gamma = \frac{2\pi J}{365} \tag{6.31}$$

$$\begin{aligned}\delta = \quad & 0.006918- & 0.399912 \cdot \cos(\Gamma) + \\ & 0.070257 \cdot \sin(\Gamma)- & 0.006758 \cdot \cos(2 \cdot \Gamma) + \\ & 0.000907 \cdot \sin(2\Gamma)- & 0.002697 \cdot \cos(3 \cdot \Gamma) + \\ & 0.001480 \cdot \sin(3\Gamma) & \end{aligned}$$

Day length is calculated as follows, with additional corrections for polar latitudes:

$$\texttt{Day Length} = \frac{\arccos(-\tan(\delta) \cdot \tan(\Lambda))}{\pi}$$

where Λ is the latitude of the location (in radians). In RAVEN, net shortwave is calculated as

$$S_n = (1 - \alpha) \cdot f_{can} \cdot f_{cloud} \cdot S_{clear} \tag{6.32}$$

where f_{can} and f_{cloud} [0..1] are correction factors for canopy cover and cloud cover, respectively, and the clear sky solar radiation is given as

$$S_n = f_{atm} \cdot f_{asp} \cdot S_{ET} \tag{6.33}$$

where f_{atm} and f_{asp} [0..1] are a correction factors for atmospheric refraction and slope/aspect of the ground surface, S_{ET} is the extra terrestrial radiation. Section 6.5.1 details methods for calculating S_{ET}, section 6.5.2 details methods for handling f_{atm}, section 6.5.3 details methods for handling f_{cloud} and section 6.5.4 details methods for handling f_{can}.

6.5.1 Extraterrestrial Shortwave Generation

The following shortwave radiation estimation algorithms are available, and are specified using the `:SWRadiationMethod` command in the .rvi file.

Default ET Flux (`SW_RAD_DEFAULT`)

Extraterrestrial radiation flux on a horizontal plane is calculated using Dingman (2002):

$$S_{ET} = I_{sc} \cdot E_0 \cdot [\cos(\delta) \cdot \cos(\Lambda) \cdot \cos(2\pi t) + \sin(\delta)\sin(\Lambda)] \tag{6.34}$$

where I_{sc} is the solar constant (118.1 MJm^{-2}d^{-1}), E_0 is an eccentricity correction (see Dingman (2002)), and t is the time of day in days (i.e., $t = 0$ is midnight, $t = 0.5$ is noon). Corrections are applied for radiation on a sloping surface (i.e., on HRUs with a non-zero slope). Aspects are corrected for in the default approach using the corrections put forth in Dingman (2002), and can handle the two sunset effect.

UBC Watershed Model approach (SW_RAD_UBCWM)

Shortwave radiation is calculated using the same equations as the `SW_RAD_DEFAULT` approach (equation 6.34), but employs a correction to the day length to account for mountain barrier effects. Two sets of monthly correction parameters are employed in this method to correct for SW radiation for north- and south-facing slopes. The parameters are included in the `UBCNorthSWCorr` and `UBCSouthSWCorr` keywords in the RVP file with one parameter for each month (January to December). The HRU orientation factor is calculated as a function of the aspect of the HRU

$$O = 1 - \left| \frac{\theta}{\pi} - 1 \right|$$

where θ is the dominant aspect direction and O is the orientation (eg. north = 0 and south = 1, east/west = 0.5). The final SW radiation estimate is

$$f_{asp} = [O \cdot C_S + (1 - O) \cdot C_N]$$

where f_{asp} is the correction factor for shortwave radiation on an inclined plane, S_{ET} is the uncorrected shortwave radiation estimate based on equation 6.34, and C_S and C_N are the south and north correction factors respectively (from `UBC_S_CORR` and `UBC_N_CORR`.

Interpolate From Data (SW_RAD_DATA)

The incident shortwave radiation is read from a file, specified at one or more gauge locations. The radiation could be either measured, generated from an atmospheric model, or estimated using an external preprocessor. If incident shortwave is provided directly, cloud cover corrections (but not aspect, or canopy corrections) are implicitly contained in this figure. What is actually being input is

$$\cdot f_{cloud} f_{atm} \cdot S_{ET}$$

Additional algorithms are required to attend to slope/aspect and canopy corrections.

6.5.2 Clear Sky Radiation

As radiation passes through the earths atmosphere, energy is absorbed and scattered by particles and water vapor, both in cloudy and cloud-free areas. Corrections must be made to extraterrestrial radiation to account for this.

Dingman (SW_RAD_DEFAULT)

The approach outlined in Dingman (2002), total incident radiation is calculated as:

$$f_{atm} = (\tau_{dir} + 0.5(1 - \tau_{diff})) \cdot (1 + 0.5(1 - \tau_{diff})\alpha)$$

where α is the surface albedo, and the scattering correction factors for diffuse and direct solar radiation τ_{diff} and τ_{dir} are given by

$$\begin{aligned} \tau_{dir} &= \exp\left(-0.124 - 0.0207W_p - (0.0682 + 0.0248W_p)M_{opt}\right) \qquad (6.35) \\ \tau_{diff} &= \exp\left(-0.0.363 - 0.0084W_p - (0.0572 + 0.0173W_p)M_{opt}\right) \end{aligned}$$

where the precipitable water vapor, W_p, is calculated as $W_p = 1.12\exp(0.0614T_d)$, where T_d is the dew point temperature, and the optical air mass, M_{opt}, is calculated using the methods of Yin (1997).

UBC Watershed Model approach (`SW_RAD_UBCWM`)

In the UBC watershed model, the corrections for atmospheric scattering and adsorption are given as

$$f_{atm} = \exp(-2.0 \cdot (0.0128 - 0.0234 \ln(m_a))$$

where the air mass, m_a is given by

$$m_a = \frac{1 - 0.001 \cdot z}{\cdot \left[\cos(\delta) \cdot \cos(\Lambda) \cdot \cos(2\pi t) + \sin(\delta)\sin(\Lambda)\right]} \tag{6.36}$$

This product $f_{atm} \cdot S_{ET}$ is numerically integrated over the course of the day to estimate the daily clear sky radiation. The day length in this integration calculation is corrected for using a mountain barrier correction.

6.5.3 Cloud Cover Corrections

Additional corrections are required to handle cloud cover. While the algorithms for estimating actual cloud cover are included in section 6.7 below, the use of the cloud cover factor for estimating incident radiation is treated here.

UBC approach (`SW_CLOUDCOV_CORR_UBC`)

The UBC watershed model corrects shortwave radiation due to cloud cover using the following equation

$$f_{cloud} = (1 - (1 - POCAST) \cdot C_c)$$

where S_C is the shortwave radiation corrected for cloud cover, S is the uncorrected shortwave radiation, C_C is the cloud cover correction factor and $POCAST$ is the cloud penetration factor specified in the RVP file with the `:UBCCloudPenetration` keyword.

UBC Watershed model approach (`SW_CLOUDCOV_CORR_DINGMAN`)

The cloud cover correction factor may also be estimated as outlined in Dingman (2002, Eq. 5-30):

$$f_{cloud} = (0.355 + 0.68 \cdot (1 - C_c)) \tag{6.37}$$

where C_c is cloud cover. This approach does not require any parameters to be set in the RVP file.

6.5.4 Canopy Cover Corrections

Calculates the ratio of solar radiation under forest canopy relative to open. The default canopy cover correction method is no correction (`SW_CANOPY_CORR_NONE`).

UBC Method (SW_CANOPY_CORR_UBC)

To correct for shortwave correction due to canopy cover the UBC watershed model method employs the following equation

$$f_{can} = F_E$$

where S_C is the shortwave energy corrected for canopy cover, S is the uncorrected shortwave energy, and F_E is the forest cover correction factor specified using the `:UBCExposureFactor` command in the RVP file.

Bulk transmittance approach (SW_CANOPY_CORR_STATIC)

The Bulk transmittance approach provides a static canopy transimittance based on leaf-area index and stem-area index estimates to produce a "sky view" factor, or the fraction of the ground that receives sunlight (Dingman, 2002):

$$f_{can} = \exp(-k(\text{LAI} + \text{SAI}))$$

where k is the extinction coefficient, LAI is the leaf-area index and SAI is the stem-area index. The extinction coefficient, leaf-area index and stem-area index are supplied or calculated from parameters within the `:VegetationClasses` parameter structure in the RVP file by the `SVF_EXTINCTION`, `MAX_LAI`, and `SAI_HT_RATIO` columns respectively.

The leaf-area index is calculated based on the sparseness:

$$\text{LAI} = (1 - f_c)(\text{LAI}_{max})$$

where f_c is the sparseness index (`FOREST_SPARSENESS`) and LAI_{max} is the maximum leaf-area index. Stem-area index is estimated as follows:

$$\text{SAI} = (1 - f_c)(C_s \cdot h_{veg})$$

where C_s is the ratio between vegetation height and the maximum stem-area index and h_{veg} is the vegetation height.

6.6 Longwave Radiation

Longwave radiation is the electromagnetic radiation emitted by materials with near-earth-surface temperatures. The net longwave is the difference between the incident longwave emitted (or back scattered) by the atmosphere, clouds, and canopy and the outgoing radiation from the land surface. Unlike with shortwave radiation, in Raven only the net longwave radiation is tracked.

6.6.1 Data method (LW_RAD_DATA)

The net longwave radiation is read from a file, specified at one or more gauge locations or as a gridded climate product. The radiation could be either measured or estimated using an external preprocessor.

6.6.2 Default method (LW_RAD_DEFAULT)

Net longwave radiation is treated using the Stefan -Boltzmann law, with a correction factor for the inefficiency of the land and atmospheres as black-body emitters.

$$L_n = \sigma \cdot \epsilon_s \cdot \left(\epsilon_{atm} \cdot T_{atm,K}^4 - T_{s,K}^4\right)$$

Where σ is the Stefan Boltzmann constant (4.9x10^{-9} MJm^{-2}d^{-1}K^{-4}), $T_{atm,K}$ and $T_{s,K}$ [°K] are the effective temperatures of the atmosphere and ground surface (here presumed equal to the air temperature in Kelvin), and ϵ_s and ϵ_{atm} are the effective emissivities of the surface and atmosphere, respectively. In RAVEN, the surface emissivity is held constant as $\epsilon_s = 0.99$ and the atmospheric emissivity is calculated as Dingman (2002)

$$\epsilon_{atm} = (1 - F_c) \cdot 1.72 \cdot \left(\frac{e}{T_{a,K}}\right)^{1/7} \cdot (1 + 0.22 \cdot C_C^2) + F_c$$

where F_c [0..1] is the forest cover (treated as a blackbody), e is the vapor pressure, $T_{a,K}$ is the air temperature in Kelvin, and C_c is the cloud cover.

6.6.3 UBC Method (LW_RAD_UBC)

The longwave radiation is estimated in the UBC Watershed model separately for open and forested covers. The open longwave radiation is estimated using

$$L_o = (1 - f_{cloud}) \cdot \lambda_f \rho_w \cdot (-20 + 0.94 T_{avg}) + f_{cloud} \cdot \lambda_f \rho_w \cdot (1.24 T_{min}$$

where L_o is the net longwave radiation estimate for open forest cover in mmd^{-1}, T_{avg} °Cis the daily average temperature, T_{min} °Cis the daily minimum temperature, f is the UBC cloud cover correction factor (see Section 6.7), and λ_f is the latent heat of fusion. The net longwave radiation estimate for forest covered areas is:

$$L_f = \lambda_f \rho_w f_{LW} T_{avg}$$

where L_f is the longwave radiation estimate for open forest cover in mmd^{-1}, t_{avg} is the daily average temperature, and f_{LW} is the temperature multiplier factor in mmd^{-1}K^{-1} which is set in the RVP file using the `:UBCLWForestFactor` keyword. If the forest cover for an HRU is greater than zero then Equation **??** is employed. Note that this expression is a linearization of the Stefan-Boltzmann law.

6.6.4 HSPF Method (LW_RAD_HSPF)

Net longwave radiation is given as a simple function of average daily temperature, T_{avg} [°C]

$$L_n = 0.361 * (T_{avg} - 6.6) \tag{6.38}$$

where L_n is in MJm^{-2}d^{-1}.

6.7 Cloud Cover

This section outlines the various method for the estimation of a cloud cover in the model and the associated cloud cover corrections for incident short wave radiation. The default cloud cover method is `CLOUDCOV_NONE`, implying no cloud cover estimation or cloud cover correction.

6.7.1 No cloud cover calculations (CLOUDCOV_NONE)

No cloud cover is the default approach to cloud cover for Raven and can be set explicitly in the RVI file using the `:CloudCoverMethod` keyword of `NONE`, or by excluding the keyword entirely.

6.7.2 Interpolate From Data (CLOUDCOV_DATA)

The cloud cover data [0-1] may be incorporated from gauge data if available in which case the `CLOUDCOV_DATA` option for the `CloudCoverMethod` keyword should be employed in the RVI file. The cloud cover data is stored in the meteorological time series data files (see Section A.4 for details).

6.7.3 UBC approach (CLOUDCOV_UBC)

Cloud cover factor in the UBC watershed model are estimated by determining the daily temperature range as observed at the meteorological gauges that influence an HRU and comparing that range to specified cloud temperature range parameters. The observed temperature range for the HRU is calculated as

$$\Delta T = T_{max} - T_{min}) \tag{6.39}$$

where T_{max} and T_{min} are the interpolated maximum and minimum temperatures and Δt is the temperature range at HRU. The cloud cover correction factor is

$$C_c = \begin{cases} 1, & \text{if } \Delta T \leq T_{cmin} \\ 1 - \frac{\Delta T - T_{cmin}}{T_{cmax} - T_{cmin}}, & \text{if } T_{cmin} > \Delta T > T_{cmax} \\ 0, & \text{if } \Delta t \geq T_{cmax} \end{cases} \tag{6.40}$$

where C_c is the cloud cover factor [0-1], and T_{cmin} and T_{cmax} are the cloud cover temperature ranges in °Cas specified for each gauge within the RVT file using the keyword `:CloudTempRanges`.

6.8 Energy

This section includes a number of processes that are involved in the energy balance in the Raven model, including the estimates of potential snowmelt

6.8.1 Potential Melt

Potential snow melt can be estimated using a number a methods in the Raven model. To set the appropriate process in the model the RVI must include the `:PotentialMeltMethod` keyword along with the appropriate value for the method selected.

Degree Day Method (`POTMELT_DEGREE_DAY`)

The degree day method estimates a potential snow melt using an temperature index approach as described in, e.g., Dingman (2002):

$$M_{melt} = M_a \cdot \max(T - T_f, 0)$$

where M_{melt} is the potential melt rate [mm/day], T is the atmospheric temperature of the HRU [deg C], T_f is the freeze/melt temperature [°C] (zero by default), and M_a is the melt factor [mm/day/deg C], specified using the land use/land type parameter `MELT_FACTOR`.

UBC approach (`POTMELT_UBC`)

The UBC watershed model approach to calculating potential snowmelt is described below. The model requires a certain number of participating parameters defined in the RVP file: `FOREST_COVERAGE` supplied in the `:LandUseClasses` table, and `UBC_MIN_SNOW_ALBEDO`, `UBC_SW_S_CORR` and `UBC_SW_N_CORR` provided as global variables. The total snow melt is an accumulation of separate melt components:

$$M_{melt} = \frac{1}{\lambda_f \rho_w}\left((1-\alpha_s)S + L_n + Q_c + Q_a + Q_r\right)$$

where M_{melt} is the total potential melt rate $[mm/d]$, S is the incoming shortwave radiation, α_s is the snow albedo, L_n [MJ/m^{2}2/d] is the long wave radiation, Q_c [MJ/m^{2}2/d] is the convective melt energy, Q_a [MJ/m^{2}2/d] is the condensation or advective melt energy and Q_r [MJ/m^{2}2/d] is the melt energy due to rainfall. The convective and advective melt energy is estimated using

$$Q_c = 0.113 \cdot p \cdot T_a \cdot V \cdot R_M$$
$$Q_a = 0.44 \cdot T_{min} \cdot V \cdot R_M \cdot [(1-f_c)p + f_c]$$

where p is the air pressure T_a is the average air temperature, T_{min} is the minimum daily air temperature, V is the wind velocity, f_c is the fraction of forest cover and R_M is a reduction factor as described below

$$R_M = 1.0 - 7.7 \cdot R_I$$
$$0 \leq R_M \leq 1.6$$

where R_I is a linearized estimate of Richardson's number

$$R_I = \frac{0.095 \cdot T_{avg}}{V^2}$$

The rainfall related melt is estimated using the following equation

$$Q_r = k \cdot T_a \cdot P_r$$

where k represents the heat content of the rain mm/C and P_r is the rainfall over the time step.

HBV Method (POTMELT_HBV)

The potential melt in the HBV method is given by a corrected version of the degree day approach, with the corrected melt coefficient given by

$$M_a' = C_f \cdot C_a \left(M_{a.min} + (M_{a.max} - M_{a.min}) \cdot \frac{1.0 - \cos(\Gamma - \Gamma_s)}{2} \right) \tag{6.41}$$

where M_a' is the potential melt coefficient, C_f is the forest correction factor, C_a is the aspect correction factor, A_c is the aspect correction factor, $M_{a.max}$ and $M_{a.min}$ are the maximum and minimum potential melt rate parameters specified using the `MELT_FACTOR` and `MIN_MELT_FACTOR` keywords respectively, and are specified in the land use parameters. Γ is the day angle calculated using equation 6.31 and Γ_s is the winter solstice angle and is a model constant of 23.5°. The forest and aspect correction factors are described below:

$$C_f = (1.0 - F_c) \cdot (1.0 + (F_c) \cdot M_{RF}) \tag{6.42}$$

$$C_a = \max\left(1 - A_m \cdot C_s \cdot \cos(\theta), 0.0\right) \tag{6.43}$$

where F_c is the fraction of forest cover, M_{RF} is the forest melt correction parameter specified using `HBV_MELT_FOR_CORR`, A_m is the aspect melt correction parameter `HBV_MELT_ASP_CORR`, and θ is the landscape aspect angle. C_s is slope correction factor described below:

$$C_s = (1.0 - F_c) \cdot (1.0 + (F_c) \cdot \sin(\theta_s)) \tag{6.44}$$

where θ_s is the landscape slope.

$$M_{melt} = M_a' \cdot (T - T_f) \tag{6.45}$$

Restricted Method (POTMELT_RESTRICTED)

The potential melt rate is given by the degree day method plus a correction term due to net incoming radiation:

$$M_{melt} = M_a \cdot (T - T_f) + \frac{S_n + L_n}{\lambda_f \rho_w} \tag{6.46}$$

where S_n and L_n are the net incoming radiation, and the melt factor, M_a is the land surface parameter `MELT_FACTOR`.

Energy Balance Method (POTMELT_EB)

Similar to the `POTMELT_UBC` approach, except the estimates for Q_c, Q_a and Q_r are obtained using the methods of Dingman (2002). This approach requires no additional parameters: all energy estimates are taken from the current air and surface temperatures, and roughness heights of the land/vegetation.

6.9 Atmospheric Variables

This section includes various methods for estimating wind speed, relative humidity,

6.9.1 Wind Speed

The following methods can be used to estimate the wind speed at 2 metres, as used for a number of ET and potential melt estimation algorithms.

Constant Method (WINDVEL_CONSTANT)

Returns a constant value of 2.0 m/s (the global average).

Interpolate From Data (WINDVEL_DATA)

Wind velocity is interpolated from data supplied at a gauge location, as specified in the .rvt file.

UBC Watershed model approach (WINDVEL_UBC)

An algorithm adapted from the UBC Watershed model. The base wind speed, v_b [km/hr] is first estimated to be between a reasonable range using the temperature range for the day

$$v_b = (1 - \omega)v_{max} + (\omega)v_{vmin}$$

where $v_{max} = 8$ km/hr, $v_{min} = 1$ km/hr, and $\omega = 0.04 \cdot \min(T_{max} - T_{min}, \Delta T_{max})$. Here T_{max} and T_{min} are the orographically corrected minimum and maximum daily temperature, ΔT_{max} is the global parameter MAX_RANGE_TEMP, which may be corrected for elevation. If the following maximum temperature range is smaller than MAX_RANGE_TEMP, it overrides MAX_RANGE_TEMP:

$$\Delta T_{max} = 25.0 - 0.001 \cdot P0TEDL \cdot z_g - 0.001 \cdot P0TEDU(z - z_g)$$

where P0TEDL and P0TEDU are global lapse rate parameters specified using the :UBCTempLapseRates command, and z_g and z are the elevation of the temperature gauge and HRU, respectively. The wind velocity is then converted to m/s, then corrected for forest cover and elevation,

$$v = \alpha_f \cdot (0.001 \cdot z)^{1/2} \cdot v_b$$

where α_f is equal to 1 for bare ground and 0.7 if FOREST_COVER is greater than zero.

6.9.2 Relative Humidity

The following algorithms may be used to estimate relative humidity in Raven:

Constant Method (RELHUM_CONSTANT)

The relative humidity is (somewhat arbitrarily) estimated to be 0.5.

Interpolate From Data (RELHUM_DATA)

Relative humidity is interpolated from data supplied at a gauge location or gridded data, as specified in the .rvt file.

Minimum Daily Temp as estimator of dew point (RELHUM_MINDEWPT)

The minimum daily temperature is assumed to be equal to the dew point, allowing relative humidity to be estimated as

$$RH = \frac{e_s(T_{min})}{e_s(T_{ave})}$$

where T_{min} and T_{ave} are the minimum and average daily temperatures and $e_s(T)$ is the saturated vapor pressure, a function of temperature.

6.9.3 Air Pressure

The following approaches may be used to estimate atmospheric pressure:

Constant Method (AIRPRESS_CONSTANT)

A constant air pressure of 101.3 kPa is used (air pressure at standard temperature of 25 °C)

Interpolate From Data (AIRPRESS_DATA)

Air pressure is interpolated from data supplied at a gauge location, as specified in the .rvt file.

UBC Watershed model approach (AIRPRESS_UBC)

Air pressure is corrected for elevation above mean sea level, z,

$$P = 101.3 \cdot (1 - 0.001z)$$

where P is in kPa.

Basic Approach (AIRPRESS_BASIC)

Air pressure is corrected for both temperature and pressure using the following relationship

$$P = 101.3 \cdot \left(1 - 0.0065\frac{z}{T_{ave}^K}\right)^{5.26}$$

where T_{ave}^K is the average temperature for the time step in °K, and z is the HRU elevation.

6.10 Sub-daily Corrections

Many of the above algorithms estimate incoming radiation, potential melt, and/or ET on a daily timescale. When simulating at a sub-daily timescale, it is advantageous to be able to downscale these estimates for smaller time intervals. If a time step less than Δt=1.0 is used, the sub-daily corrections are used to modify the following quantities:

- potential melt
- shortwave radiation
- PET

Simple Method (`SUBDAILY_NONE`)

No modification is used.

Simple Method (`SUBDAILY_SIMPLE`)

The half-day length is used to scale a cosine wave which peaks at midday, is zero after sunset and before sunrise, and has a total area of 1.0 underneath; the average value of this sine wave over the time step is used as the subdaily correction.

$$\delta = \frac{1}{\Delta t} \int_{t}^{t+\Delta t} -\frac{1}{2}\cos\left(\frac{\pi t}{DL}\right) dt$$

where DL is the day length, in days.

UBC Watershed model approach (`SUBDAILY_UBC`)

To do (5)

6.11 Monthly Interpolation

Various methods to be used for interpolation and use of all monthly data.

Uniform Method (`MONTHINT_UNIFORM`)

Monthly values are assumed to be uniform throughout the month, jumping abruptly when moving from month to month.

Related To Data Of The First Day Of The Month (`MONTHINT_LINEAR_FOM`)

Monthly values are linearly interpolated, assuming that the specified value refers to the first day of the month.

Related To Data Of The Median Day Of The Month (`MONTHINT_LINEAR_MID`)

Monthly values are linearly interpolated, assuming that the specified value refers to the middle of the month.

Related To Data Of The Twenty-First Day Of The Month (`MONTHINT_LINEAR_21`)

Monthly values are linearly interpolated, assuming that the specified value refers to the 21st day of the month (as done in the UBC Watershed model).

Chapter 7

Tracer and Contaminant Transport

Raven can be used to track contaminants and/or tracers (referred to as constituents) through a watershed via advection. It also has the capacity to (in the future) simulate dispersion, turbulent dispersion, and single and multi-species chemical reactions, volatilization, and settling; these capabilities have yet to be implemented. Transport is now limited to single-subbasin models; mass cannot yet be routed downstream through the channel reach.

The advective transport capabilities of Raven are relatively simple in concept. During each time step, water exchange in the HRU is first calculated. Using the known water fluxes between storage compartments over a given time step, and the mass of a given constituent in each storage compartment, the net mass flux is calculated between all storage compartments for the time step. Internally, the mass density (in mg/m^2) is stored in each storage compartment (i.e., soils, surface water, snow, etc.), though concentrations of constituents are reported in more natural concentration units of mg/L. Advective fluxes between all water storage compartments are calculated as

$$J = M \cdot \left(\frac{m}{\phi}\right)$$

where J is the advective flux [mg/m^2/d], M is the water exchange rate between compartments [mm/d], m is the constituent mass [mg/m^2], ϕ is the water storage of the compartment which the mass is leaving [mm]. In any of the storage compartments, constituent concentration is calculated as

$$C = \frac{m}{\phi}$$

With the `ORDERED_SERIES` global numerical algorithm, mass balance errors for each constituent should be exactly zero. Because the transport module wraps around the hydrologic water balance model, the addition of new hydrologic processes and algorithms does not require the addition of new code for simulating mass transport.

For flow tracers, the option may be used to ignore the inherent units of mass density, and instead track the percent of flows sourced from particular sources. This can be useful, for example, in tracking snow vs. rain components of streamflow, or determining the timing of outflow coming from a given HRU. In the case of a tracer, the same expression as above is valid, though using an equivalent flux and equivalent mass, i.e.,

$$J' = M \cdot \left((\frac{m'}{\phi}\right)$$

where J' is the advective flux [mm/d], and m' is the effective mass [mm]. In this case, J'/M may be interpreted as the fraction of the flow which contains the tracer fluid; likewise, m'/ϕ, the tracer concentration [unitless] can be interpreted as the fraction of storage which is marked by tracer. Tracer concentrations should range from 0 to 1.

The primary outputs from the transport simulation are the average concentrations of a given constituent in each of the various storage compartments and pollutographs at subbasin outlets.

7.1 Constituent Sources

Sources of constituents may be handled in one of two ways:

- As Dirichlet conditions, where the constituent concentration in a given compartment is fixed at a user-specified value
- As Neumann conditions, where a user-specified (dry) mass flux is applied to a given compartment

Other source types may be incorporated into RAVEN at a later date.

7.2 Catchment Routing

Constituents are routed through the catchment in a manner consistent with the catchment routing process described in section 5.1. A discrete transfer function approach is used,

$$QC(t+\Delta t) = \sum_{n=0}^{N} QC_{lat}(t - n\Delta t) \cdot UH_n \tag{7.1}$$

where QC [mg/d] is the mass loading, QC_{lat} is the loading released from the catchment at time t, and $\vec{UH}$ is a unitless vector which describes the distribution of arrival times to the channel, and is the same distribution used by the catchment routing for water, described in section 5.1.

7.3 In-channel Routing

To do (6)

Chapter 8

Model Diagnostics

While Raven doesn't have built-in calibration functionality, it supports it's own assessment by internally comparing observation data to model output. The model diagnostic output can readily be used by model-independent optimization and parameter estimation tools (as briefly discussed in section 2.5). This chapter includes information about all of the available diagnostics.

8.1 Pointwise vs. Pulsewise comparison

Note that in all cases, RAVEN is comparing a time series of observations to a time series of model output. It is assumed that the observations are instantaneous observations at a point in time (e.g., a single soil moisture measurement or snow depth measurement). The key exception to this is observed hydrographs. Most observed hydrographs available from government or municipal agencies report averaged data over discrete time intervals, e.g., daily average flows. RAVEN is careful to treat this continuous data as is appropriate, and compares the modeled average flows over each time interval to the observed average flows.

For non-hydrograph data, the model output is interpolated to the exact time of observation.

The documentation for the relevant .rvi and .rvt input commands (`:ObservationData` `:ObservationWeights` and `:EvaluateMetrics`) can be found in appendix A.

8.2 Diagnostic Algorithms

In all of the algorithms below, ϕ_i is an observation of interest, $\hat{\phi}_i$ is the corresponding modeled value, w_i is the corresponding weight of the observation (1.0 by default, 0 for blank observation data) and N is the number of non-blank observations. Note that many of these diagnostics are useful for hydrographs but may not make particular sense for other observed state variables (even though we can calculate them anyhow).

8.2.1 Nash-Sutcliffe Efficiency (`NASH_SUTCLIFFE`)

$$\text{NS} = 1 - \frac{\sum_{i=1}^{N} w_i \left(\hat{\phi}_i - \phi_i\right)^2}{\sum_{i=1}^{N} w_i \left(\bar{\phi} - \phi_i\right)^2}$$

where $\bar{\phi}$ is the weighted mean of observations,

$$\bar{\phi} = \frac{1}{N}\sum_{i=1}^{N} w_i \phi_i$$

8.2.2 Log-transformed Nash-Sutcliffe Efficiency (`LOG_NASH`)

$$\text{NS} = 1 - \frac{\sum_{i=1}^{N} w_i \left(\ln(\hat{\phi}_i) - \ln(\phi_i)\right)^2}{\sum_{i=1}^{N} w_i \left(\ln(\bar{\phi}) - \ln(\phi_i)\right)^2}$$

where $\bar{\phi}$ is the weighted mean of observations,

$$\bar{\phi} = \frac{1}{N}\sum_{i=1}^{N} w_i \phi_i$$

8.2.3 Root-mean-squared Error (`RMSE`)

$$\text{RMSE} = \sqrt{\sum_{i=1}^{N} w_i \left(\hat{\phi}_i - \phi_i\right)^2}$$

8.2.4 Percentage Bias (`PCT_BIAS`)

Returns the percent bias. Non-zero weights have no effect on this calculation, but zero weights will force the corresponding data points to be ignored.

$$\text{PCT\_BIAS} = \frac{\sum_{i=1}^{N} \left(\hat{\phi}_i - \phi_i\right)}{\sum_{i=1}^{N} (\phi_i)}$$

8.2.5 Average Absolute Error (`ABSERR`)

Returns the weighted average absolute error.

$$\text{ABSERR} = \frac{1}{N}\sum_{i=1}^{N} w_i \left|\hat{\phi}_i - \phi_i\right|$$

8.2.6 Maximum Absolute Error (ABSMAX)

The maximum absolute error between observed and modeled data. Non-zero weights have no effect on this calculation, but zero weights will force the corresponding data points to be ignored.

$$\text{ABSMAX} = \max\left\{\left|\hat{\phi}_i - \phi_i\right|\right\}$$

8.2.7 Peak difference (PDIFF)

The difference between the peak modeled data and peak observed data. Non-zero weights have no effect on this calculation, but zero weights will force the corresponding data points to be ignored.

$$\text{PDIFF} = \max\left\{\hat{\phi}_i\right\} - \max\left\{\phi_i\right\}$$

8.2.8 Monthly Mean Squared Error (TMVOL)

Describes the total monthly mean error between modeled data and observed data.

$$\text{TMVOL} = \sum_{j=1}^{M}\left[\frac{1}{N}\sum_{i=1}^{N_j} w_i\left(\hat{\phi}_i - \phi_i\right)^2\right]$$

where M is the number of months in the simulation and N_j is the number of data points in month j.

8.2.9 Correlation of Error (RCOEF)

Describes the correlation of error between adjacent time steps. It represents the tendency for the error to remain constant from one time step to the next and should only be applied to continuous time series.

$$\text{RCOEF} = \frac{1}{\sigma_{\phi}\sigma_{\hat{\phi}}}\frac{1}{N^* - 1}\sum_{i=1}^{N-1}(\hat{\phi}_i - \phi_i)(\hat{\phi}_{i+1} - \phi_{i+1})$$

where σ_{ϕ} is the standard deviation of the observed data and $\sigma_{\hat{\phi}}$ is the standard deviation of the modeled data. N^* is the number of adjacent non-blank data entries. Non-zero observation weights are ignored.

8.2.10 Number of Sign Changes (NSC)

NSC describes the number of sign changes in the error from one data point to the next. A low NSC (as compared to the total number of data points) would imply that the modeled values are constantly above or below the observed values.

8.2.11 Kling Gupta Efficiency (KLING_GUPTA)

Kling-Gupta efficiency metric as defined in Gupta et al. (2009).

Appendix A

Input Files

A.1 Primary Input file (.rvi)

The primary input file stores the model simulation options and numerical options. An example .rvi file is shown below.

Example File: modelname.rvi

```
* -------------------------------------------
* Raven Input (.rvi) file
* -------------------------------------------
:StartDate       2000-01-01 00:00:00
:Duration        366.0
:Method          EULER
:TimeStep        1.0
* -Options-----------------------------------
:Routing         ROUTE_MUSKINGUM
:CatchmentRoute  ROUTE_GAMMA_CONVOLUTION
:Evaporation     PET_PENMAN_MONTEITH
:SoilModel       SOIL_TWO_LAYER
* -Processes---------------------------------
:HydrologicalProcesses
  :Precipitation   PRECIP_RAVEN     ATMOS_PRECIP    MULTIPLE
  :Infiltration    INF_GREEN_AMPT   PONDED_WATER    SOIL[0]
  :SoilEvaporation SOILEVAP_SEQUEN  SOIL[0]         ATMOSPHERE
  :Percolation     PERC_POWER_LAW   SOIL[0]         SOIL[1]
  :Percolation     PERC_POWER_LAW   SOIL[1]         GROUNDWATER
  :Baseflow        BASE_LINEAR      SOIL[1]         SURFACE_WATER
:EndHydrologicalProcesses
* -Custom Output-----------------------------
:CustomOutput Daily   Average SOIL[0]    BY_HRU
:CustomOutput Monthly Maximum SOIL[1]    BY_BASIN
```

Note that comments may be included on individual lines using the * or # characters as the first word on the line.

A.1.1 Required Commands

The .rvi file consists of the following required commands:

- `:StartDate [yyyy-mm-dd hh:mm:ss]`
  (Required) Starting time of the simulation.
- `:Duration [days]`
  (Required) Duration of the simulation, in decimal days, beginning from the start date specified.
- `:Method [method string]`
  (Optional) Numerical method used for simulation. Can be one of the following strings:
  - `ORDERED_SERIES` (**default**) - *Process ordering is defined as being the same as the order of hydrological process in the input file*
  - `EULER` - *uses the classical Euler method, with operator-splitting. Process order as specified in the input file does not matter*
- `:TimeStep [time step in days]`
  (Required) Time step for the simulation. As Raven is intended for sub-daily calculations, it is suggested that the time step be less than or equal to 1.0.
- `:TimeStep [hh:mm:ss]`
  Time step for the simulation (alternate format).
- `:SoilModel [soilmodel string] {optional other_data}`
  (Required) Soil model used in the simulation, one of the following:
  - `SOIL_ONE_LAYER` *Single soil layer*
  - `SOIL_TWO_LAYER` *Two soil layers*
  - `SOIL_MULTILAYER [number of layers]` *Any number of soil layers*
- `:HydrologicalProcesses-:EndHydrologicalProcesses`
  (Required) These commands bracket the list of hydrological processes to be modeled (see section A.1.5)

A.1.2 Model Operational Options

The following section discusses about the several hydrological processes that are supported by Raven and their respective algorithms. Some of these algorithms require specific parameters to be entered by the users. Refer to section A.1.3 for more details about the required parameters.

- `:CatchmentRoute [approach string]`
  Catchment routing method, used to convey water from the catchment tributaries and rivulets to the subbasin outlets. Can be one of the following methods, discussed in section **??**:
  - `DUMP` (**default**) - *water from the catchment is dumped directly to the basin outlet.*
  - `ROUTE_GAMMA_CONVOLUTION` - *a Gamma distribution is used to represent the unit hydrograph*
  - `ROUTE_TRI_CONVOLUTION` - *a triangular distribution is used for the unit hydrograph*
  - `ROUTE_RESERVOIRS_SERIES` - *series of linear reservoirs (Nash Hydrograph)*
- `:Routing [approach string]`
  Channel routing method which is used to transport water from upstream to downstream *within* the main subbasin channels. Can be one of the following methods, as described in section 5.2:
  - `ROUTE_NONE` - *water is not routed from subbasin to subbasin. Intended for single-subbasin/single catchment models or numerical testing only.*
  - `STORAGE_COEFF` (**default**) - *From Williams (1969)*
  - `ROUTE_PLUG_FLOW` - *water travels as a pulse of uniform celerity along the reach*
  - `ROUTE_MUSKINGUM` - *reach storage is updated using the Muskingum-Cunge routing algorithm*
  - `ROUTE_DIFFUSIVE_WAVE`
  - `ROUTE_HYDROLOGIC`
- `:InterpolationMethod [method]`
  (Optional) Means of interpolating forcing function data (e.g., precipitation, PET, etc.) between monitoring gauges. The centroid of the HRU is used as the interpolation point. The following methods, discussed in section 6.1 are supported:
  - `INTERP_NEAREST_NEIGHBOR` (**default**) - *the nearest neighbor (Voronoi) method*
  - `INTERP_INVERSE_DISTANCE` - *inverse distance weighting*
  - `INTERP_AVERAGE_ALL` - *averages all specified gauge readings*
  - `INTERP_FROM_FILE` [filename]- *weights for each gauge at each HRU are specified in a tabular format, given by*

    ```
    [NG] [# of HRUs]
    {w_n1 w_n2 ... w_nNG} x {# of HRUs}
    ```

    where `NG` *is the number of gauges. The sum of the weights in each row (i.e., for each HRU) should be 1. It is assumed that the number of HRUs is the same as in the current model .rvh file; the orders are also assumed to be consistent.*

- `:RainSnowFraction [method]`
  (Optional) Means of partitioning precipitation into snow and rain, if these values are not specified as time series data. The following methods, discussed in detail in section 6.3.1, are supported:
  - `RAINSNOW_DINGMAN` (**default**)
  - `RAINSNOW_DATA` - *gauge or gridded time series of snowfall used*
  - `RAINSNOW_UBC`
  - `RAINSNOW_HBV`
  - `RAINSNOW_HSPF`
- `:Evaporation [method]`
  PET calculation method for land surface. Can be one of the following methods, described in detail in section 6.4:
  - `PET_CONSTANT`
  - `PET_PENMAN_MONTEITH`
  - `PET_PENMAN_COMBINATION`
  - `PET_PRIESTLEY_TAYLOR`
  - `PET_HARGREAVES`
  - `PET_HARGREAVES_1985` (**default**)
  - `PET_FROM_MONTHLY`
  - `PET_DATA` - *gauge or gridded time series used*
  - `PET_HAMON_1961`
  - `PET_TURC_1961`
  - `PET_MAKKINK_1957`
  - `PET_MONTHLY_FACTOR`
- `:OW_Evaporation [method]`
  (Optional) PET calculation method for open water. Has the same options as `Evaporation` command.
- `:OroPrecipCorrect [method]`
  (Optional) Method for correcting total precipitation for orographic (elevation) effects. The following methods, discussed in detail in section 6.3.2, are supported:
  - `OROCORR_NONE` (**default**)
  - `OROCORR_HBV`
  - `OROCORR_UBC`
  - `OROCORR_UBC_2`
  - `OROCORR_SIMPLE`

- `:OroTempCorrect [method]`
  (Optional) Method for correcting estimated Temperatures for orographic (elevation) effects. The following methods are supported:
  - `OROCORR_NONE` (**default**)
  - `OROCORR_HBV`
  - `OROCORR_UBC`
  - `OROCORR_UBC_2`
  - `OROCORR_SIMPLE`
- `:OroPETCorrect [method]`
  (Optional) Method for correcting estimated PET for orographic (elevation) effects. The following methods are supported, as discussed in section 6.3.2:
  - `OROCORR_NONE` (**default**)
  - `OROCORR_HBV`
  - `OROCORR_UBC`
  - `OROCORR_UBC_2`
  - `OROCORR_PRMS`

  Note: No specific parameter required for any of the methods mentioned above.
- `:SWRadiationMethod [method]`
  (Optional) Means of estimating shortwave radiation to the surface. The following methods, described in detail in section 6.5, are supported:
  - `SW_RAD_DEFAULT`(**default**) - *From Dingman (2002)*
  - `SW_RAD_DATA` - *gauge or gridded time series used*
  - `SW_RAD_UBCWM` - *From Quick (2003)*
- `:SWCanopyCorrect` (Optional) Means of correcting shortwave radiation to the surface due to canopy cover. The following methods, described in detail in section 6.5, are supported:
  - `SW_CANOPY_CORR_NONE`(**default**)
  - `SW_CANOPY_CORR_STATIC`
  - `SW_CANOPY_CORR_DYNAMIC`
  - `SW_CANOPY_CORR_UBC` - *From Quick (2003)*
- `:SWCloudCorrect` (Optional) Means of correcting shortwave radiation to the surface due to cloud cover. The following methods, described in detail in section 6.5, are supported:
  - `SW_CLOUDCOV_CORR_NONE`(**default**)
  - `SW_CLOUDCOV_CORR_DINGMAN`
  - `SW_CLOUDCOV_CORR_UBC` - *From Quick (2003)*
- `:LWRadiationMethod [method]`
  (Optional) Means of estimating longwave radiation. The following methods are supported, as discussed in section 6.6:

  - `LW_RAD_DATA` - *gauge or gridded time series used*
  - `LW_RAD_DEFAULT`(**default**) - *From Dingman (2002)*
  - `LW_RAD_UBC` - *From Quick (2003)*
  - `LW_RAD_HSPF`

- `:CloudCoverMethod [method]`
  (Optional) Means of calculating cloud cover percentages, if used. The following methods, as described in section 6.7, are supported:
  - `CLOUDCOV_NONE` (**default**)
  - `CLOUDCOV_DATA` - *gauge or gridded time series used*
  - `CLOUDCOV_UBC` - *From Quick (2003)*

- `:WindspeedMethod [method]`
  (Optional) Means of calculating wind speed at a reference height. The following methods are supported, as described in section 6.9.1:
  - `WINDVEL_CONSTANT` (**default**) - *constant wind velocity of 3 m/s*
  - `WINDVEL_DATA` - *gauge or gridded time series used*
  - `WINDVEL_UBC` - *From Quick (2003)*

- `:RelativeHumidityMethod [method]`
  (Optional) Means of calculating relative humidity. The following methods are supported, as described in section 6.9.2:
  - `RELHUM_CONSTANT` (**default**) - *constant relative humidity of 0.5*
  - `RELHUM_MINDEWPT`

  Note: No specific parameter required for any of the methods mentioned above.

- `:AirPressureMethod [method]`
  (Optional) Means of estimating air pressure. The following methods are supported, as described in section 6.9.3:
  - `AIRPRESS_BASIC` (**default**)
  - `AIRPRESS_CONST` - *standard atm. pressure at 20*°C
  - `AIRPRESS_DATA` - *gauge or gridded time series used*
  - `AIRPRESS_UBC` - *From Quick (2003)*

- `:PrecipIceptFract [method]`
  (Optional) Means of estimating the precipitation interception fraction (i.e., what percentage of precip is intercepted by the canopy). The following methods are supported, as described in section 4.1.1:
  - `PRECIP_ICEPT_USER` (**default**)
  - `PRECIP_ICEPT_LAI`
  - `PRECIP_ICEPT_EXPLAI`

- `:PotentialMelt [method]`
  (Optional) If used, estimates the potential melt. The following methods are supported , as discussed in section 6.8.1:
  - `POTMELT_DEGREE_DAY` (**default**)
  - `POTMELT_EB`
  - `POTMELT_RESTRICTED`
  - `POTMELT_UBC`
  - `POTMELT_HBV`
- `:MonthlyInterpolationMethod [method]`
  (Optional) If used, performs monthly interpolations. The following methods, as discussed in section 6.11, are supported:
  - `MONTHINT_UNIFORM`
  - `MONTHINT_LINEAR_MID` (**default**)
  - `MONTHINT_LINEAR_FOM`
  - `MONTHINT_LINEAR_21`

  Note: No specific parameter required for any of the methods mentioned above.
- `:SubDailyMethod [method]`
  (Optional) Used for sub-daily temporal downscaling of daily average PET and snowmelt. The supported methods are, as described in section 6.10:
  - `SUBDAILY_NONE` (**default**)
  - `SUBDAILY_UBC`
  - `SUBDAILY_SIMPLE`

  Note: No specific parameter required for any of the methods mentioned above.

A.1.3 Required Parameters for Model Operation Options

The following table (Table A.1) shows the required parameters in order to use the different Model Operation Options that were listed in the previous section (Section A.1.2).

Table A.1: Required Parameters for All Model Operation Options.
***=Default Algorithm**

OPTIONS	ALGORITMHS	REQUIRED PARAMETERS
Interpolation	INTERP_FROM_FILE	GaugeWeights Table required
	INTERP_AVERAGE_ALL	-
	INTERP_NEAREST_NEIGHBOR*	-
Routing	ROUTE_NONE	Channel Geometry and Manning's n
	ROUTE_DIFFUSIVE_WAVE*	Channel Geometry and Manning's n
	ROUTE_PLUG_FLOW	Channel Geometry and Manning's n
	ROUTE_STORAGE_COEFF	Channel Geometry and Manning's n
	ROUTE_MUSKINGUM	Channel Geometry and Manning's n
	ROUTE_MUSKINGUM_LAGGED	Channel Geometry and Manning's n
	ROUTE_MUSKINGUM_CUNGE	Channel Geometry and Manning's n
	ROUTE_HYDROLOGIC	Channel Geometry and Manning's n
CatchmentRoute	ROUTE_DUMP*	-
	ROUTE_LAG	TIME_LAG
	ROUTE_DELAYED_FIRST_ORDER	TIME_LAG and RES_CONSTANT
	ROUTE_GAMMA_CONVOLUTION	TIME_TO_PEAK
	ROUTE_TRI_CONVOLUTION	TIME_TO_PEAK and TIME_CONC
	ROUTE_RESERVOIR_SERIES	NUM_RESERVOIRS and RES_CONSTANT
	ROUTE_EXPONENTIAL	RES_CONSTANT
Evaporation and OW_Evaporation	PET_CONSTANT	-
	PET_FROMFILE	[gridded PET data or time series at gauge]
	PET_FROMMONTHLY	:MonthlyAveEvaporation and :MonthlyAveTemperature
	PET_MONTHLY_FACTOR	FOREST_PET_CORR, FOREST_COVERAGE and :MonthlyEvapFactor
	PET_PENMAN_MONTEITH	MAX_HEIGHT, RELATIVE_HT, MAX_LAI, RELATIVE_LAI, MAX_LEAF_COND and FOREST_SPARSENESS
	PET_PENMAN_COMBINATION	MAX_HEIGHT and RELATIVE_HT
	PET_HAMON	-
	PET_HARGREAVES	TEMP_MONTH_MAX and TEMP_MONTH_MIN
	PET_HARGREAVES_1985*	-
	PET_TURC_1961	-
	PET_MAKKINK_1957	-
	PET_PRIESTLEY_TAYLOR	-
OroPETCorrect	OROCORR_NONE*	-
	OROCORR_SIMPLELAPSE	-
	OROCORR_HBV	[hard coded for now]
	OROCORR_UBC & OROCORR_UB2	-
	OROCORR_PRMS	-
SWRadiationMethod	SW_RAD_DATA	[gridded data or time series at gauge]
	SW_RAD_DEFAULT*	SLOPE and ASPECT
	SW_RAD_UBCWM	HORIZON_CORR and TURBIDITY
LWRadiationMethod	LW_RAD_DATA	[gridded data or time series at gauge]
	LW_RAD_DEFAULT*	FOREST_COVERAGE
	LW_RAD_UBCWM	FOREST_COVERAGE
CloudCoverMethod	CLOUDCOV_NONE*	-

Continued on next page

Table A.1 – continued from previous page

OPTIONS	ALGORITMHS	REQUIRED PARAMETERS
	CLOUDCOV_DATA	[gridded data or time series at gauge]
	CLOUDCOV_UBC	-
RainSnowFraction	RAINSNOW_DATA	[gridded data or time series at gauge]
	RAINSNOW_DINGMAN	RAINSNOW_TEMP
	RAINSNOW_HBV	RAINSNOW_TEMP and RAINSNOW_DELTA
	RAINSNOW_UBC	RAINSNOW_TEMP and RAINSNOW_DELTA
PrecipIceptFract	PRECIP_ICEPT_USER	RAIN_ICEPT_PCT and SNOW_ICEPT_PCT
	PRECIP_ICEPT_LAI	RAIN_ICEPT_FACT and SNOW_ICEPT_FACT
	PRECIP_ICEPT_EXPLAI	-
	PRECIP_ICEPT_HEDSTROM	-
OroPrecipCorrect	OROCORR_NONE*	-
	OROCORR_UBC	:UBCPrecipLapseRates
	OROCORR_HBV	:RainCorrection and :SnowCorrection
	OROCORR_SIMPLELAPSE	-
OroTempCorrect	OROCORR_NONE*	-
	OROCORR_UBC	:UBCPrecipLapseRates
	OROCORR_HBV	ADIABATIC_LAPSE
	OROCORR_SIMPLELAPSE	ADIABATIC_LAPSE
PotentialMeltMethod	POTMELT_DEGREE_DAY*	MELT_FACTOR
	POTMELT_RESTRICTED	MELT_FACTOR
	POTMELT_HBV	MIN_MELT_FACTOR, HBV_MELT_ASP_CORR, HBV_MELT_FOR_CORR and FOREST_COVERAGE
	POTMELT_UBC	MIN_SNOW_ALBEDO, FOREST_COVERAGE and ASPECT, :UBCNorthSWCorr, :UBCSouthSWCorr and FOERGY,
SubDailyMethod	SUBDAILY_NONE*	-
	SUBDAILY_SIMPLE	-
	SUBDAILY_UBC	-
WindspeedMethod	WINDVEL_CONSTANT*	-
	WINDVEL_DATA	[time series at gauge]
	WINDVEL_UBC	:UBCTempLapseRates and FOREST_COVERAGE
RelativeHumidityMethod	RELHUM_CONSTANT*	-
	RELHUM_DATA	-
	RELHUM_MINDEWPT	-
AirPressureMethod	AIRPRESS_BASIC*	-
	AIRPRESS_UBC	-
	AIRPRESS_DATA	[gridded data or time series at gauge]
	AIRPRESS_CONST	-
MonthlyInterpolationMethod	MONTHINT_UNIFORM	-
	MONTHINT_LINEAR_FOM	-
	MONTHINT_LINEAR_MID*	-
	MONTHINT_LINEAR_21	-

To do (7)

A.1.4 Input/Output Control Commands

- `:RunName [name]`
  (Optional) The name of the model run. This acts as a prefix to all output files generated by the program. The default is no run name, and no prefix is appended to the file outputs.
- `:rvh_Filename [name]`
  (Optional) The name of the *.rvh file. By default, the .rvh file has the same name as the .rvi file; this command allows the user to override this. If no directory is specified, it is assumed the file exists in the working directory. Equivalent to the command prompt argument -h [name].
- `:rvc_Filename, :rvp_Filename, :rvt_Filename`
  (Optional) Same as `:rvh_Filename [name]` above, but for .rvc,.rvp, and .rvt files, respectively
- `:OutputDirectory [directory name]`
  (Optional) Sets the output directory, which by default is the working directory from which the executable is called. Directory name is usually in a system independent format, using all forward slashes for folders, ending with a forward slash, e.g., C:/Temp/Model Output/run 3/. Equivalent to the command line argument -o [directory name]. If used, this should be called as early as possible in the .rvi file.
- `:CreateRVPTemplate`
  (Optional) Produces a template .rvp file in the same directory as the .rvi file based upon the hydrological process list and model options in the .rvi file, so the user knows what parameters need to be specified for the given model configuration. NOTE: this turns off model operation, only the template file will be created.
- `:OutputInterval [frequency]`
  The frequency of printing output to the output files. Default of 1 (printing every timestep). Typically used for simulations with small timesteps (e.g., if frequency=60 for a model with a timestep of 1 minute, output is printed hourly).
- `:WriteMassBalanceFile`
  (Optional) The file runname_WatershedMassEnergyBalance.csv (or .tb0) is generated (see appendix B)
- `:WriteForcingFunctions`
  (Optional) The file runname_ForcingFunctions.csv (or .tb0) is generated (see appendix B)
- `:WriteEnergyStorage`
  (Optional) The file runname_WatershedEnergyStorage.csv is generated (see appendix B)
- `:WriteParametersFile`
  (Optional) The file runname_WatershedEnergyStorage.csv is generated (see appendix B)
- `:WriteEnsimFormat [yes or no]`
  (Optional) Specify whether the output files generated by Raven should be in an EnSim format (e.g. tb0, ts3, etc.) or standard text files (.csv). String values can be one of:
  - `yes` - file output in Ensim formats, or
  - `no` - file output in standard formats

  The default is standard format

- `:WriteExhaustiveMB`
  (Optional) The file runname_ExhaustiveMB.csv is generated (see appendix B

- `:EndPause [yes or no]`
  (Optional) if `:EndPause` is set to 'yes' then the program output will stay on the screen (e.g., as a DOS window) until the user exits manually.

- `:DebugMode [yes or no]`
  (Optional) If set to 'yes', the equivalent of including `:WriteMassBalanceFile`, `:WriteForcingFunctions`, `:WriteEnergyStorage`, and `:WriteParameters`.

- `:SilentMode`
  (Optional) If the SilentMode command is included, output to the command prompt is minimized.

- `:SuppressOutput`
  (Optional) Suppresses all standard output, incuding creation of Hydrographs, transport output, and watershed storage files. Does not turn off optional outputs which were requested elsewhere in in the input file. Does not turn off creation of diagnostics.csv.

- `:WaterYearStartMonth [integer month]`
  (Optional) Changes the start of the water year from October 1st (the default) to the 1st of another month (for example, 7=July for Australian application). The water year is only used for reporting of annual (`WATERYEARLY`) budget reporting in the `:CustomOutput` command.

- `:CustomOutput [time_per] [processing] [variable/parameter] [space_aggregation] filename`
  (Optional) This command is used to create a custom output file that tracks a single variable, parameter, or forcing function over time at a number of basins, HRUs, or across the watershed. Here, the `variable` is specified using either the state variable name(for an exhaustive list, see table C.1), the forcing name (see table C.2), or parameter name. `time_per` refers to the time period, one of:

  - `DAILY`
  - `MONTHLY`
  - `YEARLY`
  - `WATER_YEARLY`
  - `CONTINUOUS` (for output created every time step)

  For the water year aggregation, a default water year of October 1-September 30 is used. The start month can be changed using the `:WaterYearStartMonth` command above. `processing` is the statistic reported over each time interval, one of:

  - `AVERAGE`
  - `MAXIMUM`
  - `MINIMUM`
  - `RANGE`
  - `MEDIAN`
  - `QUARTILES`

  - `HISTOGRAM [min] [max] [num. of bins]`

  If `HISTOGRAM` is selected, the command should be followed (in the same line) with the minimum and maximum bounding values of the histogram range and the number of evenly spaced bins.

  `space_aggregation` refers to the evaluation domain, and is either `BY_BASIN`, `BY_HRU`, `BY_HRU_GROUPS`, or `ENTIRE_WATERSHED`.

  If the state variable is not used in the model (it does not participate in any of the user-specified hydrologic processes), the output file will not be created; a warning will be generated.

  As an example, the custom output command may be used as follows:

  ```
  :CustomOutput DAILY MAXIMUM SNOW BY_BASIN
  ```

  This would create the file `runname_DailyMaximumSnowByBasin.csv`, which would include a time series of daily maximum snow contents (as mm SWE) for all subbasins in the model. An optional specified filename may be appended to the end of any command to override the default filename.

- `:LakeStorage [lake storage variable]`
  (Optional) Specifies variable to be used for rainfall on lake HRUs, typically `SURFACE_WATER` (default) or `LAKE_STORAGE`

- `:OutputDump [timestamp (YYY-MM-DD hh:mm:ss)]` Outputs snapshot of all state variables to file state_(timestamp).rvc. Format is the same as solution.rvc. This can later be used as an initial condition file. Multiple calls to this command will cause snapshots to be written at all requested dump times.

- `:SnapshotHydrograph` Hydrographs are reported using the values at the end of each time step. By default, hydrographs are reported as averaged over the time step, to be consistent with observation data, typically reported using time-averaged values.

- `:EvaluationMetrics [metric1] metric2 metric3 ... metricN` If observation time series are provided (see `:ObservationData` command in appendix A.4.2), RAVEN will generate the evaluation metrics listed in this command. The metrics include:

  - `NASH_SUTCLIFFE`
  - `RMSE`
  - `PCT_BIAS`
  - `ABSERR`
  - `ABSMAX`
  - `PDIFF`
  - `TMVOL`
  - `RCOEF`
  - `NSC`
  - `RSR`
  - `R2`
  - `LOG_NASH`

– KLING_GUPTA

These metrics are defined in section 8.2.

A.1.5 Hydrologic Processes

In addition to the above commands, the .rvi file must include the list of all of the necessary hydrological processes to be included in the model, which are bracketed by the `:HydrologicalProcesses` and `:EndHydrologicalProcesses` commands. The process commands are typically in some variation of the following format:

- `:ProcessName ProcessAlgorithm {ProcessFrom} {ProcessTo}`

Where `:ProcessName` is the name of the process (e.g., :CanopyDrip), `ProcessAlgorithm` refers to the particular algorithm used for simulation (e.g., `RUTTER` corresponds to the (Rutter et al., 1971) model for loss of water from canopy to ground surface), and `ProcessFrom` and `ProcessTo` are the state variable code for the source and sink storage compartments, which are selected from the complete list of state variables in table .

The state variables `SURFACE_WATER`, `PONDED_WATER`, `ATMOS_PRECIP` and `ATMOSPHERE` are automatically included in all models. The others will be dynamically included in the model as processes are added. For example, the `SNOW` variable will be automatically added if a snowmelt or sublimation hydrological process is added to the list. Note that the computational cost of a model is directly related to the number of state variables and number of processes included in that model. Note that the `SOIL` variable is followed by the index of the soil layers in the model, with `[0]` corresponding to the topmost layer. The `MULTIPLE` tag is a placeholder, indicating that there are more than one compartments either receiving water/energy/mass, or more than one losing. The specific compartments are determined from the chosen algorithm.

Important: depending upon the numerical method chosen, the ordering of the processes in the input file may determine the accuracy and/or behavior of the solution.

Table A.2 includes a detailed description of the process commands available in Raven.

Table A.2: Hydrologic Process Commands for the .rvi file. Compartments with an asterisk must be specified within the command (placeholder `[T]`-'To' or `[F]`-'From').

Process Command	Algorithms	Valid "From" Compartments	Valid "To" Compartments
`:Precipitation` section 4.1	`PRECIP_RAVEN`	`ATMOS_PRECIP`	`MULTIPLE`

`:Infiltration [F]`
section 4.2
	`INF_PARTITION_COEFF`
	`INF_SCS`
	`INF_GREEN_AMPT`
	`INF_GA_SIMPLE`
	`INF_VIC`
	`INF_VIC_ARNO`
	`INF_HBV`
	`INF_PRMS`
	`INF_UBC`
	`INF_GR4J`
		`PONDED_WATER`	`SURFACE_WATER`*
			`SOIL[0]`*
			`SOIL[m]`*

`:Baseflow [F]`
section 4.3
	`BASE_LINEAR`
	`BASE_VIC`
		`SOIL[m]`*
		`AQUIFER`*
			`SURFACE_WATER`

Continued on next page

Table A.2 – continued from previous page

Process Command	Algorithms	Valid "From" Compartments	Valid "To" Compartments
	BASE_POWER_LAW BASE_TOPMODEL BASE_SAC BASE_CONSTANT BASE_THRESH_POWER		
:Percolation [F] [T] section 4.4	PERC_CONSTANT PERC_GAWSER PERC_POWER_LAW PERC_PRMS PERC_SACRAMENTO PERC_GR4JEXCH PERC_GR4JEXCH2	SOIL[m]*	SOIL[m]*
:Interflow [F] section 4.5	INTERFLOW_PRMS	SOIL[m]*	SURFACE_WATER
:SoilEvaporation section 4.6	SOILEVAP_VIC SOILEVAP_HBV SOILEVAP_TOPMODEL SOILEVAP_ROOT SOILEVAP_SEQUEN SOILEVAP_GR4J	SOIL[0] SOIL[m]	ATMOSPHERE
:CapillaryRise [F] [T] section 4.7	CRISE_HBV	SOIL[m]*	SOIL[m]*
:CanopyEvap section 4.8	CANEVAP_MAXIMUM CANEVAP_ALL CANEVAP_RUTTER	CANOPY	ATMOSPHERE
:CanopyDrip section 4.9	CANEVAP_RUTTER CANEVAP_SLOWDRAIN	CANOPY	PONDED_WATER
:SnowBalance section 4.12	SNOBAL_SIMPLE_MELT SNOBAL_HBV SNOBAL_UBCWM SNOBAL_CEMA_NIEGE SNOBAL_TWO_LAYER	MULTIPLE	MULTIPLE
:SnowMelt [T]	MELT_SIMPLE	SNOW	PONDED_WATER* SNOW_LIQ*
:Sublimation section 4.13	SUBLIM_KUZMIN SUBLIM_CENTRAL_SIERRA	SNOW	ATMOSPHERE
:SnowAlbedoEvolve section 4.16	SNOALB_UBC	SNOW_ALBEDO	SNOW_ALBEDO
:GlacialMelt section 4.17	GMELT_DEGREE_DAY GMELT_UBC	GLACIER_ICE GLACIER_CC	GLACIER GLACIER_CC
:GlacierRelease section 4.18	GRELEASE_LINEAR_STORAGE GRELEASE_HBV_EC	GLACIER	SURFACE_WATER
:OpenWaterEvaporation	OPEN_WATER_EVAP	PONDED_WATER DEPRESSION	ATMOSPHERE
:Flush section 4.20	N/A	any	any
:Overflow section 4.20	N/A	any	any

To do (8)

Note that application of any given process algorithm can be made conditional using the `:-->Conditional` command immediately after the process command. For example,

```
:Flush             PONDED_WATER   SURFACE_WATER
:-->Conditional    HRU_TYPE       IS_NOT GLACIER
:Flush             PONDED_WATER   GLACIER
:-->Conditional    HRU_TYPE       IS GLACIER
```

The above input file snippet moves ponded water to surface water, unless the HRU type is a glacier (as defined by its soil profile properties). Currently, the conditional command supports

- conditionals based upon HRU type (`HRU_TYPE`), where the type is one of `{GLACIER,LAKE,ROCK,STANDARD}`)
- conditionals based upon land use type, e.g., `:-->Conditional LAND_USE IS WETLAND` where `LAND_USE` names are as defined in the `:LandUseClasses` command in the .rvp file
- conditionals based upon HRU group, e.g., `:-->Conditional HRU_GROUP IS_NOT BURNED_FOREST` where the `HRU_GROUP`s are defined using the `:HRUGroup` command in the .rvh file.

The only available comparison operators are `IS` and `IS_NOT`.

To do (9)

A.1.6 Transport Commands

- `:Transport [constituent_name] {units}`
  (Optional) This command declares a new transport constituent named `constituent_name` which can be advected through the system. The optional units command should be either mg/l or none (for tracers).
- `:FixedConcentration [constituent_name] [storage_compartment] [concentration] {HRU_group}`
  (Optional) This command specifies a type one boundary condition in all water storage compartments of type `storage_compartment` (taken from the state variable list of ??) in HRU group `HRU_group`. All water passing through this storage compartment will be assigned the specified concentration. Note that the constituent name needs to be specified using the `:Transport` command prior to calling this command. If the optional `HRU_group` is omitted, then the condition applies to all storage compartments of this type throughout the watershed. For tracers, it is useful to specify a concentration of 1 (no units).

A.2 Classed Parameter Input file (.rvp)

The classed parameter input file stores a database of soil, vegetation, river, aquifer, and land class properties. Not all classes specified in the *.rvp file need to be included in the model. An example .rvp file is shown below.

Example File: modelname.rvp

```
* -------------------------------------------
* Raven Classed Parameter File
* -------------------------------------------
:SoilClasses
  :Attributes, %SAND,  %CLAY, %SILT, %ORGANIC
  :Units,       none,   none,  none,     none
   SAND,           1,      0,     0,        0
   LOAM,         0.5,    0.1,   0.4,      0.4
:EndSoilClasses
:SoilProfiles
*     name, #horizons, hor1, th1, hor2, th2
      LAKE,         0
   GLACIER,         0
  LOAM_SEQ,         2, LOAM, 0.5, SAND, 1.5
  ALL_SAND,         1, SAND, 2.0
:EndSoilProfiles
:VegetationClasses
  :Attributes,       MAX_HT,       MAX_LAI,      MAX_LEAF_COND
  :Units,                m,          none,           mm_per_s
  CONIFER_FOREST,       25,           6.0,                5.3
       BROADLEAF,       25,           5.0,                5.3
:EndVegetationClasses
:LandUseClasses
  :Attributes,  IMPERMEABLE_FRAC, FOREST_COVERAGE,
  :Units     ,             fract,           fract,
    GRASSLAND,                 0,               0,
     SUBURBAN,               0.3,             0.3,
:EndLandUseClasses
```

As with the *.rvi file, `*` or `#` denotes a comment.

A.2.1 Required Commands

-
  ```
  :SoilClasses
    :Attributes, %SAND,  %CLAY, %SILT, %ORGANIC
    :Units,       none,   none,  none,   none
    {soil_class_name,%sand,%clay,%silt,%organic}x[NSC]
  :EndSoilClasses
  ```

  or

  ```
  :SoilClasses
    {soil_class_name}x[NSC]
  :EndSoilClasses
  ```

Defines each soil class and (optionally) specifies the mineral and organic composition of the soil which can be used to automatically generate some physical properties such as porosity or hydraulic conductivity. These parameters are defined as follows:

  - `soil_class_name` is the code (less than 30 characters) used to identify the soil class, within the code, the .rvp file and in the .rvh file, discussed below. The name may not contain spaces or special characters.
  - `%SAND,%CLAY,%SILT,%ORGANIC` [0..1] are the percent sand, clay, and organic matter of the soil, expressed in decimal form, between 0 and 1. The sand, silt, and clay fractions refer to the non-organic component of the soil, i.e., specifying %SAND=0.5, %CLAY=0.3, %SILT=0.2, %ORGANIC=0.1 indicates a soil composition of 45% sand, 27% clay, 18%silt, and 10% organic matter. The sum of the mineral components (%SAND, %CLAY, and %SILT) must be 1.

With soil information provided, Raven can autogenerate many other physically-based (i.e., measurable) soil properties such as hydraulic and thermal conductivities, wilting pressure, etc. To override these autogenerated parameters or to specify other soil parameters, an additional command (`:SoilParameterList`), described below, may optionally be added to the input file *after* the `SoilProperties` command has been called. For conceptual (i.e., box) models, the soil composition should generally not be specified.

-
  ```
  :SoilProfiles
    {profile_name,#horizons,{soil_class_name,thickness}x{#horizons}}x[NP]
  :EndSoilProflles
  ```

  Defines all N_P stored soil profiles, which is a collection of soil horizons with known depth and thickness, each belonging to a soil class. The soils should be specified from the top downward. Because `soil_class_name` is required, this command must come after the `SoilClasses` command. The thickness of each horizon is specified in meters.

  The special cases of lakes and glaciers (land surface elements with 'no' surface soils, or where it is not appropriate to simulate using soil infiltration and evaporation routines, are represented with the special profile names `LAKE`, `ROCK`, and `GLACIER`, all with zero horizons. ANY soil profile that starts with these terms is not subject to soil-based process algorithms.

-
  ```
  :VegetationClasses
    :Attributes, MAX_HT, MAX_LAI,MAX_LEAF_COND
    :Units,           m,    none,     mm_per_s
    {veg_class_name,MAX_CANOPY_HT,MAX_LAI,MAX_LEAF_COND}x[NVC]
  :EndVegetationClasses
  ```

  Defines the basic parameters for each vegetation class, which are used to optionally autogenerate many canopy and root properties. Here,

  - `veg_class_name` is the tag (less than 30 characters) used to identify the vegetation class, within the code, the .rvp file and in the .rvh file, discussed below.
  - `MAX_CANOPY_HT` [m] is the maximum canopy height reached during the year
  - `MAX_LAI` [m^2/m^2] is the maximum leaf area index (LAI) of the vegetation
  - `MAX_LEAF_COND` [mm/s] is the maximum leaf conductance of the vegetation

-
  ```
  :LandUseClasses
    :Attributes,  IMPERMEABLE_FRAC, FOREST_COVERAGE
  ```

```
  :Units     ,               fract,           fract
  {LU_class_name,IMPERMEABLE_FRAC,FOREST_COVERAGE}x[NLUC]
:EndLandUseClasses
```

Defines all N_{LU} land use/land type classes in the model. Land use is assumed to determine many of the surface roughness, albedo, and snow parameters. Here,

- `LU_class_name` is the tag (less than 30 characters) used to identify the land use class, within the code, the .rvp file and in the .rvh file, discussed below.
- `IMPERMEABLE_FRAC` [0..1] is the percentage of the land surface that is considered impermeable.
- `FOREST_COVERAGE` [0..1] is the percentage of the land surface that is covered with a vegetation canopy.

A.2.2 Optional Classes and Objects

Terrain classes and channel profiles do not need to be included in all models.

-
  ```
  :TerrainClasses
    :Attributes,  HILLSLOPE_LENGTH, DRAINAGE_DENSITY
    :Units     ,                 m,          km/km2
    {terrain_class_name, HILLSLOPE_LENGTH, DRAINAGE_DENSITY}x[NTC]
  :EndTerrainClasses
  ```

  Defines all N_{TC} physiographic terrain classes in the model, ranging from flat to hilly to steep and mountainous. Here,

  - `terrain_class_name` is the tag (less than 30 characters) used to identify the terrain class, within the code, the .rvp file and in the .rvh file, discussed below.
  - `HILLSLOPE_LENGTH` [m] is the representative hillslope length within the terrain
  - `DRAINAGE_DENSITY` [km/km^2] is the terrain drainage density

  If no terrain classes are specified, the tag `[NONE]` should be placed in the `:HRUs` command under terrain class.

-
  ```
  :ChannelProfile [channel_name]
    :Bedslope [slope]
    :SurveyPoints
      {[x]  [bed_elev]}x num survey points
    :EndSurveyPoints
    :RoughnessZones
      {[x_zone] [mannings_n]} x num roughness zones
    :EndRoughnessZones
  :EndChannelProfile
  ```

  Defines a channel profile with the unique name `channel_name`. The channel geometry is fully defined by a number of survey points (at least 2) along a transect. At the leftmost and rightmost points along the transect, it is assumed that the channel is bounded with infinitely steep sides. The x-coordinate system is arbitrary. In the same coordinate system, at least one zone with one Manning's n value must be specified. The coordinate x_{zone} is the leftmost boundary of the zone, and therefore the leftmost x_{zone} must be to the left of the leftmost (smallest) survey coordinate x. The channel configuration definitions are depicted in figure A.1. A representative bedslope is also needed: this is used to calculate flow rates using Manning's equation.

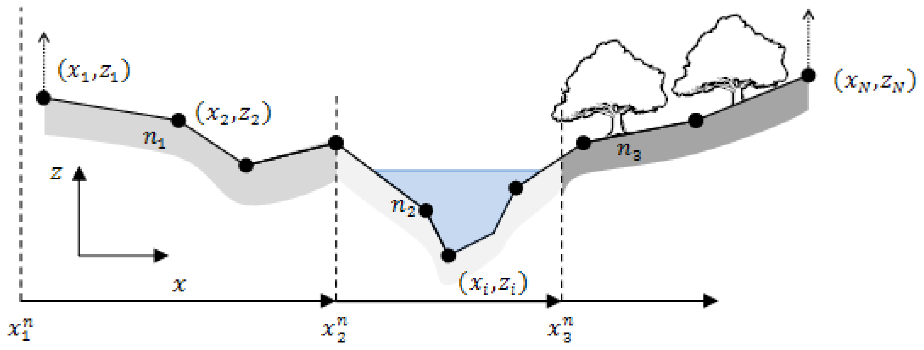


Figure A.1: Channel Profile definition. Each channel is defined by a cross sectional profile and a number of zones with different Manning's n values.

-
  ```
  :ChannelRatingCurves [channel_name]
    :Bedslope [slope]
    :StageRelationships
      {[stage] [area] [width] [flow]} x num curve points
    :EndStageRelationships
  :EndChannelRatingCurves
  ```

  Defines a channel profile with the unique name `channel_name`, and is used as an alternative to `:ChannelProfile`. Here, the stage-area, stage-top width, and stage-flow rating curves are explicitly provided. The first data point should correspond to stage and flow equal to zero, with all values entered with increasing stage. The units are stage [m], area [m^2], width [m], flow [m^3/s].

A.2.3 Parameter Specification

In addition to the required terms above, the following optional commands may be used to override autogeneration of parameters and specify parameters that cannot be autogenerated. If these are not included, either for an entire class or individual parameter, it is assumed that the parameter is to be autogenerated.

Soil Parameter Specification

- The following command is used to specify parameters linked to each soil class.

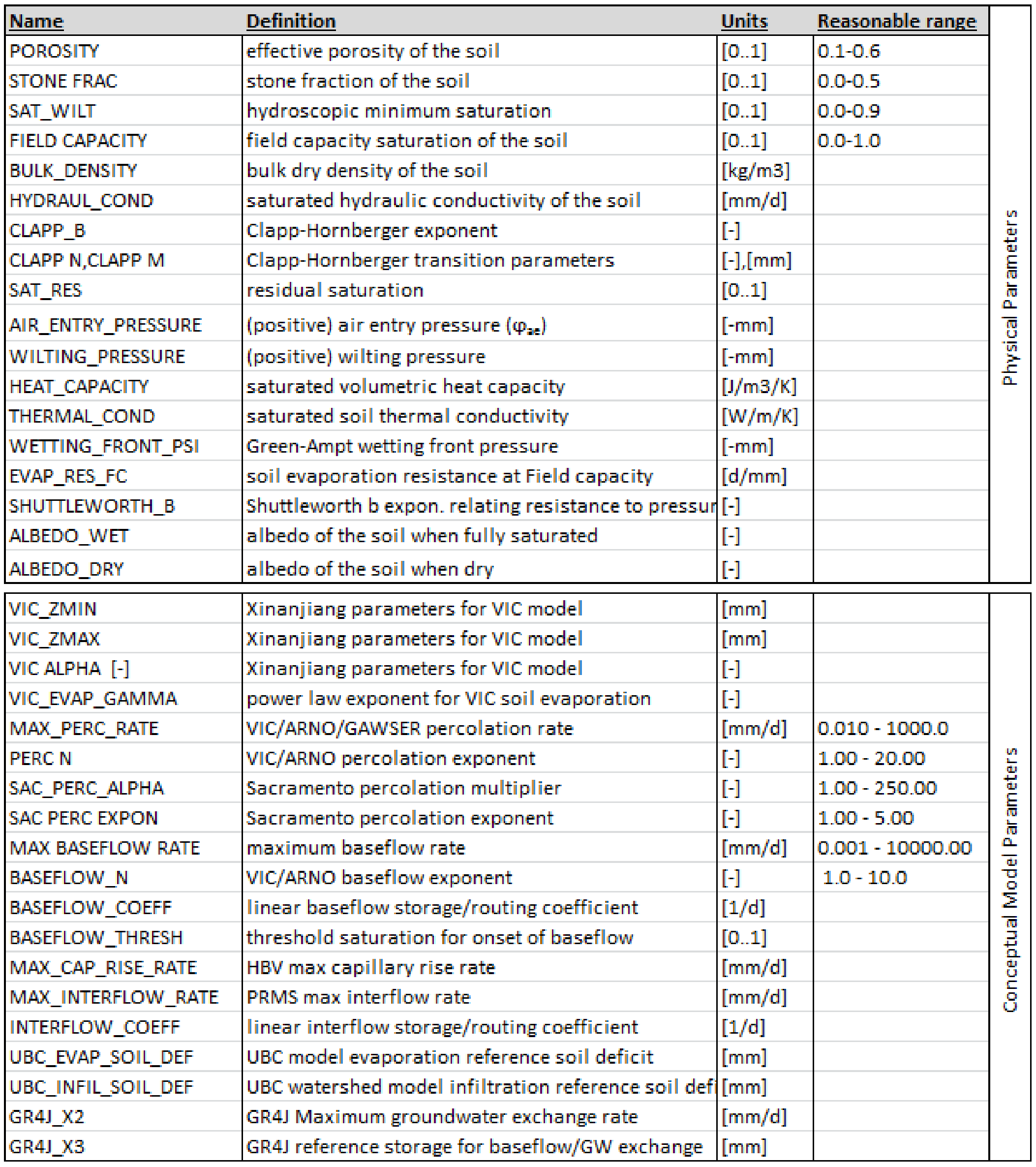
  ```
  :SoilParameterList
    :Parameters     , { param_name1, param_name1,...,  param_nameNP}
    :Units          , {  unit_type1,  unit_type2,...,   unit_typeNP}
    {[DEFAULT]      , {default_val1,default_val2,..., default_valNP} [optional]
    {soil_class_name, {  param_val1,  param_val2,...,   param_valNP}}x[<=NSC]
  :EndSoilProperties
  ```

  where, available soil parameter names (`param_name`) are described in the table A.2 and the soil class names (with the exception of the special `[DEFAULT]` tag) must already have been declared in the `:SoilClasses` command.

  The `[DEFAULT]` soil class name is used to specify parameter values for all classes not explicitly included as rows in the parameter list. Only soil classes which have parameters different from the default soil properties need to be specified in this list. If the user desires to autogenerate any of the parameters in the list (if RAVEN has the capacity to autogenerate these parameters), the `_AUTO` flag should be placed instead of a numerical value, as depicted in the example file. The `_DEFAULT` flag may be used if the default property (which can also be `_AUTO`) should be applied.

  Note that the units must be consistent with the native units of each parameter indicated in table A.2 - this line is intended for user interface processing and readability; **units will not be automatically converted if alternative unit specifiers are used.**

  While many watershed model and algorithm parameters have a physical basis (e.g., hydraulic conductivity), certain algorithms, particularly for lumped models, abstract a physical process so that coefficients in the relationships between storage and fluxes are completely artificial. These artificial parameters, which cannot be automatically generated based upon soil type, need to be specified directly by the user, and are often used as calibration (or 'tuning') parameters. These parameters are described in the second section of table A.2.



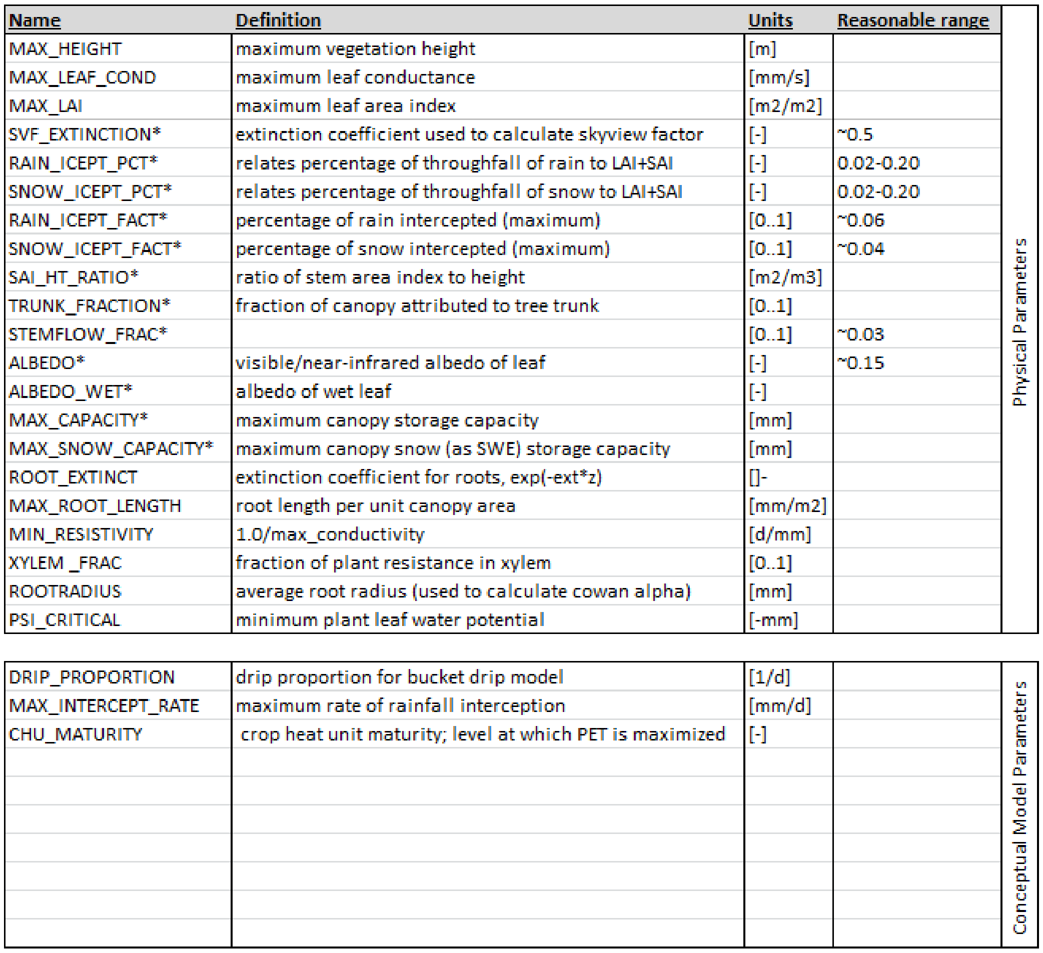
Name	Definition	Units	Reasonable range	
POROSITY	effective porosity of the soil	[0..1]	0.1-0.6	Physical Parameters
STONE FRAC	stone fraction of the soil	[0..1]	0.0-0.5	
SAT_WILT	hydroscopic minimum saturation	[0..1]	0.0-0.9	
FIELD CAPACITY	field capacity saturation of the soil	[0..1]	0.0-1.0	
BULK_DENSITY	bulk dry density of the soil	[kg/m3]		
HYDRAUL_COND	saturated hydraulic conductivity of the soil	[mm/d]		
CLAPP_B	Clapp-Hornberger exponent	[-]		
CLAPP N,CLAPP M	Clapp-Hornberger transition parameters	[-],[mm]		
SAT_RES	residual saturation	[0..1]		
AIR_ENTRY_PRESSURE	(positive) air entry pressure (φ_{ae})	[-mm]		
WILTING_PRESSURE	(positive) wilting pressure	[-mm]		
HEAT_CAPACITY	saturated volumetric heat capacity	[J/m3/K]		
THERMAL_COND	saturated soil thermal conductivity	[W/m/K]		
WETTING_FRONT_PSI	Green-Ampt wetting front pressure	[-mm]		
EVAP_RES_FC	soil evaporation resistance at Field capacity	[d/mm]		
SHUTTLEWORTH_B	Shuttleworth b expon. relating resistance to pressur	[-]		
ALBEDO_WET	albedo of the soil when fully saturated	[-]		
ALBEDO_DRY	albedo of the soil when dry	[-]		
VIC_ZMIN	Xinanjiang parameters for VIC model	[mm]		Conceptual Model Parameters
VIC_ZMAX	Xinanjiang parameters for VIC model	[mm]		
VIC ALPHA [-]	Xinanjiang parameters for VIC model	[-]		
VIC_EVAP_GAMMA	power law exponent for VIC soil evaporation	[-]		
MAX_PERC_RATE	VIC/ARNO/GAWSER percolation rate	[mm/d]	0.010 - 1000.0	
PERC N	VIC/ARNO percolation exponent	[-]	1.00 - 20.00	
SAC_PERC_ALPHA	Sacramento percolation multiplier	[-]	1.00 - 250.00	
SAC PERC EXPON	Sacramento percolation exponent	[-]	1.00 - 5.00	
MAX BASEFLOW RATE	maximum baseflow rate	[mm/d]	0.001 - 10000.00	
BASEFLOW_N	VIC/ARNO baseflow exponent	[-]	1.0 - 10.0	
BASEFLOW_COEFF	linear baseflow storage/routing coefficient	[1/d]		
BASEFLOW_THRESH	threshold saturation for onset of baseflow	[0..1]		
MAX_CAP_RISE_RATE	HBV max capillary rise rate	[mm/d]		
MAX_INTERFLOW_RATE	PRMS max interflow rate	[mm/d]		
INTERFLOW_COEFF	linear interflow storage/routing coefficient	[1/d]		
UBC_EVAP_SOIL_DEF	UBC model evaporation reference soil deficit	[mm]		
UBC_INFIL_SOIL_DEF	UBC watershed model infiltration reference soil defi	[mm]		
GR4J_X2	GR4J Maximum groundwater exchange rate	[mm/d]		
GR4J_X3	GR4J reference storage for baseflow/GW exchange	[mm]		

Figure A.2: Soil Parameters. The top section described autocalculable parameters which may be generated automatically using only the base soil class information (sand, clay, silt, and organic content). The bottom section must be user-specified.

Vegetation Parameter Specification

-
  ```
  :VegetationParameterList
    :Parameters     , { param_name1, param_name1,...,  param_nameNP}
    :Units          , {  unit_type1,  unit_type2,...,   unit_typeNP}
    [DEFAULT]       , {default_val1,default_val2,..., default_valNP} [optional]
    {VEG_CLASS_NAME , {  param_val1,  param_val2,...,   param_valNP}}x[<=NVC]
  :EndSoilProperties
  ```

  The `:VegetationParameterList` command operates in the same fashion as the `:SoilParameterList` command described above. The available vegetation parameters in Raven are described in table A.3



Name	Definition	Units	Reasonable range	
MAX_HEIGHT	maximum vegetation height	[m]		Physical Parameters
MAX_LEAF_COND	maximum leaf conductance	[mm/s]		
MAX_LAI	maximum leaf area index	[m2/m2]		
SVF_EXTINCTION*	extinction coefficient used to calculate skyview factor	[-]	~0.5	
RAIN_ICEPT_PCT*	relates percentage of throughfall of rain to LAI+SAI	[-]	0.02-0.20	
SNOW_ICEPT_PCT*	relates percentage of throughfall of snow to LAI+SAI	[-]	0.02-0.20	
RAIN_ICEPT_FACT*	percentage of rain intercepted (maximum)	[0..1]	~0.06	
SNOW_ICEPT_FACT*	percentage of snow intercepted (maximum)	[0..1]	~0.04	
SAI_HT_RATIO*	ratio of stem area index to height	[m2/m3]		
TRUNK_FRACTION*	fraction of canopy attributed to tree trunk	[0..1]		
STEMFLOW_FRAC*		[0..1]	~0.03	
ALBEDO*	visible/near-infrared albedo of leaf	[-]	~0.15	
ALBEDO_WET*	albedo of wet leaf	[-]		
MAX_CAPACITY*	maximum canopy storage capacity	[mm]		
MAX_SNOW_CAPACITY*	maximum canopy snow (as SWE) storage capacity	[mm]		
ROOT_EXTINCT	extinction coefficient for roots, exp(-ext*z)	[]-		
MAX_ROOT_LENGTH	root length per unit canopy area	[mm/m2]		
MIN_RESISTIVITY	1.0/max_conductivity	[d/mm]		
XYLEM _FRAC	fraction of plant resistance in xylem	[0..1]		
ROOTRADIUS	average root radius (used to calculate cowan alpha)	[mm]		
PSI_CRITICAL	minimum plant leaf water potential	[-mm]		
DRIP_PROPORTION	drip proportion for bucket drip model	[1/d]		Conceptual Model Parameters
MAX_INTERCEPT_RATE	maximum rate of rainfall interception	[mm/d]		
CHU_MATURITY	crop heat unit maturity; level at which PET is maximized	[-]		

Figure A.3: Vegetation Parameters. The parameters with an asterisk can be autogenerated by Raven or overriden by the model user

-
  ```
      :SeasonalCanopyLAI
        {[DEFAULT]      ,  J, F, M, A, M, J, J, A, S, O, N, D} {optional}
        { veg_class_name,  J, F, M, A, M, J, J, A, S, O, N, D}x[<=NVC]
      :EndSeasonalCanopyLAI
  ```

  The `:SeasonalCanopyLAI` command provides a monthly correction factor that can be used to adjust leaf area indices as the seasons change, i.e., $LAI = LAI_{max} \cdot f$, where $f(t)$ is the monthly correction factor for time t. By default, no correction factor is applied. This correction factor must be between zero and one for all months and will be interpolated based upon the specification of the `:MonthlyInterpolationMethod` command in the .rvi file.

-
  ```
  :SeasonalCanopyHeight
    {[DEFAULT]      ,  J, F, M, A, M, J, J, A, S, O, N, D} {optional}
    { veg_class_name,  J, F, M, A, M, J, J, A, S, O, N, D}x[<=NVC]
  :EndSeasonalCanopyHeight
  ```

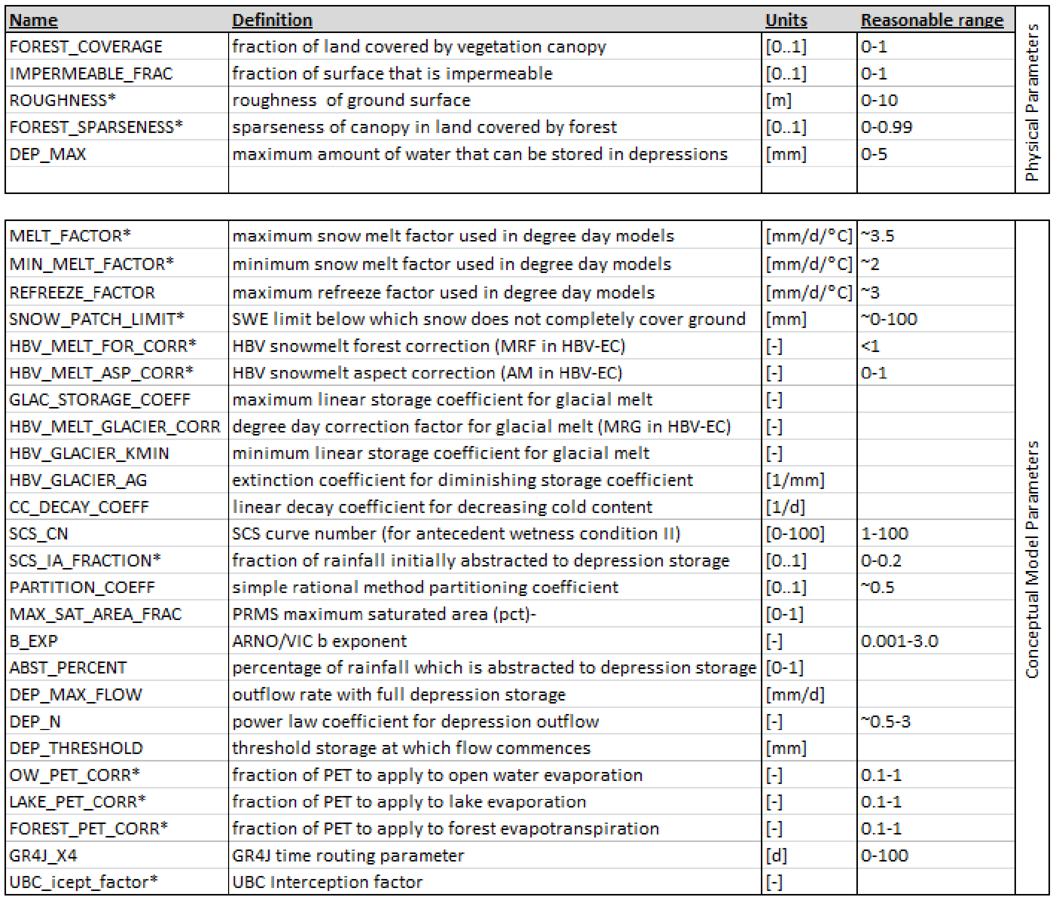
The `:SeasonalCanopyHeight` command provides a monthly correction factor that can be used to adjust vegetation height as the seasons change, i.e., $h_{veg} = h_{max} \cdot f$, where $f(t)$ is the monthly correction factor for time t. By default, no correction factor is applied. This correction factor must be between zero and one for all months and will be interpolated based upon the specification of the `:MonthlyInterpolationMethod` command in the .rvi file.

Land Use / Land Type Parameter Specification

- `:LandUseParameterList`

```
:LandUseParameterList
  :Parameters      , { param_name1, param_name1,...,  param_nameNP}
  :Units           , {  unit_type1,  unit_type2,...,   unit_typeNP}
  {[DEFAULT]       , {default_val1,default_val2,..., default_valNP} [optional]
  {lult_class_name, {  param_val1,  param_val2,...,   param_valNP}}x[<=NSC]
:EndSoilProperties
```

The `:LandUseParameterList` command operates in the same fashion as the `:SoilParameterList` command described above. The available land use parameters in Raven are described in table A.4



Name	Definition	Units	Reasonable range	
FOREST_COVERAGE	fraction of land covered by vegetation canopy	[0..1]	0-1	Physical Parameters
IMPERMEABLE_FRAC	fraction of surface that is impermeable	[0..1]	0-1	
ROUGHNESS*	roughness of ground surface	[m]	0-10	
FOREST_SPARSENESS*	sparseness of canopy in land covered by forest	[0..1]	0-0.99	
DEP_MAX	maximum amount of water that can be stored in depressions	[mm]	0-5	
MELT_FACTOR*	maximum snow melt factor used in degree day models	[mm/d/°C]	~3.5	Conceptual Model Parameters
MIN_MELT_FACTOR*	minimum snow melt factor used in degree day models	[mm/d/°C]	~2	
REFREEZE_FACTOR	maximum refreeze factor used in degree day models	[mm/d/°C]	~3	
SNOW_PATCH_LIMIT*	SWE limit below which snow does not completely cover ground	[mm]	~0-100	
HBV_MELT_FOR_CORR*	HBV snowmelt forest correction (MRF in HBV-EC)	[-]	<1	
HBV_MELT_ASP_CORR*	HBV snowmelt aspect correction (AM in HBV-EC)	[-]	0-1	
GLAC_STORAGE_COEFF	maximum linear storage coefficient for glacial melt	[-]		
HBV_MELT_GLACIER_CORR	degree day correction factor for glacial melt (MRG in HBV-EC)	[-]		
HBV_GLACIER_KMIN	minimum linear storage coefficient for glacial melt	[-]		
HBV_GLACIER_AG	extinction coefficient for diminishing storage coefficient	[1/mm]		
CC_DECAY_COEFF	linear decay coefficient for decreasing cold content	[1/d]		
SCS_CN	SCS curve number (for antecedent wetness condition II)	[0-100]	1-100	
SCS_IA_FRACTION*	fraction of rainfall initially abstracted to depression storage	[0..1]	0-0.2	
PARTITION_COEFF	simple rational method partitioning coefficient	[0..1]	~0.5	
MAX_SAT_AREA_FRAC	PRMS maximum saturated area (pct)-	[0-1]		
B_EXP	ARNO/VIC b exponent	[-]	0.001-3.0	
ABST_PERCENT	percentage of rainfall which is abstracted to depression storage	[0-1]		
DEP_MAX_FLOW	outflow rate with full depression storage	[mm/d]		
DEP_N	power law coefficient for depression outflow	[-]	~0.5-3	
DEP_THRESHOLD	threshold storage at which flow commences	[mm]		
OW_PET_CORR*	fraction of PET to apply to open water evaporation	[-]	0.1-1	
LAKE_PET_CORR*	fraction of PET to apply to lake evaporation	[-]	0.1-1	
FOREST_PET_CORR*	fraction of PET to apply to forest evapotranspiration	[-]	0.1-1	
GR4J_X4	GR4J time routing parameter	[d]	0-100	
UBC_icept_factor*	UBC Interception factor	[-]		

Figure A.4: Land Use Parameters. The parameters with an asterisk can be autogenerated by Raven or overriden by the model user

Global Parameter Specification

The following global parameters can also be specified, anywhere in the .rvp file:

- `:AdiabaticLapseRate [rate]`
  The base adiabatic lapse rate [°C/m]
- `:WetAdiabaticLapseRate [rate]`
  The wet adiabatic lapse rate [°C/m]
- `:PrecipitationLapseRate`
  The simple linear precipitation lapse rate [mm/d/km], as used in the `OROCORR_SIMPLELAPSE` orographic correction algorithm.
- `:RainSnowTransition [rainsnow_temp] [rainsnow_delta]`
  Specifies the range of temperatures (`rainsnow_delta`, [°C]) over which there will be a rain/snow mix when partitioning total precipitation into rain and snow components. The midpoint of the range is `rainsnow_temp`.
- `:IrreducibleSnowSaturation [saturation]`
  Maximum liquid water content of snow, as percentage of SWE [0..1]
- `:ReferenceMaxTemperatureRange [range]`
  A parameter (A0TERM) used in the UBC watershed model orographic corrections for temperature [°C]
- `:AverageAnnualRunoff`
  This parameter should be the average annual runoff for the entire modeled watershed, in mm. It is used to autogenerate initial flows and reference flows in the channel network. While the resultant estimates of initial flows will wash out with time, reference flows may be critical and modelers may wish to overwrite these by specifying the `Q_REFERENCE` parameter for each channel in the `:SubBasinProperties` command of the .rvp file.
- `:AvgAnnualSnow`
  This parameter is the average annual snow for the entire watershed in mm SWE. It is used in the `CEMA_NIEGE` snowmelt algorithm.
- `:UBCTempLapseRates [A0TLXM A0TLNM A0TLXH A0TLNH P0TEDL P0TEDU]`
  Parameters used in the UBC watershed model orographic corrections for temperature. A0TLXM and A0TLXH [°C/km] are the low and high elevation lapse rates of the maximum daily temperature; A0TLNM and A0TLNH [°C/km] are the low and high elevation lapse rates of the minimum daily temperature; P0TEDL and P0TEDU [°C/km] are the low and high elevation lapse rates of the maximum temperature range. Low and high elevation refer to below or above 2000 masl.
- `:UBCPrecipLapseRates [E0LLOW E0LMID E0LHI P0GRADL P0GRADM P0GRADU A0STAB]`
  Parameters used in the UBC watershed model orographic corrections for precipitation. E0LLOW E0LMID and E0LHI, are the low, medium, and high reference elevations; P0GRADL, P0GRADM, and P0GRADU are the precipitation gradient factors (%) applied below E0LMID, between E0LMID and E0LHI, and above E0LHI, respectively; A0STAB is a precipitation gradient modification factor.
- `:UBCEvapLapseRates [A0PELA]`
  The PET lapse rate [°C/km].

- `:UBCNorthSWCorr [J F M A M J J A S O N D]`
  Monthly correction factors for shortwave radiation on north-facing slopes, used in the UBC shortwave generation routine.
- `:UBCSouthSWCorr [J F M A M J J A S O N D]`
  Monthly correction factors for shortwave radiation on south-facing slopes, used in the UBC shortwave generation routine.
- `:UBCSnowParams [P0ALBMIN P0ALBMAX P0ALBREC P0ALBASE P0ALBSNW P0ALBMLX]`
  Parameters used in the UBCWM-style snow albedo evolution algorithm. `P0ALBREC` [-] is the recessional constant for albedo decay of new snow (0.9); `P0ALBSNW` [mm] is the daily snowfall required to bring albedo to that of new snow; `P0ALBMAX` is the albedo of fresh snow (0.95); `P0ALBMIN` is the albedo of an aged snowpack or glacier (0.30); `P0ALBMLX` [mm] is a constant on the order of total snowmelt in one year; P0ALBASE is the albedo initial decay value (0.65).
- `:UBCGroundwaterSplit [value]`
  The UBC watershed model deep zone share, which controls how much infiltration goes to deep vs. shallow storage.
- `:UBCExposureFactor`
  The UBCWM sun exposure factor for forested areas (0.01), indicating the percentage of forested areas exposed to solar radiation. Used in the `SW_CANOPY_CORR_UBCWM` canopy correction algorithm.
- `:UBCCloudPenetration`
  The UBCWM fraction of solar radiation penetrating cloud cover [0..1], as used in the `SW_CLOUD_CORR_UBCW`
  cloud cover correction algorithm.
- `:UBCLWForestFactor`
  The UBCMW Longwave correction factor for forests [mm/d/K](0.75), as used in the `LW_RAD_UBCWM` longwave radiation estimation routine.
- `:AirSnowCoeff`
  This is the air/snow heat transfer coefficient in units of [1/d], as used in the `SNOTEMP_NEWTONS` snow temperature evolution routine.

Special Commands

The following special commands can be used for temporally variable landscape change (e.g., to simulate urbanization, forest fire impacts, or changes in agricultural practices).

– `:LandUseChange [HRU group] [new LULT tag] [YYYY-mm-dd]`
  The land use for the specified HRU group is changed to the new LULT type (as specified in the `:LandUseClasses-:EndLandUseClasses` block) on the specified date in ANSI YYYY-mm-dd format. The change occurs just after midnight of the night before. Note that all parameters from the new land use class are applied to all of the specified HRUs in the group. There is no limit to the number of land use changes in the model.

– `:VegetationChange [HRU group] [new vegetation tag] [YYYY-mm-dd]`
  The vegetation for the specified HRU group is changed to the new vegetation type (as specified in the `:VegetationClasses-:EndVegetationClasses` block) on the specified date in ANSI YYYY-mm-dd format. The change occurs just after midnight of the night before. Note that all parameters from the new vegetation class are applied to all of the specified HRUs in the group. There is no limit to the number of vegetation changes in the model.

–
```
    :TransientParameter [PARAM_NAME] [Parameter_class] {(optional) ClassName}
      [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [number of entries (N
      {double value} x N
    :EndTransientParameter
```

  This command may be used to replace any (usually fixed) parameter specified in the .rvp file with a time series of user-specified parameter values. This is often used to represent the influence of changing land use, seasonal impacts of agriculture, or unmodeled hydrologic processes such as frozen soils. Here, `PARAM_NAME` corresponds to one of the parameters included in tables A.2, A.4, or A.3. `Parameter_class` is one of `SOIL`, `VEGETATION`, `LANDUSE`, `TERRAIN` or `GLOBALS`. The optional ClassName specifies the particular soil/vegetation/land use class to override; if not included, the parameter will be overridden for all soil/vegetation/land use classes. Note that the specified transient parameter completely overwrites the static value specified earlier in the .rvp file. It is common to put this time series in another file and point to it via the `:RedirectToFile` command.

– `:RedirectToFile [filename]`
  This treats the contents of file "filename" as if they were simply inserted into the .rvp file at the location of the `:RedirectToFile` command. This is useful for storing individual sets of commands in an organized format (e.g., the `:TransientParameter` time series). If no path is specified, the filename must be reported relative to the working directory. Note that this command cannot work within data blocks (e.g., a the entire `:SoilParameters-:EndSoilParameters` block would have to be in a single file, not just the tabular data in that block).

A.3 HRU / Basin Definition file (.rvh)

The HRU/basin definition file describes the topology of the basin network and the class membership of all constituent HRUs. An example .rvh file is shown below:

Example File: modelname.rvh

```
* -------------------------------------------
* Raven HRU Input file
* TEST input
* Author: JRC
* -------------------------------------------
:SubBasins
 :Attributes,   NAME, DOWNSTREAM_ID, PROFILE, REACH_LENGTH, GAUGED
 :Units,        none,          none,    none,           km,   none
     1,  Downstream,            -1, DEFAULT,          3.0,      1
     2,    Upstream,             1, DEFAULT,          3.0,      0
:EndSubBasins
:HRUs
 :Attributes, AREA, ELEVATION, LATITUDE, LONGITUDE, BASIN_ID, LAND_USE_CLASS,  ...
                VEG_CLASS,SOIL_PROFILE, AQUIFER_PROFILE, TERRAIN_CLASS, SLOPE, ASPECT
 :Units,      km2,          m,      deg,      deg,      none,           none, ...
                     none,        none,            none,         none,    deg,   degN
  101,  10,143, 43,-80,   1,FORESTED,BROADLEAF, ALL_SAND,SAND_AQ,  HILLY,0.0,0.0
  102,  10,145, 43,-80,   1,URBAN   ,BROADLEAF, ALL_SAND,SAND_AQ,  HILLY,0.0,0.0
  103,  10,143, 43,-80,   2,FORESTED,BROADLEAF,     TILL,SAND_AQ,  HILLY,0.0,0.0
  104,  10,147, 43,-80,   2,FORESTED,BROADLEAF,     TILL,SAND_AQ,  HILLY,0.0,0.0
:EndHRUs
:HRUGroup ForestedHRUs
 101,103,104
:EndHRUGroup
:RedirectToFile Reservoirs.rvh
```

Note that, as with the .rvi file, comments may be included on individual lines using the `*` or `#` characters as the first word on the line.

A.3.1 Required Commands

The .rvh file consists of the following required commands:

-
  ```
  :SubBasins
    :Attributes,   ID, NAME, DOWNSTREAM_ID, PROFILE, REACH_LENGTH, GAUGED,
    :Units     , none, none,          none,    none,           km,   none,
    {ID,name,downstream_ID profile,reach_length,gauged}x[number of subbasins]
  :EndSubBasins
  ```

  To specify an array of SubBasins of the watershed and the connectivity between subbasins. Each subbasin may only have one outlet subbasin, specified by ID (a unique positive integer). The subbasin-specific parameters are defined as follows:

  - `ID` A positive integer unique to this subbasin. Used to refer to the subbasin in other parts of the input file.
  - `name` The nickname for the basin (cannot include commas or spaces)
  - `downstream_ID` The ID of the basin that receives this subbasins outflowing waters. If the drainage for this subbasin leaves the modeled watershed, a value of -1 for the downstream ID should be specified.
  - `profile` The representative channel profile code (channel profiles specified in the .rvp file)
  - `reach_length` The length of the primary reach channel in the basin (in km). If this is a headwater basin, in-channel routing can be avoided by setting `reach_length` to zero. If set to `_AUTO`, the reach length will be estimated from total subbasin area.
  - `gauged` Flag which determines whether modeled hydrographs for this subbasin are generated as output from the model (either 1 or 0, true or false)

-
  ```
  :HRUs
    :Attributes,AREA,ELEVATION,LATITUDE,LONGITUDE,BASIN_ID,LAND_USE_CLASS,
       VEG_CLASS,SOIL_PROFILE,AQUIFER_PROFILE,TERRAIN_CLASS,SLOPE,ASPECT
    :Units     ,km2,         m,     deg,      deg,    none,          none,
           none,        none,           none,         none,  deg,  degN
    {ID,area,lat,long,basin_ID,...
       LU/LT,veg_class_name,soil_profile_name,...
           terrain_class_name,slope,aspect}x[number of HRUs]
  :EndHRUs
  ```

  To specify an array of HRUs within the subbasins defined above. Each HRU is defined by an ID (a unique positive integer), a total HRU area (in km^2), a latitude-longitude location of the HRU centroid (in decimal degrees), the ID of the basin in which the HRU is located (as defined in the `:SubBasins` command), land use,terrain, aquifer classes and a soil profile (as defined in the .rvp file), an average slope (in degrees), and average aspect (in degrees).

  If terrain classes or aquifer profiles are not used in the model, the flag `[NONE]` goes in the place of the class specifier.

A.3.2 Optional Commands

-
  ```
  :SubBasinProperties
    :Parameters, {PARAM_1, PARAM_2, .. , PARAM_N}
    :Units     , {UNITS_1, UNITS_2, .. , UNITS_N}
    {[basin ID], [p_1]   , [p_2]  , .. , [p_N] }} x NSB
  ```

  Subbasin properties are used to control the in-catchment routing behaviour of individual subbasins. Here, `PARAM_i` represents the name of a subbasin parameter (the full list of valid parameters can be found in table C.3), `UNITS_i` is the units tag (not used by Raven), `p_i` refers to numeric values of each parameter, `basin id` is the subbasin ID as defined in the `:SubBasins` command, and `NSB` is the number of subbasins in the model.

-
  ```
  :HRUGroup [group_name]
    17,18,30-37
  :EndHRUGroup
  ```

HRU Groups are used for a number of reasons: to generate custom output only for a select set of HRUs (or organize/aggregate output for multiple sets) or to control which processes are applied in what locations. Group names are typically specified using the `:DefineHRUGroups` command in the .rvi file; this command populates the memberships of these predefined groups. Individual HRUs are specified with their ID numbers (as defined in the `:HRUs` command), separated by commas. Ranges of HRUs can be specified using the hyphen, as shown above.

-
  ```
  :SubBasinProperties
     :Parameters PARAMETER_1 PARAMETER_2 PARAMETER_3, ...
  :Units unit_1 unit_2 unit_3,...
      {basin ID,v_1, v_2, v_3,...}x[number of SubBasins]
  :EndSubBasinProperties
  ```

  This command allows the user to specify subbasin properties, mostly those use to control the in-catchment routing schemes. The list of sub-basin parameters is included in table C.3

-
  ```
  :Reservoir {name}
    :SubBasinID {SBID}
    :HRUID {HRUID}
    :StageRelations
      {# of points on rating curve}
      {stage [m], flow [m3/s], volume [m3], area [m2]}x[# of points on rating curve]
    :EndStageRelations
  :EndReservoir
  ```

  This command creates a reservoir at the outlet of the subbasin referenced by `SBID`. Evaporation from the reservoir surface are obtained from the HRU referenced by `HRUID` (this is the only purpose for this; a special HRU for the reservoir is not strictly required, though often appropriate if the reservoir is relatively large). The reservoir volume, outflow, and net precipitation to the reservoir surface are obtained by interpolating their value from the specified stage-discharge, stage-area, and stage-volume relations. Note that the minimium stage supplied in the `:StageRelations` should be the minimum expected stage (usually the bottom of the reservoir).

A.4 Time Series Input file (.rvt)

The time series input file is used to store time series of forcing functions (precipitation, temperature, etc.).

Example File: modelname.rvt

```
* -------------------------------------------
* Raven Time Series Input file
* -------------------------------------------
:Gauge Stratford MOE (ID:6148105)
  :Latitude   43.37250
  :Longitude -80.55360
  :Elevation 53
  :RedirectToFile StratfordMOEData.rvt
:EndGauge
:RedirectToFile UpstreamInflow.rvt
:RedirectToFile LandCoverChange.rvt
:RedirectToFile ObservedHydrograph.rvt
```

A.4.1 Gauge Data Commands

The entries in the .rvt file are predominantly meteorological gauge locations (either real or hypothetical) that provide time series of needed precipitation, temperature and other atmospheric forcings used by the model (see appendix A.4.3 for information about using gridded model inputs instead of gauges). This is supplemented by information about other time series needed for simulation. Each gauge entry is specified within a bracketed statement,

```
:Gauge [gaugename]
  :Latitude [latitude]
  :Longitude [longitude]
  :Elevation [elevation]
  [other gauge data and time series information here]
:EndGauge
```

and must contain the latitude/longitude (using the `:Latitude`, `:Longitude` commands) and typically contain a number of time series. Two formats, `:Data` (for a single time series) and `:MultiData` (for multiple time series), may be used to specify collections of forcing functions measured at the gauge. These are often stored in their own individual file and accessed via the `:RedirectToFile` command.

-
  ```
  :Data PARAMETER
    [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [number of entries (N)]
    v_1
    v_2
    v_3
    ...
    v_N
  :EndData
  ```

Where here, `v_i` are the i^{th} time series values and the `PARAMETER` term is one of the forcings listed in table C.2 (e.g., `PRECIP`, `TEMP_MIN`. etc.).

It is assumed that the array of values specified are time-averaged values over the specified time interval. All forcings are in period-starting format, so that if the start date is 2002-10-01 00:00:00 with a time interval of 1.0 days, then the first data item represents the average forcing value on October 1st. Note that the terms may be space-, comma-, or tab-delimited and would typically be entered as a single column. Also note that the time interval must be specified as a double, and cannot be specified using a format of 00:00:00.

IMPORTANT: The default units of the forcing functions (as tabulated in C.2) must be respected. Though non-intuitive to many hydrologists, precipitation intensity (in mm/d) must be specified even for hourly data intervals, e.g., 1 cm of rain in an hour would be specified as a rainfall rate of 240 mm/d.

- `:MultiData`

```
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [number of entries (N)]
  :Parameters PARAMETER_1 PARAMETER_2 ... PARAMETER_J
  :Units      units_tag_1 units_tag_2 ... units_tag_J
  v_11, v_12, v_13
  v_21, v_22, v_23
  ...
  v_N1, v_N2, v_N3
:EndMultiData
```

  This command is an alternate to the `:Data` approach, allowing multiple data to be included as a single data table using the `:MultiData` command, with columns corresponding to individual data types. Here, `PARAMETER_i` corresponds to the name of the input parameter (one of the forcing values in table C.2), and the units tags should be consistent with the actual desired units in table C.2. Again, note that the time interval must be specified as a double, and cannot be specified using a format of 00:00:00.

Other additional terms may be associated with each gauge, contained between the `:Gauge`-`:EndGauge` brackets:

- `:Elevation [elevation]`
  elevation of gauge, typically in meters above mean sea level
- `:MonthlyAveTemperature [J F M A M J J A S O N D]`
  a list of 12 representative monthly average temperatures at the gauge, from Jan to Dec, in °C.
- `:MonthlyMinTemperature [J F M A M J J A S O N D]`
  a list of 12 representative monthly minimum temperatures at the gauge, from Jan to Dec, in °C.
- `:MonthlyMaxTemperature [J F M A M J J A S O N D]` a list of 12 representative monthly maximum temperatures at the gauge, from Jan to Dec, in °C.
- `:MonthlyAveEvaporation [J F M A M J J A S O N D]`
  a list of 12 representative monthly average potential evapotranspiration rates at the gauge, from Jan to Dec, in mm/d.
- `:MonthlyEvapFactor [J F M A M J J A S O N D]`

a list of 12 monthly evaporation factors [mm/d/K]. This is used in the UBCWM PET estimation routine.

- `:RainCorrection [value]`
  a multiplier (hopefully near 1.0) applied to all reported rainfall rates at this gauge; often used as a correction factor for estimating proper rainfall volumes
- `:SnowCorrection [value]`
  a multiplier (hopefully near 1.0) applied to all reported snowfall rates at this gauge; often used as a correction factor for estimating proper snow volumes.
- `:CloudTempRanges [cloud_temp_min] [cloud_temp_max]`
  temperature ranges used for estimation of cloud cover using the UBCWM model approach (`CLOUDCOV_UBCWM`).
- `:RedirectToFile [filename]`
  This treats the contents of file "filename" as if they were simply inserted into the .rvt file at the location of the `:RedirectToFile` command. This is useful for storing individual time series or gauges in separate files. If no path is specified, the filename must be reported relative to the working directory. Note that this command can work within a `:Gauge`-`:EndGauge` structure, but not within other structures (e.g., a `:Multidata` entry cannot be split into multiple files in this manner).
- `:EnsimTimeSeries [filename]`
  A table of timeseries (similar to the `:MultiData` command may be specified using the Ensim .tb0 format. The input parameter names are the same which are provided in table C.2. An example is provided below:

```
#########################################################################
:FileType tb0  ASCII  EnSim 1.0
#------------------------------------------------------------------------
:ColumnMetaData
  :ColumnName  TEMP_MAX TEMP_MIN PRECIP
  :ColumnUnits DegC     DegC     mm/d
  :ColumnType  float    float    float
:EndColumnMetaData
#
:StartTime       1983/02/01 00:00:00.000
:DeltaT          24:00:00.000
#
:EndHeader
4.4000001 -0.60000002 0
5        -2.5     0.60000002
...
5.5999999 -3       0.30000001
4.4000001 -4.5999999 0
1.1      -4.4000001 0
```

For dense gauge networks, it may be more practical to generate gauge inputs in bulk. Therefore, the following commands have been supplied:

- `:GaugeList`

```
  :Attributes,  LATITUDE, LONGITUDE, ELEVATION, ...
  :Units,   dec.deg, dec.deg, masl, ...
  [name1, lat1, long1,elev1,...]
  ...
  [nameN, latN, longN,elevN,...]
:EndGaugeList
```

The Gauge List command allows specification of single-valued parameters for each gauge. In addition to `LATITUDE` (which corresponds to a `:Longitude` entry in a `:Gauge-:EndGauge` command), it also supports all of the other single-valued entries listed above, including `RAINFALL_CORR`, `SNOWFALL_CORR`, `CLOUD_MIN_RANGE`, and `CLOUD_MAX_RANGE`.

- `:GaugeDataTable`

```
  :DataType        PRECIP
  :Units           mm/d
  :StartTime       01-01-2012 00:00:00.0
  :TimeIncrement   01:00:00.0
  :NumMeasurements 730
  :Gauge, Cell_11,Cell_12,Cell_13, ... Cell_240360
    1,  0.0,0.0,0.0,               ...,0.2
    2,  0.0,0.0,0.0,               ...,0.1
    ...
:EndGaugeDataTable
```

- `:MonthlyMaxTemperatures`
This command does the same thing as `:MonthlyMaxTemperature` but for a list of gauges, and would therefore typically be used in conjunction with the `:GaugeList` and `:GaugeDataTable` commands. The format is

```
:MonthlyMaxTemperatures
  {[gaugename1], J,F,M,A...O,N,D} x NG
:EndMonthlyMaxTemperatures
```

where each of the month initials would be replaced by an typical maximum temperature for the month.

- `:MonthlyMinTemperatures`
This command is identical to `MonthlyMaxTemperatures`, but for representative minimum monthly temperatures.

- `:MonthlyAveEvaporations`
This command is identical to `MonthlyMaxTemperatures`, but for representative average monthly evaporation rates (in mm/d).

- `:MonthlyAveTemperatures`
This command is identical to `MonthlyMaxTemperatures`, but for representative average monthly temperatures.

- `:MonthlyEvapFactors`
This command is identical to `MonthlyMaxTemperatures`, but for monthly evaporation factors [mm/d/K] as defined using the `:MonthlyEvapFactor` command above.

Here, the `:DataType` entry corresponds to a forcing tag as specified in table C.2.

A.4.2 Other Time Series Commands

Time series of known flows and model parameters may also need to be specified to support the model. These are not linked to a specific Gauge, and would therefore not be included in an `:Gauge`...`:EndGauge` bracket. Most of these time series would be stored in a separate .rvt file and referred to in the main .rvt file using the `:RedirectToFile` command.

-
  ```
  :ObservationData [data_type] [basin_ID or HRU_ID] {units}
    [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [number of entries (N)]
    v_1 v_2 v_3 v_4 v_5
    ...
    v_N-2 v_N-1 v_N
  :EndObservationData
  ```

  Similar to the `:Data` command above. This specifies a continuous time series of observations of type `data_type` with units `units` located either at the outlet of the basin specified with `basin_ID` or the HRU specified with `HRU_ID`. The data types correspond to state variables in the model, and the `data_type` therefore must be taken from table C.1, unless the data is (1) a hydrograph, in which case the `HYDROGRAPH` tag is used or (2) a reservoir stage, in which case the `RESERVOIR_STAGE` tag is used. For hydrographs and reservoir stages, the basin ID is specified. For all other variables, the HRU ID is specified. With the exception of the hydrograph, it is assumed that the observations correspond to instantaneous observations in time rather than time-averaged quantities. This command defines a time series of regularly spaced consecutive values. If the time series time interval doesn't match the model time step then the time series is re-sampled to match the model. For irregularly spaced observations, use the `:IrregularObservations` command.

  Missing or unknown observations should be specified using the flag -1.2345. Note that the observation time series does not have to overlap the model simulation duration. All data outside the supplied time interval is treated as blank.

  If an observed hydrograph is supplied, it will be output to the `Hydrographs.csv` file. Hydrographs should be specified in period-starting format, i.e., for a time series of daily discharges starting on October 1, 2006, the start time would be 2006-10-01 00:00:00, at the *start* of the first data period provided.

-
  ```
  :IrregularObservations [data type] [ID] [number of entries (N)] {(optional) units}
    [date yyyy-mm-dd] [time hh:mm:ss.0] v_1
    [date yyyy-mm-dd] [time hh:mm:ss.0] v_2
    ...
    [date yyyy-mm-dd] [time hh:mm:ss.0] v_N
  :EndIrregularObservations
  ```

  This command is used for time series where observations are discontinuous or irregularly spaced. Values in these time series are assumed to be instantaneous and modelled values are linearly interpolated to match the observation times for comparison.

  Missing or unknown observations should be specified using the flag -1.2345. Note that the observation time series does not have to overlap the model simulation duration. All data outside the supplied time interval is treated as blank.

-
  ```
  :ObservationWeights [data type] [ID]
    [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [number of entries (N)]
    v_1 v_2 v_3 v_4 v_5
  ```

```
  ...
  v_N-2 v_N-1 v_N
:EndObservationWeights
```

This command is used apply weights to observation data for the calculation of diagnostics. The data type, ID, and number of entries all need to match an existing `:ObservationData` time series. Not all evaluation metrics can be weighted, in which case all weights are ignored except weights of zero.

-
  ```
  :IrregularWeights [data type] [ID] [number of entries (N)]
    [date yyyy-mm-dd] [time hh:mm:ss.0] v_1
    [date yyyy-mm-dd] [time hh:mm:ss.0] v_2
    ...
    [date yyyy-mm-dd] [time hh:mm:ss.0] v_N
  :EndIrregularWeights
  ```

  This command is used apply weights to irregular observations. The data type, ID, and number of entries all need to match an existing `:IrregularObservations` time series.

-
  ```
  :BasinInflowHydrograph [Basin ID]
    [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [number of entries (N)]
    Q_1 Q_2 Q_3 Q_4 Q_5
    ...
    Q_N-2 Q_N-1 Q_N
  :EndBasinInflowHydrograph
  ```

  where `Q_i` is the i^{th} inflow in m^3d^{-1}. This command is typically used to (1) specify inflows coming from an unmodeled portion of the domain; (2) override modeled inflow to a stream reach with observed inflows from a stream gauge, as might be done during calibration; or (3) add additional inflows to a stream reach from human activities, e.g., a wastewater treatment plant inflow.

-
  ```
  :ReservoirExtraction [Basin ID]
    [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [number of entries (N)]
    Q_1 Q_2 Q_3 Q_4 Q_5
    ...
    Q_N-2 Q_N-1 Q_N
  :EndReservoirExtraction
  ```

  where `Q_i` is the i^{th} inflow in m^3d^{-1}. Discharges are positive for reservoir extraction and negative for injection of water into the reservoir located at the outlet of the subbasin indicated by the basin ID.

A.4.3 Gridded Input Data

Raven supports gridded forcing inputs exclusively in netCDF format (*.nc files). In case of gridded inputs, the user needs to define some information about the variables and structure of the gridded NetCDF input file; in addition, the mapping of grid cells to HRUs needs to be specified through a weighting table.

Example File: modelname.rvt

```
* -------------------------------------------
* Example Raven Gridded Input file
* -------------------------------------------
:GriddedForcing PRECIPITATION
  :ForcingType      PRECIP
  :FileNameNC       gridded_precip.nc
  :VarNameNC        pre
  :DimNamesNC       lon lat ntime   # must be in the order of (x,y,t)
  :GridWeights
    :NumberHRUs      3
    :NumberGridCells 24
     # HRU  GridCell  Weight
       1    15        0.4
       1    16        0.6
       2    14        1.0
       3    14        0.2
       3    15        0.3
       3    13        0.5
  :EndGridWeights
:EndGriddedForcing
```

The forcing inputs like precipitation and temperature are traditionally given as time series per gauging station (see sections A.4.2 and A.4.1). This becomes inconvenient if you have inputs available for multiple gauging stations or you even have the forcings available on a grid covering your whole modeling domain. Hence, Raven supports gridded input in NetCDF format. Instead of specifying a time series per gauge or grid cell in the .rvt file, one can specify a single input grid inside a `:GriddedForcing`-`:EndGriddedForcing` command structure:

```
:GriddedForcing {forcing name}
  :ForcingType      {type}
  :FileNameNC       {path/filename of .nc file}
  :VarNameNC        {name of variable in .nc file}
  :DimNamesNC       {long_name} {lat_name} {time_name}   # must be in the order of (x,y,t)
  :GridWeights
    :NumberHRUs      {total number of HRUs}
    :NumberGridCells {total number of grid cells}
      {HRU ID} {Cell ID} {weight}
      ...
  :EndGridWeights
:EndGriddedForcing
```

One has to specify the type of the forcing input in the `:ForcingType` command, e.g. `PRECIP` or `TEMP_AVE` (see Table C.2 for complete list). The name of the file containing the data has to be given

:FileNameNC. The file can contain more data than only this specific forcing; only the data of the specified variable :VarNameNC will be read and used by Raven. Since the order of the dimensions in a NetCDF file is not unique, one has to specify the dimension names starting with the x-dimension (usually longitudes), y-dimension (usually latitudes) and at last the name of the time dimension. To obtain the information about variable name :VarNameNC and dimension names :DimNamesNC, one can use the command line tool ncdump available with the netCDF library. Running the command

```
> ncdump -h gridded_precip.nc
```

will display the header information of the NetCDF file gridded_precip.nc and provide all the necessary information. The last required information is the :GridWeights block specifying how much each grid cell is contributing to each HRUs. Only non-zero weights have to be given; missing pairs are automatically assumed to be zero. The HRU ID has to correspond to the numbering in the :HRUs block of the .rvh file. The numbering of the grid cells is linewise starting with zero in the upper left corner of the grid. The weights per HRU ID have to sum up to 1.0 otherwise Raven raises an error messsage. The list of grid weights will get very long with large grids and multiple HRUs. In such a case, the :GridWeights block would typically be stored in a separate file then and the :RedirectFile functionality be used instead.

A.5 Initial Conditions Input file (.rvc)

The initial conditions input file is used to store the initial conditions for the model. By default, the initial conditions for all model state variables is zero, and there are no required commands in this file (it could even be completely empty).

Example File: modelname.rvc

```
* -------------------------------------------
* Raven Initial Conditions Input file
* -------------------------------------------
:HRUStateVariableTable
  :Attributes, SOIL[0], SNOW,
  :Units     ,      mm,   mm,
           1,      145,   33,
           2,      150,   13,
...
:EndHRUStateVariableTable
:BasinInitialConditions
  :Attributes,     Q
  :Units     ,  m3/s
          1 ,   3.6
:EndBasinInitialConditions
```

A.5.1 Optional Commands

-
  ```
  :HRUStateVariableTable
    :Attributes, {SV_TAG_1, SV_TAG_2,...,SV_TAG_NSV}
    :Units     , {units_1, units_2,...,units_NSV}
        {HRUID, SV_value_1,SV_value_2,...,SV_value_NSV}xNHRUs
  :EndHRUStateVariableTable
  ```

  Provides initial conditions for state variables in each HRU within the model. Here, `NSV` is the number of state variables for which initial conditions are provided, and $NHRUs$ is the number of HRUs in the model. `SV_TAG` refers to the state variable tag, with the complete list of state variable tags in table C.1. Note that initial conditions have to be provided for all HRUs in the model and initial conditions have to be entered in the same order as in the `:HRUs` command in the .rvh file.

-
  ```
  :BasinInitialConditions
    :Attributes,     Q
    :Units     ,  m3/s
            {SBID, FLOWRATE} x nSubBasins
  :EndBasinInitialConditions
  ```

  A list of initial outflow rates from the subbasins, indexed by subbasin ID as specified within the `:SubBasins` command of the .rvh file.

- `:UniformInitialConditions [SV_TAG] [value]`
  Applies a uniform initial condition (value) to the state variable corresponding to `SV_TAG`, with

the complete list of state variable tags in table C.1. If called after `:HRUStateVariableTable`, it will overwrite the initial conditions previously specified.

-
  ```
  :BasinStateVariables
    :BasinIndex ID, name
      :ChannelStorage [val]
      :RivuletStorage [val]
      :Qout [nsegs] [aQout x nsegs] [aQoutLast]
      :Qlat [nQlatHist] [aQlatHist x nQlatHist] [QlatLast]
      :Qin  [nQinHist] [aQinHist x nQinHist]
      {reservoir variables}
    :BasinIndex 1D, name
      ...
  :EndBasinStateVariables
  ```

  This command is usually generated only as part of the Raven solution file and would not typically be modified by the user. It fully describes the flow variables linked to the subbasin. Here, `:ChannelStorage` [m^3] is the volume of water in the channel, `:RivuletStorage` [m^3] is the volume of water waiting in catchment storage, `Qout` [m^3/s] the array of outflows at each reach segment, `Qlat` [m^3/s] is an array storing the time history of outflows to the channel, `Qin` [m^3/s] is the time history of inflows to the uppermost segment of the reach.

- `:TimeStamp`
  Specifies time stamp linked to the initial conditions file. This is generated automatically by Raven when it produces a snapshot of the state variables, such as when it generates the solution.rvc output file. The time stamp should be consistent with the start time of the model.

Appendix B

Output Files

- `WatershedStorage.csv`
  A comma-delimited file describing the total storage of water (in mm) in all water storage compartments for each time step of the simulation. Mass balance errors, cumulative input (precipitation), and output (channel losses) are also included. Note that the precipitation rates in this file are period-ending, i.e., this is the precipitation rate for the time step preceding the time stamp; all water storage variables represent instantaneous reports of the storage at the time stamp indicate. Created by default.
- `Hydrographs.csv`
  A comma-delimited file containing the outflow hydrographs (in m^3/s) for all subbasins specified as 'gauged' in the .rvh file. If the `:SnapshotHydrograph` command is used, this reports instantaneous flows at the end of each time step (plus the initial conditions at the start of the first time step). Without, this reports period-ending time-averaged flows for the preceding time step, as is consistent with most measured stream gauge data (again, the initial flow conditions at the start of the first time step are included). If observed hydrographs are specified, they will be output adjacent to the corresponding modelled hydrograph. Created by default.
- `ForcingFunctions.csv` (optional)
  A comma-delimited file containing the time series of all watershed-averaged system forcing functions (e.g., rainfall, radiation, PET, etc.). The output is all period-ending, i.e., the values reported correspond to the time-averaged forcings for the time step before the indicated time stamp. Created if `:WriteForcingFunctions` command included in .rvi file.
- `WatershedMassEnergyBalance.csv` (optional)
  A comma-delimited file describing the total cumulative fluxes of energy and water (in MJ/m^2 or mm) from all energy storage compartments for each time step of the simulation. Created if `:WriteMassBalanceFile` command included in .rvi file.
- `Parameters.csv` (optional)
  A comma-delimited file containing the values for all static specified and auto-generated parameters for all soil, vegetation, land use, and terrain classes. Created if `:WriteParametersFile` command included in .rvi file.
- `ReservoirStages.csv` (optional)
  A comma-delimited file reporting the instantaneous stage of all modeled reservoirs. Created automatically if reservoirs are present in the model.
- `{constituent}concentrations.csv` (optional)

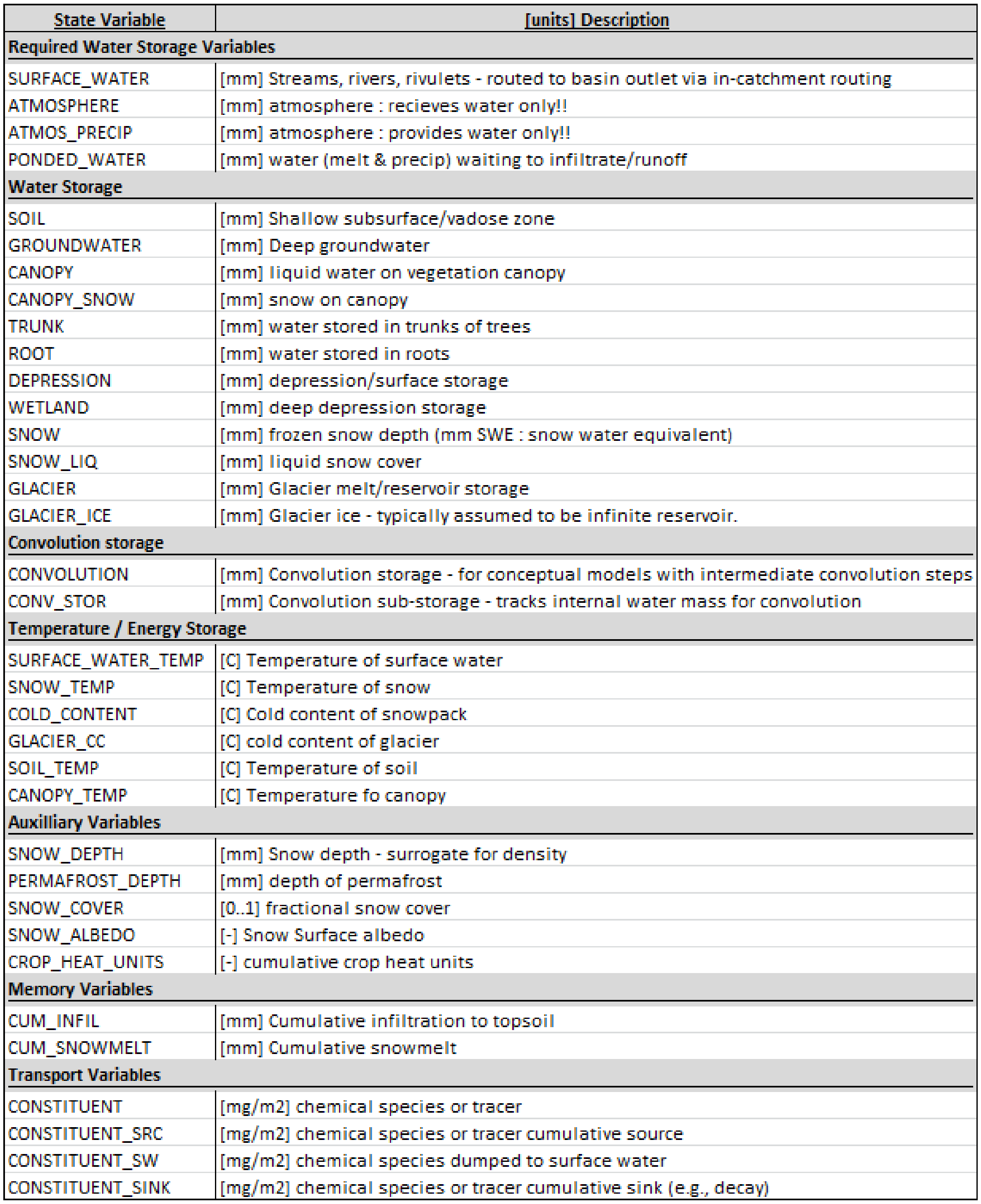
A comma-delimited file reporting the instantaneous watershed-averaged concentration of the transport constituent in all water storage units. Created automatically if transport is included in the model.

- {`constituent`}`pollutograph.csv` (optional)
  A comma-delimited file reporting the instantaneous concentration of water flowing out from all gauged subbasins. Created automatically if transport is included in the model.
- `Diagnostics.csv` (optional)
  A comma-delimited file reporting the quality of fit between model and supplied observations. Created if observations are present and the `:EvaluateMetrics` command is used.

If the `:RunName` parameter is specified in the .rvi file, this run name is pre-appended to the above filenames.

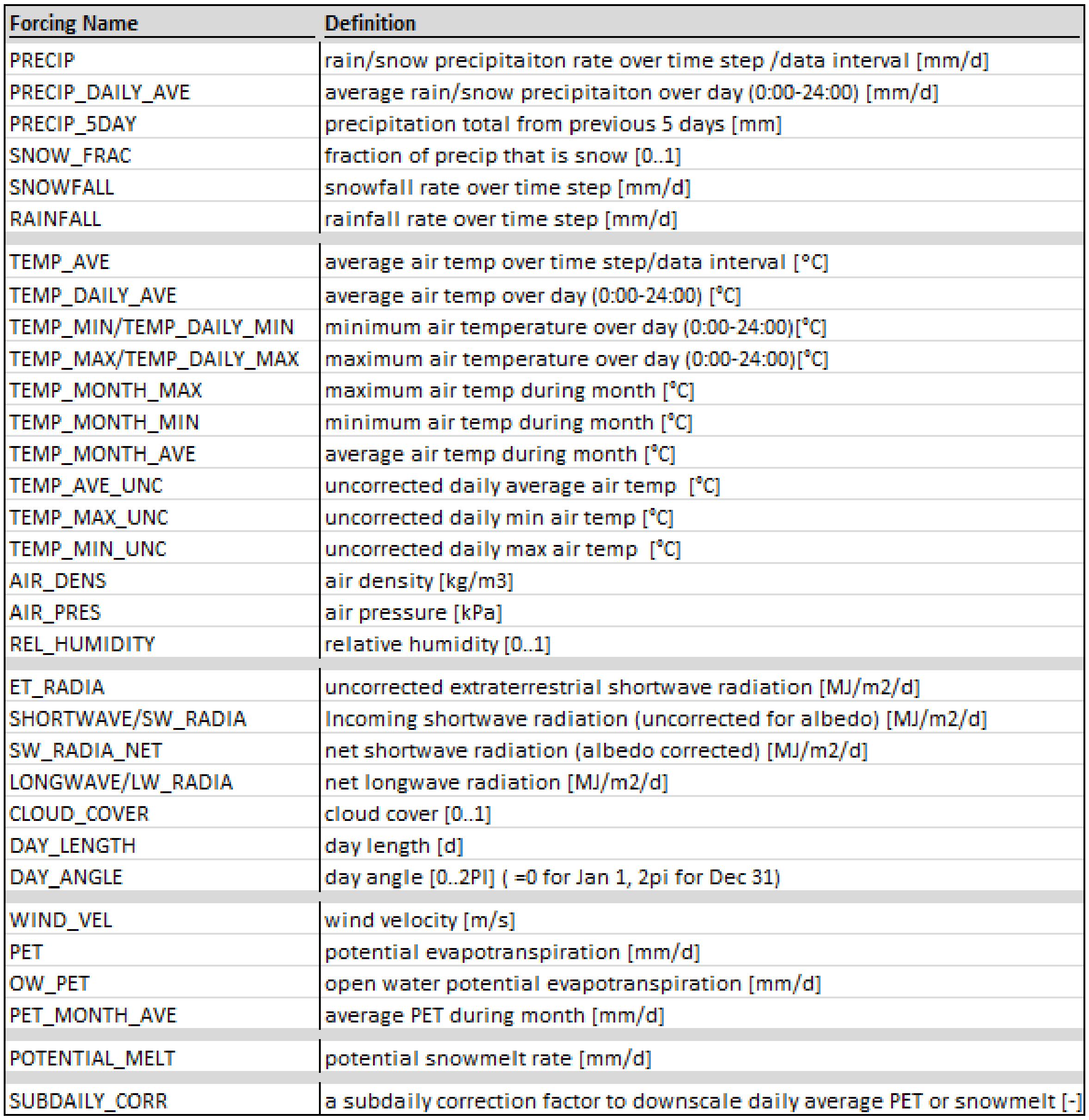
Appendix C

Reference Tables



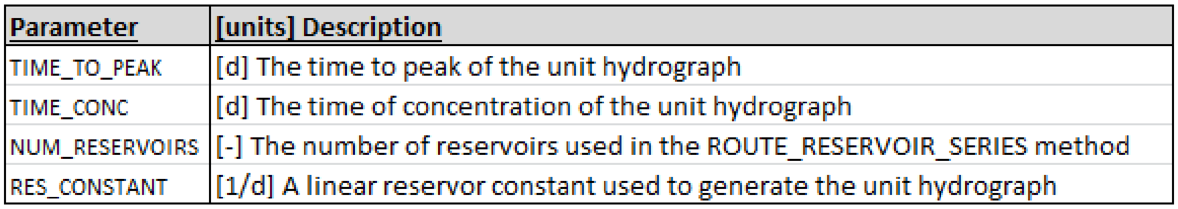
State Variable	[units] Description
Required Water Storage Variables	
SURFACE_WATER	[mm] Streams, rivers, rivulets - routed to basin outlet via in-catchment routing
ATMOSPHERE	[mm] atmosphere : recieves water only!!
ATMOS_PRECIP	[mm] atmosphere : provides water only!!
PONDED_WATER	[mm] water (melt & precip) waiting to infiltrate/runoff
Water Storage	
SOIL	[mm] Shallow subsurface/vadose zone
GROUNDWATER	[mm] Deep groundwater
CANOPY	[mm] liquid water on vegetation canopy
CANOPY_SNOW	[mm] snow on canopy
TRUNK	[mm] water stored in trunks of trees
ROOT	[mm] water stored in roots
DEPRESSION	[mm] depression/surface storage
WETLAND	[mm] deep depression storage
SNOW	[mm] frozen snow depth (mm SWE : snow water equivalent)
SNOW_LIQ	[mm] liquid snow cover
GLACIER	[mm] Glacier melt/reservoir storage
GLACIER_ICE	[mm] Glacier ice - typically assumed to be infinite reservoir.
Convolution storage	
CONVOLUTION	[mm] Convolution storage - for conceptual models with intermediate convolution steps
CONV_STOR	[mm] Convolution sub-storage - tracks internal water mass for convolution
Temperature / Energy Storage	
SURFACE_WATER_TEMP	[C] Temperature of surface water
SNOW_TEMP	[C] Temperature of snow
COLD_CONTENT	[C] Cold content of snowpack
GLACIER_CC	[C] cold content of glacier
SOIL_TEMP	[C] Temperature of soil
CANOPY_TEMP	[C] Temperature fo canopy
Auxilliary Variables	
SNOW_DEPTH	[mm] Snow depth - surrogate for density
PERMAFROST_DEPTH	[mm] depth of permafrost
SNOW_COVER	[0..1] fractional snow cover
SNOW_ALBEDO	[-] Snow Surface albedo
CROP_HEAT_UNITS	[-] cumulative crop heat units
Memory Variables	
CUM_INFIL	[mm] Cumulative infiltration to topsoil
CUM_SNOWMELT	[mm] Cumulative snowmelt
Transport Variables	
CONSTITUENT	[mg/m2] chemical species or tracer
CONSTITUENT_SRC	[mg/m2] chemical species or tracer cumulative source
CONSTITUENT_SW	[mg/m2] chemical species dumped to surface water
CONSTITUENT_SINK	[mg/m2] chemical species or tracer cumulative sink (e.g., decay)

Figure C.1: All state variables currently available in Raven. This list of state variables is supported by the `:HydroProcesses` commands and `:CustomOutput` commands, amongst others.



Forcing Name	Definition
PRECIP	rain/snow precipitaiton rate over time step /data interval [mm/d]
PRECIP_DAILY_AVE	average rain/snow precipitaiton over day (0:00-24:00) [mm/d]
PRECIP_5DAY	precipitation total from previous 5 days [mm]
SNOW_FRAC	fraction of precip that is snow [0..1]
SNOWFALL	snowfall rate over time step [mm/d]
RAINFALL	rainfall rate over time step [mm/d]
TEMP_AVE	average air temp over time step/data interval [°C]
TEMP_DAILY_AVE	average air temp over day (0:00-24:00) [°C]
TEMP_MIN/TEMP_DAILY_MIN	minimum air temperature over day (0:00-24:00)[°C]
TEMP_MAX/TEMP_DAILY_MAX	maximum air temperature over day (0:00-24:00)[°C]
TEMP_MONTH_MAX	maximum air temp during month [°C]
TEMP_MONTH_MIN	minimum air temp during month [°C]
TEMP_MONTH_AVE	average air temp during month [°C]
TEMP_AVE_UNC	uncorrected daily average air temp [°C]
TEMP_MAX_UNC	uncorrected daily min air temp [°C]
TEMP_MIN_UNC	uncorrected daily max air temp [°C]
AIR_DENS	air density [kg/m3]
AIR_PRES	air pressure [kPa]
REL_HUMIDITY	relative humidity [0..1]
ET_RADIA	uncorrected extraterrestrial shortwave radiation [MJ/m2/d]
SHORTWAVE/SW_RADIA	Incoming shortwave radiation (uncorrected for albedo) [MJ/m2/d]
SW_RADIA_NET	net shortwave radiation (albedo corrected) [MJ/m2/d]
LONGWAVE/LW_RADIA	net longwave radiation [MJ/m2/d]
CLOUD_COVER	cloud cover [0..1]
DAY_LENGTH	day length [d]
DAY_ANGLE	day angle [0..2PI] (=0 for Jan 1, 2pi for Dec 31)
WIND_VEL	wind velocity [m/s]
PET	potential evapotranspiration [mm/d]
OW_PET	open water potential evapotranspiration [mm/d]
PET_MONTH_AVE	average PET during month [mm/d]
POTENTIAL_MELT	potential snowmelt rate [mm/d]
SUBDAILY_CORR	a subdaily correction factor to downscale daily average PET or snowmelt [-]

Figure C.2: All forcing functions currently available in Raven. This list of forcing functions is supported by the `:Data`, `:MultiData`, `:CustomOutput`, and `:GaugeMultiData` commands, amongst others.



Parameter	[units] Description
TIME_TO_PEAK	[d] The time to peak of the unit hydrograph
TIME_CONC	[d] The time of concentration of the unit hydrograph
NUM_RESERVOIRS	[-] The number of reservoirs used in the ROUTE_RESERVOIR_SERIES method
RES_CONSTANT	[1/d] A linear reservor constant used to generate the unit hydrograph

Figure C.3: All subbasin parameters currently available in Raven. These parameters may be specified in the `:SubBasinParameters` command in the .rvh file

Appendix D

Template Files

The following section provides template .rvi files.

To do (10)

D.1 UBCWM Emulation

```
# ----------------------------------------------------------------
# Raven Template Input File
# UBC Watershed Model v5 Emulation
# ----------------------------------------------------------------
:StartDate    1991-10-01 00:00:00
:Duration     365
:TimeStep     24:00:00
#
:Method              ORDERED_SERIES
:Interpolation       INTERP_NEAREST_NEIGHBOR

:Routing             ROUTE_NONE
:CatchmentRoute      ROUTE_DUMP

:Evaporation         PET_MONTHLY_FACTOR
:OW_Evaporation      PET_MONTHLY_FACTOR
:SWRadiationMethod   SW_RAD_UBCWM
:SWCloudCorrect      SW_CLOUD_CORR_UBCWM
:SWCanopyCorrect     SW_CANOPY_CORR_UBCWM
:LWRadiationMethod   LW_RAD_UBCWM
:WindspeedMethod     WINDVEL_UBCWM
:RainSnowFraction    RAINSNOW_UBCWM
:PotentialMeltMethod POTMELT_UBCWM
:OroTempCorrect      OROCORR_UBCWM
:OroPrecipCorrect    OROCORR_UBCWM2
:OroPETCorrect       OROCORR_UBCWM
:CloudCoverMethod    CLOUDCOV_UBCWM
:PrecipIceptFract    PRECIP_ICEPT_USER
```

```
:MonthlyInterpolationMethod  MONTHINT_LINEAR_21

:SoilModel          SOIL_MULTILAYER 6
:SnapshotHydrograph
#
# -Processes-------------------------------------------------
:Alias TOP_SOIL      SOIL[0]
:Alias INT_SOIL      SOIL[1]
:Alias SHALLOW_GW    SOIL[2]
:Alias DEEP_GW       SOIL[3]
:Alias INT_SOIL2     SOIL[4]
:Alias INT_SOIL3     SOIL[5]
# -UBCWM EMULATION:------------------------------------------
:HydrologicProcesses
   :SnowAlbedoEvolve  SNOALB_UBCWM
   :SnowBalance       SNOBAL_UBCWM     MULTIPLE      MULTIPLE
   :Flush             RAVEN_DEFAULT    PONDED_WATER  INT_SOIL2 # moves snowmelt to fast runo
     :-->Conditional    HRU_TYPE IS GLACIER
   :GlacierMelt       GMELT_UBC        GLACIER_ICE   PONDED_WATER
   :Precipitation     PRECIP_RAVEN     ATMOS_PRECIP  MULTIPLE
   :SoilEvaporation   SOILEVAP_UBC     MULTIPLE      ATMOSPHERE
   :Infiltration      INF_UBC          PONDED_WATER  MULTIPLE
   :Flush             RAVEN_DEFAULT    SURFACE_WATER INT_SOIL2  # from infiltration to routi
   :GlacierInfiltration GINFIL_UBCWM   PONDED_WATER  MULTIPLE
   :Percolation       PERC_LINEAR_ANALYTIC      INT_SOIL      INT_SOIL2 #routing
   :Percolation       PERC_LINEAR_ANALYTIC      INT_SOIL2     INT_SOIL3 #routing
   :Baseflow          BASE_LINEAR      INT_SOIL3     SURFACE_WATER      #routing
   :Baseflow          BASE_LINEAR      SHALLOW_GW    SURFACE_WATER
   :Baseflow          BASE_LINEAR      DEEP_GW       SURFACE_WATER
   :GlacierRelease    GRELEASE_LINEAR  GLACIER       SURFACE_WATER
:EndHydrologicProcesses
```

See the Alouette tutorial example for a template .rvp file for UBCWM emulation, indicating all required parameters.

D.2 HBV-EC Emulation

```
# ------------------------------------------------
# Raven Input file
# HBV-EC Emulation
# ------------------------------------------------
# --Simulation Details -------------------------
:StartDate    1991-10-01 00:00:00
:Duration     365
:TimeStep     1.0
#
# --Model Details -------------------------------
:Method              ORDERED_SERIES
:Interpolation       INTERP_NEAREST_NEIGHBOR

:Routing             ROUTE_NONE
:CatchmentRoute      TRIANGULAR_UH

:Evaporation         PET_FROMMONTHLY
:OW_Evaporation      PET_FROMMONTHLY
:SWRadiationMethod   SW_RAD_DEFAULT
:SWCloudCorrect      SW_CLOUD_CORR_NONE
:SWCanopyCorrect     SW_CANOPY_CORR_NONE
:LWRadiationMethod   LW_RAD_DEFAULT
:RainSnowFraction    RAINSNOW_HBV
:PotentialMeltMethod POTMELT_HBV
:OroTempCorrect      OROCORR_HBV
:OroPrecipCorrect    OROCORR_HBV
:OroPETCorrect       OROCORR_HBV
:CloudCoverMethod    CLOUDCOV_NONE
:PrecipIceptFract    PRECIP_ICEPT_USER
:MonthlyInterpolationMethod  MONTHINT_LINEAR_21

:SoilModel           SOIL_MULTILAYER 3

# --Hydrologic Processes-------------------------
:Alias       FAST_RESERVOIR SOIL[1]
:Alias       SLOW_RESERVOIR SOIL[2]
:LakeStorage SLOW_RESERVOIR
:HydrologicProcesses
  :SnowRefreeze      FREEZE_DEGREE_DAY  SNOW_LIQ        SNOW
  :Precipitation     PRECIP_RAVEN       ATMOS_PRECIP    MULTIPLE
  :CanopyEvaporation CANEVP_ALL         CANOPY          ATMOSPHERE
  :CanopySnowEvap    CANEVP_ALL         CANOPY_SNOW     ATMOSPHERE
  :SnowBalance       SNOBAL_SIMPLE_MELT SNOW            SNOW_LIQ
    :-->Overflow     RAVEN_DEFAULT      SNOW_LIQ        PONDED_WATER
  :Flush             RAVEN_DEFAULT      PONDED_WATER    GLACIER
    :-->Conditional HRU_TYPE IS GLACIER
  :GlacierMelt       GMELT_HBV          GLACIER_ICE     GLACIER
  :GlacierRelease    GRELEASE_HBV_EC    GLACIER          SURFACE_WATER
  :Infiltration      INF_HBV            PONDED_WATER     MULTIPLE
  :Flush             RAVEN_DEFAULT      SURFACE_WATER    FAST_RESERVOIR
    :-->Conditional HRU_TYPE IS_NOT GLACIER
  :SoilEvaporation   SOILEVAP_HBV       SOIL[0]          ATMOSPHERE
  :CapillaryRise     RISE_HBV           FAST_RESERVOIR  SOIL[0]
  :LakeEvaporation   LAKE_EVAP_BASIC    SLOW_RESERVOIR  ATMOSPHERE
  :Percolation       PERC_CONSTANT      FAST_RESERVOIR  SLOW_RESERVOIR
  :Baseflow          BASE_POWER_LAW     FAST_RESERVOIR  SURFACE_WATER
  :Baseflow          BASE_LINEAR        SLOW_RESERVOIR  SURFACE_WATER
:EndHydrologicProcesses
#
:AggregatedVariable FAST_RESERVOIR AllHRUs
:AggregatedVariable SLOW_RESERVOIR AllHRUs
```

See the Alouette2 tutorial example for a template .rvp file for HBV-EC emulation, indicating all required parameters.

D.3 GR4J Emulation

```
# --------------------------------------------------
# Raven Input file
# GR4J Emulation
# --------------------------------------------------
:StartDate           2000-01-01 00:00:00
:Duration            365
:TimeStep            1.0

:Method              ORDERED_SERIES
:Interpolation       INTERP_NEAREST_NEIGHBOR

:Routing             ROUTE_NONE
:CatchmentRoute      ROUTE_DUMP

:Evaporation         PET_DATA
:RainSnowFraction    RAINSNOW_DINGMAN
:PotentialMeltMethod POTMELT_DEGREE_DAY
:OroTempCorrect      OROCORR_SIMPLELAPSE
:OroPrecipCorrect    OROCORR_SIMPLELAPSE

:SoilModel           SOIL_MULTILAYER  4

# --Hydrologic Processes-------------------------
:Alias PRODUCT_STORE      SOIL[0]
:Alias ROUTING_STORE      SOIL[1]
:Alias TEMP_STORE   SOIL[2]
:Alias GW_STORE           SOIL[3]
:HydrologicProcesses
 :Precipitation            PRECIP_RAVEN       ATMOS_PRECIP   MULTIPLE
 :SnowTempEvolve           SNOTEMP_NEWTONS    SNOW_TEMP
 :SnowBalance              SNOBAL_CEMA_NIEGE  SNOW            PONDED_WATER
 :OpenWaterEvaporation     OPEN_WATER_EVAP    PONDED_WATER ATMOSPHERE
 :Infiltration             INF_GR4J           PONDED_WATER   MULTIPLE
 :SoilEvaporation          SOILEVAP_GR4J      PRODUCT_STORE   ATMOSPHERE
 :Percolation              PERC_GR4J          PRODUCT_STORE   TEMP_STORE
 :Flush                    RAVEN_DEFAULT      SURFACE_WATER   TEMP_STORE
 :Split                    RAVEN_DEFAULT      TEMP_STORE      CONVOLUTION[0] CONVOLUTION[1]
 :Convolve                 CONVOL_GR4J_1      CONVOLUTION[0]  ROUTING_STORE
 :Convolve                 CONVOL_GR4J_2      CONVOLUTION[1]  TEMP_STORE
 :Percolation              PERC_GR4JEXCH      ROUTING_STORE   GW_STORE
 :Percolation              PERC_GR4JEXCH2     TEMP_STORE      GW_STORE
 :Flush                    RAVEN_DEFAULT      TEMP_STORE      SURFACE_WATER
 :Baseflow                 BASE_GR4J          ROUTING_STORE   SURFACE_WATER
:EndHydrologicProcesses
```

See the Irondequoit tutorial example for a template .rvp file for GR4J emulation, indicating all required parameters.

Bibliography

Barry, D., Parlange, J.-Y., Li, L., Jeng, D.-S., Crappert, M., 2005. Green ampt approximations. Advances in Water Resources 28 (1), 1003–1009.

Bergstrom, S., 1995. Computer models of watershed hydrology. Water Resources Publications, Highlands Ranch, Colorado, Ch. The HBV Model, pp. 443–476.

Brown, D. M., Bootsma, A., 1993. Crop heat units for corn and other warm season crops in ontario. Tech. Rep. Fact sheet 93-119, Ontario Ministry for Food and Rural Affairs.

Chow, V., Maidment, D., Mays, L., 1988. Applied Hydrology. McGraw-Hill.

Clark, M. P., Slater, A. G., Rupp, D. E., Woods, R. A., Vrugt, J. A., Gupta, H. V., Wagener, T., Hay, L. E., 2008. Framework for Understanding Structural Errors (FUSE): A modular framework to diagnose differences between hydrological models. Water Resources Research 44, w00B02, doi:10.1029/2007WR006735.

Dingman, S., 2002. Physical Hydrology. Waveland Press Inc.

Green, W. H., Ampt, G. A., 1911. Studies on soil physics. The Journal of Agricultural Science-Doi:10.1017/S0021859600001441.

Gupta, H. V., Kling, H., Yilmaz, K. K., Martinez, G. F., 2009. Decomposition of the mean squared error and nse performance criteria: Implications for improving hydrological modelling. Journal of Hydrology 377, 80–91.

Hamon, W., 1961. Estimating potential evapotranspiration. Journal of Hydraulics Division, Proceedings of the ASCE 871, 107–120.

Hargreaves, G., Samani, Z., September 1982. Estimating potential evapotranspiration. Journal of the Irrigation and Drainage Division, ASCE 108 (3), 225–230.

Hargreaves, G., Samani, Z., 1985. Reference crop evapotranspiration from temperature. Applied Engineering in Agriculture 1 (2), 96–99.

Hedstrom, N. R., Pomeroy, J. W., 1998. Measurements and modelling of snow interception in the boreal forest. Hydrological Processes 12, 1611–1625.

Kuzmin, P., 1957. Hydrophysical investigations of land waters. Vol. 3.

Leavesley, G., Stannard, L., 1995. Computer models of watershed hydrology. Water Resources Publications, Highlands Ranch, Colorado, Ch. The precipitationrunoff modeling system PRMS, p. 281310.

Liu, J., Sun, G., McNulty, S. G., Amatya, D., 2005. A comparison of sic potential evapotranspiration methods for regional use in the southeastern United States. Journal of the American Water Resources Association 41 (3), 621–633.

Makkink, G. F., 1957. Testing the Penman formula by means of lysimeters. J. Inst. of Water Eng. 11, 277–288.

Monteith, J., 1965. The State and Movement of Water in Living Organisms. Vol. 17. Academic Press Inc., New York, Ch. Evaporation and environment, pp. 205–234.

Penman, H., 1948. Natural evaporation from open water, bare soil and grass. Royal Society of London Proceedings, Series A 193, 120–145.

Perrin, C., Michel, C., Andrassian, V., 2003. Improvement of a parsimonious model for streamflow simulation. Journal of Hydrology 279 (1-4), 275–289.

Priestley, C., Taylor, R., 1972. On the assessment of surface heat flux and evaporation using large-scale parameters. Monthly Weather Review (100), 81–92.

Quick, M., 1995. Computer models of watershed hydrology. Water Resources Publications, Highlands Ranch, Colorado, Ch. The UBC Watershed Model, pp. 233–280.

Quick, M., 2003. Ubc watershed model documentation. Tech. rep., University of British Columbia.

Rutter, A., Kershaw, K., Robins, P., Morton, A., January 1971. A predictive model of rainfall interception in forests, 1. derivation of the model from observations in a plantation of corsican pine. Agricultural Meteorology 9, 367–384.

Schroeter, H., 1989. GAWSER Training Guide and Reference Manual. Grand River Conservation Authority (GRCA), Waterloo, ON.

Soil Conservation Service, 1986. Urban hydrology for small watersheds, 2nd ed. Tech. Rep. Tech. Release No. 55 (NTIS PB87-101580), U.S. Department of Agriculture, Washington, D.C.

Turc, L., 1961. Evaluation de besoins en eau d'irrigation, et potentielle. Ann. Agron. 12, 13–49.

U.S. Dept. of Commerce, O. o. T. S., 1956. Snow Hydrology. Washington, D.C.

Williams, J., 1969. Flood routing with variable travel time or variable storage coefficients. Trans. ASAE 12 (1), 100–103.

Wood, E., Lettenmaier, D., Zartarian, V., 1992. A land-surface hydrology parameterization with subgrid variability for general circulation models. Journal of Geophysical Research 97 (D3), 2717–2728.

Yin, X., 1997. Optical air mass: Daily integration and its applications. Meteorology and Atmospheric Physics 63 (3), 227–233, doi:10.1007/BF01027387.

To do. . .

☐ 1 (p. 25): Forcing estimator code development section

☐ 2 (p. 56): Add Abstraction Section

☐ 3 (p. 68): Sub-daily temperature orographic and lapsing temp ranges not yet described

☐ 4 (p. 76): PET Orographic Effects - PRMS Method

☐ 5 (p. 87): SUBDAILY_UBC description

☐ 6 (p. 90): In-channel Transport Routing section

☐ 7 (p. 102): Update the Required Parameters for Model Operation Options Table - do in Excel (import as image)

☐ 8 (p. 107): Move this table to Excel, import as figure

☐ 9 (p. 108): Create a table for Required Parameters for Hydrological Processes Options

☐ 10 (p. 139): Report default Raven ”vanilla” configuration