Raven:
User's and Developer's Manual v4.0

the Raven development team

Contributions to Raven, its utilities, documentation, testing, and source code, have been made by numerous students and faculty at the University of Waterloo and collaborators elsewhere, including Rezgar Arabzadeh, Erfan Amiri, Richard Arsenault, Nandita Basu, Genevieve Brown, Matthew Chernos, Rob Chlumsky, Élise Devoie, Sarah Grass, Ming Han, David Huard, Susan Huang, Koen Jansen, Ayman Khedr, Anjali Kuruppu, Francis Lapointe, Konhee Lee, Ryan MacDonald, Stuart Pearson, Arelia Schoenenberg, Nick Sgro, Mayhar Shafii, Trevor James Smith, Silvie Spraakman, Graham Stonebridge, Mahkameh Taheri, Kai Tsuruta, Arnejan van Loenen, Connor Werstuck, and Cloud Zhang. Andrew P. Snowdon contributed much to the early library of hydrologic process algorithms, and to the global numerical solver. Dr. Juliane Mai has provided MacOS/Linux compilation support and implemented the code allowing for NetCDF gridded data support. Leland Scantlebury provided the coupling interface with the MODFLOW-USG groundwater model (to hopefully emerge in a forthcoming version). Andre Della Libera Zanchetta updated the model for BMI compatibility. Martin Serrer from the National Research Council (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, particularly for the UBCWM emulation. The base software architecture and much of the core program was developed by James R. Craig at the University of Waterloo, with support through hundreds of discussions with Dr. Bryan Tolson and the late Dr. Eric Soulis.

Contents

Chapter 1

Introduction

This document describes the design and operation of the Raven hydrologic modelling framework, a software package for watershed simulation. 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 hydrologic 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 flexibility; Raven is able to use a wide variety of algorithms to represent each individual component of the water 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 precisely emulate (and augment) a number of existing hydrologic models - also known as model mimicry Jansen et al. (2021). Raven has achieved level 1 (near-perfect) emulation of the UBC Watershed Model (Quick, 1995), Environment Canada's version of the HBV model (Lindström et al., 1997; Hamilton et al., 2000), HBV-Light Seibert, HMETS (Martel et al., 2017), MOHYSE (Fortin and Turcotte, 2006), SAC-SMA (Burnash et al., 1973), HYMOD (Moore, 2007; Wagener et al., 2001), 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, and/or described within various hydrology texts, such as Dingman's *Physical Hydrology* (Dingman, 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 hydrologic

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 hydrologic 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.

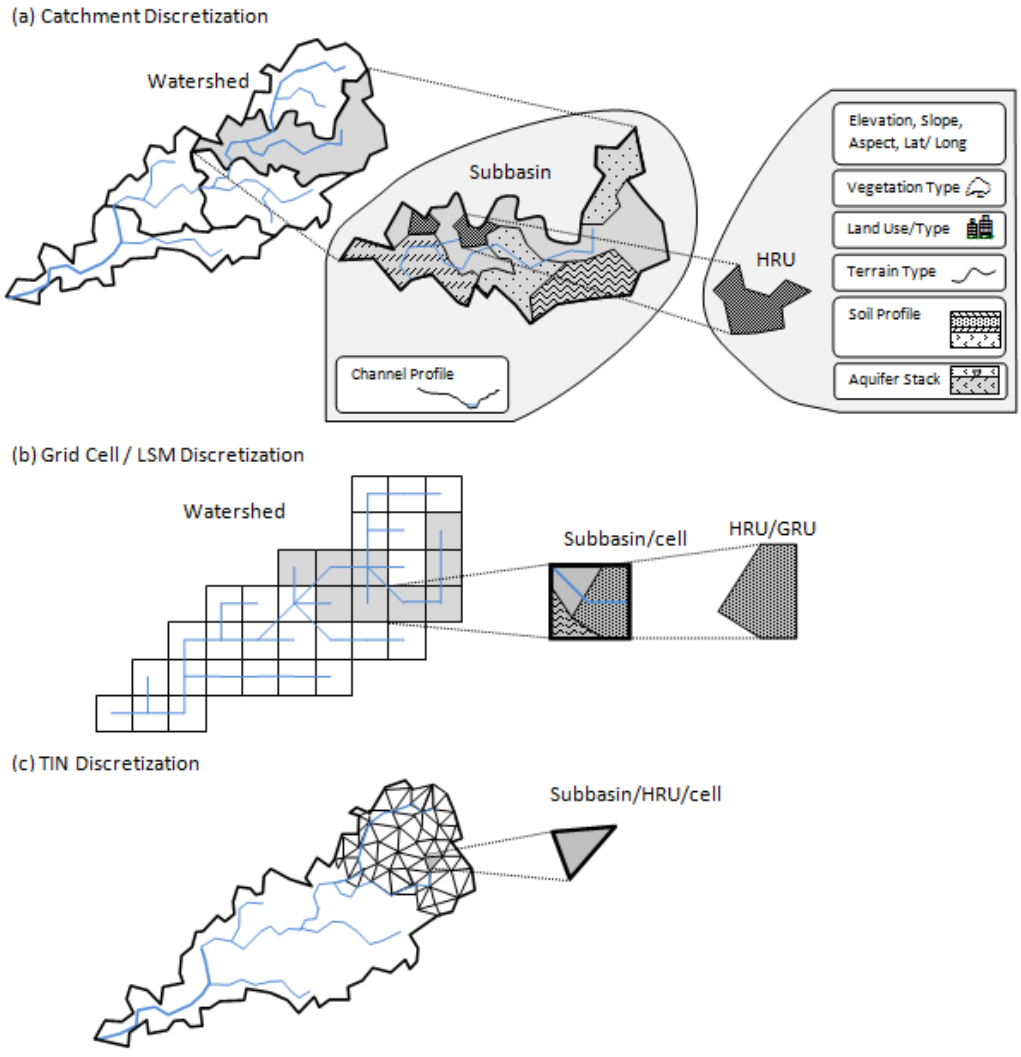

Figure 1.1: Land surface partitioning in Raven

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 membership in these classes, i.e., most of the properties belong to the class, not the HRU, enabling the solution of a finely discretized model (>10,000

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

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

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

Table 1.2: Common hydrologic processes that may be included in a Raven model

HRUs) without generating an equally large number of unknown parameters.

As a generalization of standard methods used to represent shallow soils in hydrologic 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.

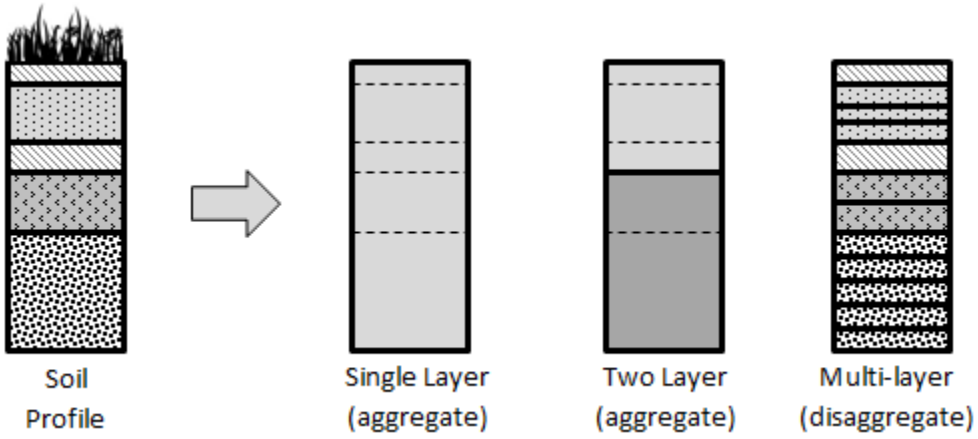

Figure 1.2: Translation of soil profiles to soil models. Properties are aggregated or disaggregated depending upon specified vertical resolution of soil model

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 lumped 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 dozens of individual storage compartments and forced using measured hourly radiation, wind velocity, and relative humidity. 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.

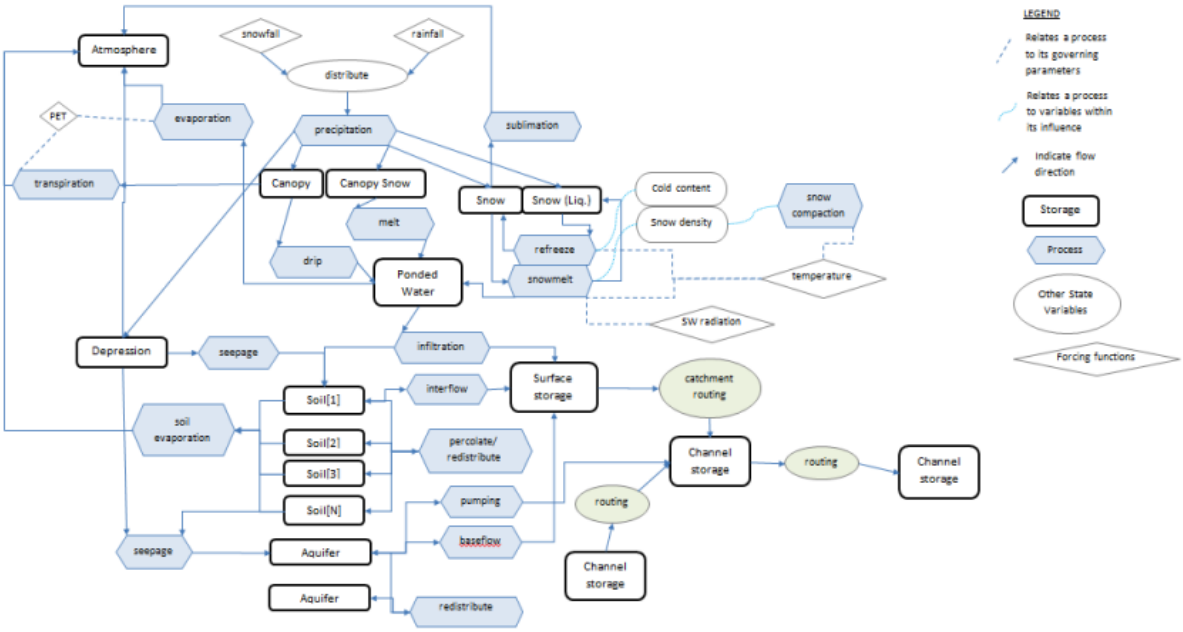

Figure 1.3: Example flowchart of the water balance in a Raven model. Note that individual processes and storage compartments may be added or subtracted from this schematic.

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 hydrologic 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 constituents (contaminants/tracers/enthalpy) 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 simulation 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 hydrologic 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 hydrologic 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.

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.,

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

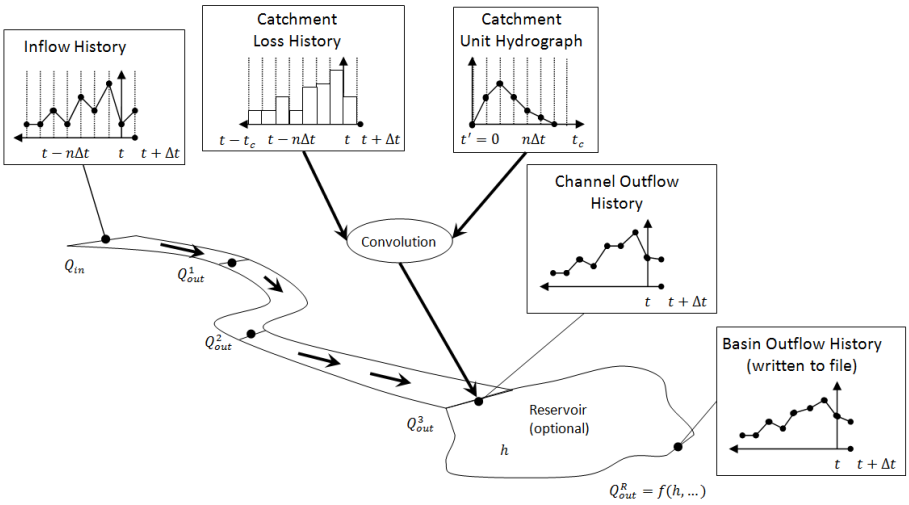

Figure 1.4: The general routing model of Raven

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/lake routing. At the outlet of each subbasin, the option exists to specify a managed reservoir or natural lake 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.

Irrigation demand, diversions, and plant discharges Man-made extractions and injections of water are incorporated directly into the mass balance formulations at reach inflows, reach outflows, or reservoirs in the form of user-specified time series of discharge and/or rule curves. These can be constrained by environmental minimum flows.

1.3 Watershed Conceptual Model

The critical feature of Raven is that it does not make any assumptions about the functioning of the watershed. That is the modeler's 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 posisble by providing reasonable defaults, an intuitive file interface, and a

large library of hydrologic and algorithmic options. In addition, it allows users to assess the utility and appropriateness of their conceptual model and revise it as needed.

1.4 Citing Raven

The preferred citation for the Raven hydrological modelling framework in a research manuscript or technical report is the following paper in Environmental Modelling and Software:

> Craig, J.R., G. Brown, R. Chlumsky, W. Jenkinson, G. Jost, K. Lee, J. Mai, M.Serrer, M. Shafii, N. Sgro, A. Snowdon, and B.A. Tolson, Flexible watershed simulation with the Raven hydrological modelling framework, Environmental Modelling and Software, 129, 104728, doi:10.1016/j.envsoft.2020.104728, July 2020

To cite Raven technical details for technical reports, this manual may be cited as:

> Craig, J.R., and the Raven Development Team, Raven user's and developer's manual (Version 4.0), URL: https://raven.uwaterloo.ca/ (Accessed xxx, 2025).

If you are using a model emulation housed within Raven, then that model configuration should be explicitly cited to give credit where credit is due. For instance, if you use the unmodified UBC watershed model configuration from appendix F, the preferred citation format would be (e.g.,) "we used the UBC Watershed Model (Quick, 1995) as implemented in the Raven hydrologic modelling framework v4.0 (Craig et al., 2020)". If the base conceptual model was significantly revised and/or merged with other tools, then it may be acceptable to refer to the model as (e.g.,) UBCWM* or UBCWM-Raven. In all cases, it is recommended to provide the model version number and input files in supplementary material so the details and attribution of the model components may be identified.

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. Input 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

2.1.1 Microsoft Windows

On Windows, there is no formal installation package for Raven unless you wish to use NetCDF-formatted input or output, and no special programs are libraries are required to operate Raven. Simply download the Windows executable `Raven.exe` and unzip to a local drive. If you require NetCDF support (for instance, if you are using NetCDF-format gridded climate data), instructions for installing the NetCDF libraries are below.

> NOTE: just clicking on Raven.exe to test it will lead to a briefly appearing command window which closes immediately. This is expected behaviour. Raven is a command-line executable and requires command line arguments.

(Optional, but recommended) If you wish to configure Raven to run from a single executable anywhere on your system, follow the steps below. Note that you will require administrative permissions to modify files under Program Files and to edit Environment Variables.

1. unzip the Raven executable, and save it to a folder you wish to keep it in (such as `C:\Program Files\Raven`)
2. (Optional) if you wish to run the water management optimization capabilities in Raven, save the `lpsolve.dll` file to the same folder as the Raven executable (`Raven.exe`)
3. Open the Environment Variables menu by clicking the Windows button, and searching for 'edit environment variables'.
4. Under User variables at the top of the menu, click on the Path variable and click Edit

5. Add a new location that is the folder where the Raven executable is saved, e.g., `C:\Program Files\Raven\`, and click ok.
6. Test that `Raven.exe` may now be called from anywhere by running `Raven.exe` from Command Prompt in a folder with no such executable. if you see the `Raven_errors.txt` file created, this was successful!
7. If you need to update or change the version of Raven, you can simply replace the executable in your stored location. There is no need to change or re-define the Path environment variable once set.

> Only if you are using the Raven version with NetCDF support (i.e., for supporting gridded forcing data such as that generated in regional climate forecasts):
> Install the NetCDF 4 Library (without DAP) from
> https://www.unidata.ucar.edu/software/netcdf/docs/winbin.html.
> You then must ensure that the directory path of the installed NetCDF.dll file is in your PATH environment variable, similar to the steps above for the Raven executable.

2.1.2 MacOS and Unix/Linux

MacOS users are encouraged to re-compile the Raven source code for their machine. Assuming you have g++ installed on your machine, you may follow the sequence below:

1. (Optional) if using with NetCDF or lp_solve support, see below information box first. Specific instructions are included in the makefile for compiling with NetCDF support and/or compiling with support for water management optimization using the `lp_solve` library.
2. unzip the Raven source code, obtained from the website or github to a directory of your choice
3. open the terminal and move to the directory of the source code/makefile
4. enter the following command: `make clean` (this cleans up previous build output, and is only necessary if compiling a second time)
5. enter the following command: `make` (this creates the executable)
6. move the executable Raven.exe to your working directory
7. enter the command `chmod 774 Raven.exe`. This sets the permissions so you can execute the code.
8. test run Raven by running the following command: `./Raven.exe modelname -o ./output/`

> If you are using the Raven version with NetCDF support (i.e., for supporting gridded forcing data such as that generated in regional climate forecasts). MacOS and linux users, use the brew command `$ brew install netcdf` from your terminal (https://formulae.brew.sh/formula/netcdf) prior to compiling Raven using the makefile. You will have to adjust the makefile where indicated in the comments to ensure Raven is linked to the NetCDF libraries.

(Optional) To configure Raven to be run from anywhere on your MacOS system (similar instructions will apply for Linux/Unix systems), follow these steps:

1. compile the Raven executable as per the previous steps for MacOS

2. Open Terminal (can be found in the `/Applications/Utilities` folder or via a Spotlight search with Cmd + Space)
3. Enter the command `nano ~/.zshrc` to edit the zsh configuration file in the nano text editor.
4. Use the down arrow key to navigate to the bottom of the file.
5. Add two lines to the file where your/path/to/raven is replaced with your path to the Raven.exe executable:
   ```
   export RAVEN_HOME=''your/path/to/raven''
   export PATH=''RAVEN_HOME:\$PATH''
   ```
6. Press Ctrl + 0 together to save, then Enter to confirm. Finally Ctrl + X to exit the text editor.
7. Quit Terminal (using Cmd + Q or right clicking the app icon in the Taskbar and choosing Quit)
8. Test that `Raven.exe` may now be called from anywhere by running `Raven.exe` from Terminal in a folder with no such executable. if you see the `Raven_errors.txt` file created, this was successful!
9. If you need to update or change the version of Raven, you can simply replace the executable in your stored location. There is no need to change or re-define the Path environment variable once set.

Linux/Unix users may also have to compile from source code using the makefile provided. Alternately, they may use the flatpak utility, running the following command from the terminal:

```
flatpak install flathub ca.uwaterloo.Raven
```

which should install Raven with NetCDF. If flatpak is used, then all executable calls to Raven.exe in 2.3 must be replaced with calls to `flatpak run ca.uwaterloo.Raven`. This assumes users have flatpak installed on their system.

2.1.3 International Users

For all users outside of North America - you may have to temporarily change your number settings for Raven to operate correctly; Raven assumes numerical formats of 1,000.00 (i.e., comma as the thousands separator, and a period as the decimal separator)

2.1.4 Developers

For developers modifying the code, the source code can be compiled using the provided makefile (in g++), CMakeList.txt file (using CMake), or Visual Studio Community Edition 2022 solution file (in windows). Raven has been tested on several platforms and compilers, but if you run into errors which impede compilation, please reach out to the development team.

2.2 Input Files

In order to perform a simulation using Raven, the following five input files are required (where 'modelname' is the placeholder for the model name, often named after the watershed outlet, e.g., NicolaRiver):

- `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 hydrologic 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.1.3 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.

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 stream gauge and meteorological data may be imported with utilities in the RavenR package. The .rvh file is likely best prepared with the assistance of a GIS database which can be used to determine unique class combination and the topology of the watershed subbasins, or using the BasinMaker python library. Note that, if the size of .rvt or .rvh files becomes

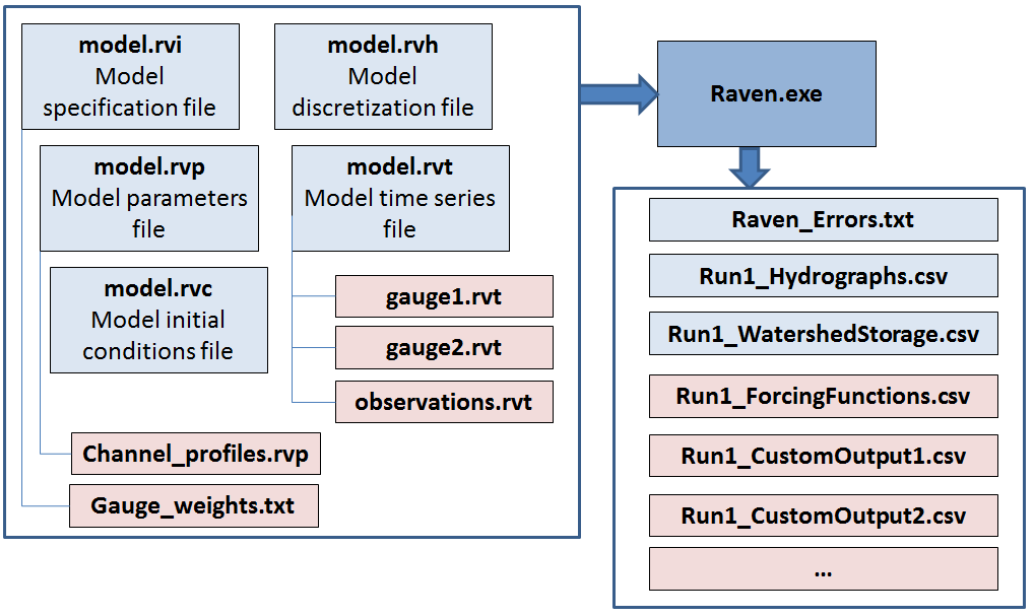

Figure 2.1: Standard input/output configuration of Raven. Light blue input files are required, light blue output files are the default output (which may be suppressed if desirable). The light red input files are files referred to by the primary input files, and are kept separate mostly for organization.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

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., in the windows command prompt,

```
> 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, .rvp (and optional .rvl/.rve) extensions. There are no special flags needed, just the name of the model.

Note that if you have configured Raven to be run from anywhere on your system, and your active directory is the same location as your model file directory, you can simply call Raven as:

```
> Raven.exe modelname
```

The command line also supports the following argument flags:

- `-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
- `-l {rvl_filename.rvl}` : specifies the rvl file location
- `-e {rve_filename.rve}` : specifies the rve file location
- `-r {runname}` : specifies the run name for the simulation
- `-m {mode character}` : specifies the run mode character
- `-n` : run in 'noisy mode'
- `-s` : run in 'silent mode'
- `-tt {shift}` : shift start time/endtime/assimilation time by `shift` hours
- `-we {runname}` : specifies warm start ensemble run name for EnKF
- `-template`: generates a template .rvp file given only an .rvi file, and does not run a simulation (equivalent to including the `:CreateRVPTemplate` command in the .rvi file).

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.1). Use of the `-o` flag is recommended, however. File paths and filenames with spaces are NOT supported from the command line.

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 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.

For MacOS or Linux users, note that the Raven.exe, despite its .exe extension, runs like any other command line tool. This can be run by opening the `terminal` application. The only difference then is the use of forward slashes rather than backward slashes, e.g.,:

```
machine:~ username$ Raven.exe modelname -o ./output/
```

2.4 Output Files

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

- `Hydrographs.csv` - the hydrograph output file
  Contains the flow rates, $Q(t)$ [m^3/s], at the outlets of specified subbasins in the watersheds (usually corresponding to those with stream gauges). Which subbasin hydrographs are reported is specified in the .rvh file.
- `WatershedStorage.csv` - the watershed storage file
  Contains watershed-averaged water storage in all of the simulated 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 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 hydrologic process over time. (enabled using the `:WriteMassBalanceFile` command.)
- `ForcingFunctions.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)
- `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.
- `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 and is automatically generated if reservoirs are present.
- `ReservoirMassBalance.csv` - reservoir mass balance history file
  Stores the time history of reservoir inflows and outflows for all simulated reservoirs. Requires at least one reservoir in the model.
- `Demands.csv` - irrigation demands file
  Stores the time history of irrigation, environmental flow constraints, and unmet demand (enabled using the `:WriteDemandFile` command)
- `WaterLevels.csv` - water levels file
  Stores the time history of stream/river water levels at the outlet of each gauged basin.

- `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 `:WriteExhaustiveMB` command.)
- State (`.rvc`) 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 time stamp, e.g., `RunName_state_2001-10-01.rvc`, and may be used as initial conditions for later simulation runs.

Lastly, the extremely flexible `:CustomOutput` command can be used to indicate that Raven should track and store in .csv, NetCDF, or .tb0 flat files any user-specified parameter, state variable, or mass/energy flux 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). Other special outputs are detailed in appendix B.

Additional output files generated by the transport and stream temperature simulation routines are discussed in chapter 7.

2.4.1 Alternative .nc (NetCDF) Output Format

For compatibility with software based on NetCDF files (e.g., the Deltares FEWS forecasting system) it is also possible to write outputs in that format. The `:WriteNetCDFFormat` command should be present in the .rvi file (prior to any `:CustomOutput` commands) if the NetCDF output should be written instead of .csv files. Details are documented in appendix B.3.

2.4.2 Alternative .tb0 (Ensim) Output Format

For compatibility with the Green Kenue™ software interface, the option is also available to generate output in .tb0 (Green Kenue™ tabular) format. Custom output will be written to a .tb0 table output file if the `:WriteEnsimFormat` command is present in the .rvi file. Note that not all outputs are generated in .tb0 format, and **.tb0 support may be deprecated in future versions of Raven**.

2.5 Building a Model

Base model: rvi and rvp files
It is recommended that users initially start with an existing model template such as the UBCWM, HBV-EC, HMETS, MOHYSE, or Canadian Shield model configurations reported in appendix F. Once you get more experienced with Raven, you may have existing model configurations that you have found work well on similar landscapes to those you have simulated previously.

Template .rvp files can be generated by running the .rvi template file (and only the .rvi file) with the `–template` command line argument, which builds a hollow .rvp file with all of the parameters necessary for simulation using the particular model configuration specified in the .rvi file. Only the .rvi file is required for this. Reasonable initial parameter value ranges are reported in appendix A.2, but manual calibration will be required in pretty much all cases.

Landscape discretization: rvh files
The best approach for generating the subbasin delineation and HRU delineation (i.e., the .rvh file) is to

use a GIS program such as ArcMap, SAGA, TauDEM or GRASS; or using the BasinMaker python library developed at the University of Waterloo. The latter is embedded in the Magpie google collab utility downloadable from the Raven website. These tools enable the generation of basin geometry from a hydrologically conditioned DEM and additionally enable the overlay of map layers to determine HRU areas. Basin outlets should at the very least correspond to locations of known streamgauges, but would also be added at the outlets of hydrologically important lakes and reservoirs, at major stream junctions, or at locations which divide the network into hydrologically similar landscapes (e.g., separating mountains from foothills). HRUs are commonly generated by reclassification of raster- or vector-based land use maps overlain with subbasin boundaries, though these may be additionally overlain with soil maps and/or elevation bands, where appropriate, using a union operation. Slope, aspect, elevation, latitude, longitude, and subbasin membership for each unique vegetation/land use/soil profile parcel would then be determined by spatial averaging and geometric operations within the GIS. Note that HRUs do not have to be spatially contiguous. The mechanics on how this is done vary from application to application. If the resultant HRU map is in vector format, its data table may be exported to a text file then rearranged using any number of text editing, spreadsheet, or scripting tools to be converted to .rvh format. Likely the hardest part to automate here is the specification of subbasin connectivity (i.e., the downstream subbasin ID for each subbasin), which typically would be done by eye.

Initial conditions: rvc file
The simplest initial conditions file can be empty. This can be modified later, but most storage compartments in the model when run in continuous (rather than event or forecasting) mode have a spin-up period that can compensate for an initially dry watershed. Groundwater storage and initial reservoir stage are two notable exceptions that may have to be carefully initialized, as they are slow to run up.

Meteorological inputs and observations: rvt files
The populating of the time series (.rvt) file is generally a problem of finding appropriate and available data and converting it to the .rvt format, which is relatively straightforward. Of course, there are many complications arising in infilling missing forcing data, interpreting what data is useful, and determining how to interpolate spatial data. Users can start with a single meteorological gauge initially and readily add or remove meteorological gauges in a minimally invasive manner.

The RavenR R library provides a number of utilities for directly downloading Canadian meteorological and stream gauge data and converting them to .rvt format. These RavenR tools are also embedded in the Magpie workflow tool available from the Raven website.

Iterative improvement
Once you get a base model created and running, then you can start swapping out individual processes, moving towards a landscape-appropriate model with complexity justified by the amount of data available at the site. A large amount of meteorological data and hydrograph data can justify a quite complex model with finely discretized landscape and more complicated model configuration. Modifying model configuration should be assessed one step a time, confirming each process addition or swapout of forcing function representation lead to a more appropriate or otherwise more effective representation of watershed hydrology. Note that modifying and iteratively evaluating model structure in this way can be a time consuming and arduous process, so many users will choose to stick with a fairly standard model configuration with a few minor tweaks.

2.6 Calibration, Visualization, and Uncertainty Analysis

Unlike many hydrologic 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:

- RavenView
  An online map-based visualization tool for viewing Raven model input and output.
  https://raven.uwaterloo.ca/RavenView/RavenView.html
- RavenR
  A set of R utilities available from the Raven website. Requires the R open-source software environment.
  https://raven.uwaterloo.ca/RavenR.html
- RavenPy
  A set of Python utilities available on github. Developed by Ouranos to support the PAVICS-Hydro online hydrological modelling environment.

- BasinMaker
  A python library for manipulating basin discretization files available for download across North America.
  http://hydrology.uwaterloo.ca/basinmaker/index.html
- 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...
  https://www.civil.uwaterloo.ca/envmodelling/Ostrich.html
- R
  An open-source software environment for statistical computing and scientific graphics. Available at https://cran.r-project.org/

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

2.7 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.

Automated Calibration
Multiple tools are provided within Raven for supporting automatic calibration by other software packages. It is encouraged to use the algorithms within the Ostrich software package, and an example Ostrich-Raven configuration is provided with the Raven documentation. To constrain the calibration, it is recommended to allow Raven to generate the diagnostics used to build the objective function using one or more of the diagnostics described in section 8.2, which supports the provision of observation weights to (1) include a spinup period (2) represent a calibration period (3) represent a validation period (4) discount seasonal (e.g., winter) data during diagnostic calculation.

Other useful commands for calibration support include the ability to suppress all output but the diagnostics file (`:SuppressOutput`) and suppress all console output (`:SilentMode`). This maximizes the speed of repeated model application (output generation can be more than 90 percent of computational cost). Users also have the ability to override historical stream and reservoir flows and replace simulated hydrographs with observed hydrographs at locations within the stream network (`:OverrideReservoirFlow` for reservoirs and `:OverrideStreamflow` for stream gauges). Lastly, portions of the model may be calibrated independently by disabling the remainder of the model using the `:DisableHRUGroup` command.

Large Models
For larger models with considerable data inputs and outputs, it is suggested to lean on the power of the :RedirectToFile command to organize the data. For instance, in a large basin model, it is useful to have folders to store the observation data, meteorological gauge data, reservoir, and channel data and keep it separate from the main body of Raven model files. A sample file structure might look like:

```
model folder/
 ./channels/
 ./observations/
 ./output/
    ./run1/
    ./run2/
    ./run3/
 ./metdata/
 ./reservoirs/
 modelname.rvi
 modelname.rvt
 modelname.rvh
 modelname.rvp
 modelname.rvc
```

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).

Forecasting
For forecasting, standard practice would be to hindcast / spin-up the model for a period of time, often prior to winter to properly account for snow depths. The state of the model would be saved at the current date and used as a 'warm start' .rvc file for short-term forecasts fueled by weather forecasts, rather than meteorological gauge data, thus only the .rvt files and .rvc files are changed when moving from spinup to forecast, plus the start date and end date in the .rvi file. The initial state of the model (for instance snow depth, soil moisture, or upstream flows) could be corrected if real-time data are available to compensate for model errors by revising the .rvc state file. Operational choices can be evaluated, for instance, using the :OverrideReservoirFlow time series to control reservoir flows.

Subdomain simulation
Sometimes for computational expediency it is worthwhile to only simulate a subset of the watershed. Raven supports the ability to simulate part of a larger watershed model by disabling subbasins (and their constituent HRUs) using the :DisableSubBasinGroup command. This readily handles simulation of only headwater basins (attached to the rest of the model only downstream) or parallel basins (not attached). For basins in the middle of the model, the watershed subset may be appropriately simulated if

all inflows to the basin are user-specified via the `:BasinInflowHydrograph` command.

2.8 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.

Always Check the RavenErrors.txt file in the output directory first. Often, the error messages and warnings will contain sufficient information to diagnose and repair the problem. This is always the best first step.

Use the Forum Posting questions and answers to the online Raven forum at (https://www.civil.uwaterloo.ca/raven_forum/) ensures the whole community can learn. (Note: the forum has been mostly defunct since 2021 after spambots forced us to change the registration process, however, there are still hundreds of public answers to Raven modelling questions there.)

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

1. **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 .rvi file. Running the code in noisy mode 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 the problematic command commented out. If it does, there may be (a) improper command syntax or (b) a missing input parameter for the chosen method/algorithm or (c) erroneous input data linked to this method/algorithm that Raven is not currently able to detect.

2. **If the model runs to completion but generates clearly erroneous output** (i.e., NaN or -#inf in the output)
   This type of error is likely due to (a) an error in input which Raven did not detect (e.g, a parameter outside reasonable bounds like a porosity of 3.8); (b) a missing model parameter which Raven did not detect; or (c) an error in the Raven modelling library.

   (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, first ask questions on the Raven forum (http://www.civil.uwaterloo.ca/raven_forum/), then (if needed) send the problematic input files with a short description to jrcraig@uwaterloo.ca.

3. **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 hydrologic 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_ID`s 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, leading you to be off by a factor of 24.

4. **Turn on :NoisyMode**
If the issue is prior to simulation, or if the RavenErrors.txt warnings and errors are difficult to comprehend, adding the `:NoisyMode` and `:EndPause` command to the top of the .rvi file writes an extensive stream of information to the command prompt/console. Occasionally, this can direct you to a bad input command.

2.9 Version Notes

2.9.1 Major Changes from v3.8 to v4.0 (Feb 2025)

The following features have been added:

1. Water management optimization supported with the open-source `lp_solve` library. Support for complex operating-regime-specific management goals and constraints, looped commands, demand/delivery/return/diversion calculations subject to environmental flow constraints. Management goals can be expressed in terms of user-specified lookup tables, named constants, workflow variables, or time series. This functionality allows the user to codify general operating protocols enforced via solution of a linear programming problem in each time step. Operates at daily and hourly (or intermediate) time steps. This significant v4.0 upgrade is documented in section 4.5 and appendix A.6.
2. Support for frozen wetlands which can accumulate snow.
3. Support for `WATER`-type HRUs via special soil profile. These water HRUs, intended to represent stream reaches or other small water bodies, support canopy processes and use sub-canopy incoming longwave radiation (unlike `LAKE` type HRUs) while still disabling subsurface processes.
4. Transport simulation in models with `:Convolution` is now supported. This was the only remaining process which did not previously support transport.
5. Provision of time series as repeating patterns using `:AnnualPattern` command.
6. Support for netCDF output of stream temperature model results.
7. New diagnostics `ABSERR_RUN` (running average absolute error), `KGE_PRIME` (revised Kling-Gupta which uses COV instead of standard deviation), and `FUZZY_NASH` (Nash Sutcliffe Efficiency which only penalizes errors greater than a fixed percentage of observed flow). Lake area can now be provided as observation type; New diagnostic variable `TOTAL_SWE` for tracking sum of multilayer `SNOW` and `SNOW_LIQ`.
8. New command line support for writing RVP template file with the `-template` command line argument.
9. New command `:AllowSoilOverfill` to better support `:Overflow` process in conceptual/-compartment soil models. New command `:WriteNetReservoirInflows` which adds net inflow reporting to Hydrographs.csv output.

10. Support for nested :RedirectToFile commands, allowing this command to be used not only in the primary model rvt, rvp, and rvm files, but also files referenced from the primary model files.
11. Can now support subbasin and HRU IDs longer than 9 digits.
12. Major bug fixes: Issue with :LandUseChange command introduced in v3.8 fixed; failure of stream temperature model with zero-length streams; multiple minor bug fixes.

Minor backward compatibility issues:

1. :BasinInflowHydrograph2 is no longer subject to in-catchment routing; will lead to peakier outflow hydrographs. Diversion are now removed upstream of the reporting point (rather than from the water passed to the next reach). This means hydrographs.csv now reports post-diversion/demand delivery basin outflows.
2. UPSTREAM_OF option for :PopulateSBGroup now is inclusive of most downstream subbasin. May lead to different simulation results when subbasin groups are used for parameterization or conditional process application.

2.9.2 Major Changes from v3.7 to v3.8 (Jan 2024)

The following features have been added:

1. Support for Basic Model Interface (BMI) interoperability and compilation as linked library (major structural changes thanks to Andre Della Libera Zanchetta at the University of Manitoba)
2. Representation of simple lake freezing and snow on lakes with :LakeFreeze command
3. Support for :LWIncomingMethod and new methods LW_INC_SKYVIEW and LW_INC_DINGMAN
4. New PET estimation method PET_VAPDEFICIT for estimating above-stream evaporation
5. New parameters RELHUM_CORR and WIND_VEL_CORR for locally adjusting relative humidity and wind speed; formerly hard-coded parameters CAP_LAI_RATIO and SNOCAP_LAI_RATIO now exposed for adjusting relationship between canopy storage capacity and LAI. Parameter REFERENCE_FLOW_MU now exposed for estimating system-wide reference discharges.
6. New methods RAINSNOW_WANG and RAINSNOW_SNTHERM89 for rain-snow partitioning
7. Emulation of HYMOD2 model (Roy et al., 2017); addition of SOILEVAP_HYMOD2 evaporation routine.
8. Support for sensible heat transfer and groundwater mixing during in-catchment routing
9. New model diagnostic DIAG_SPEARMAN for calculating the Spearman ranked correlation coefficient
10. Improved support for Ensemble Kalman filter in FEWS environment, including perturbation of temperature
11. Improved handling of orographic corrections using gridded data with supplied cell elevations
12. New command :WriteLocalFlows to write local contribution to hydrograph in hydrographs.csv.
13. Addition of :TemperatureCorrection for gridded or gauged temperature forcing bias correction
14. Writing of Reservoir mass balance file in NetCDF format in addition to .csv format.
15. Ported entirely over to GitHub (thanks to Trevor James Smith at Ouranos)

16. Improved CMake compilation setup
17. major/minor bug fixes: repair of UBCWM subdaily snow balance, lake evaporation issues when LAKE_PET_CORR parameter was not specified or when :HRUID not specified for reservoir; all PET routines which should use daily temperature inputs no longer using time-step averaged temperatures (backwards incompatibility issue); proper handling of rainfall on reservoir when :LakeStorage is something other than SURFACE_WATER; repair of under-tested reservoir stage assimilation; correction of error in LW_RAD_DEFAULT (backwards compatibility issue)

2.9.3 Major Changes from v3.6 to v3.7 (May 2023)

The following features have been added:

1. Reconfiguration of and improvements to the EnKF assimilation implementation; ensemble member-specific inputs; timestamp-specific random seeds; support for RESERVOIR_STAGE as assimilation data. Previous EnKF implementation workflows will require updating.
2. Improved support for orographic correction of gridded data
3. Additional tools for defining HRU and SubBasin groups via intersection and merging
4. DAILY_KGE diagnostic metrics available for sub-daily time steps
5. Added RUNOFF diagnostic variable which can be reported as custom output
6. Support for grouped vegetation change and HRU type change from live file
7. Support for mixing gauged temperatures and gridded precipitation
8. Checks for missing/fill values in all NetCDF inputs
9. Updated CMake makefile
10. major/minor bug fixes: repaired issues with sub-daily NetCDF inputs being incorrectly read; memory error leading to issues when reservoirs dry out completely; NetCDF data gaps upon buffered read

2.9.4 Major Changes from v3.5 to v3.6 (Jan 2023)

The following features have been added:

1. simulation of basic geochemistry of transported constituents, including decay, transformation, equilibrium, and sorption; simulation of non-advecting constituents
2. improved support for distributed loading sources of constituents/pollutants
3. simulation of full in-lake energy balance described in section 7.5
4. new :AnnualEvents time series command for pulse events such as nutrient loadings
5. EnKF data assimilation
6. support for more rigorous thermal simulation of lakes and reservoirs, with energy balance reporting
7. full emulation of the Australian Water Balance Model (AWBM)
8. addition of new process algorithms: PRECIP_ICEPT_STICKY, and AWBM processes.
9. support for sewer conduits and circular channel/conduit cross-sections
10. reporting of water levels at catchment outlets

11. :AggregateDiagnostic command to report (e.g.) median NSE across multiple hydrographs
12. addition to :EvaluationPeriod command to only calculate diagnostics on high or low flows
13. new AGG_CUMULSUM custom output aggregation statistic (for, e.g., calculating monthly total precipitation)
14. new diagnostic DIAG_YEARS_OF_RECORD
15. new :TrapezoidalChannel command for simply characterizing channels
16. support for zero-area subbasins
17. support for wind speed, relative humidity, and net shortwave radiation as gridded inputs
18. writing of MassLoadings.csv output file using :WriteMassLoadings command
19. AET from reservoirs now reported in custom output
20. major bug fixes: support for southern hemisphere calculations, correct calculation of celerity when specifying subbasin-specific Manning's n via :SubBasinProperties or :SBGroupMultiplier commands. Fixed bug in :TargetBasin usage within control structures. Fixed bug in sublimation calculations leading to zero sublimation in all routines (present since v3.5). Fixed bug in reading elevations from NetCDF file. Potential melt now subjected to sub-daily corrections, where appropriate.

2.9.5 Major Changes from v3.0.4 to v3.5 (Jan 2022)

The following features have been added:

1. simulation of stream temperature via an unconditionally stable semi-analytical Lagrangian approach
2. general support for multiple reservoir control structures with complex operating regime rules
3. full Deltares-FEWS adaptor-free support (see appendix C)
4. level 1 (full) emulation of SAC-SMA, HYMOD, and HBV-Light, plus the corresponding hydrologic process algorithms (including SOILEVAP_PDM, INF_PDM, and SOILBAL_SACSMA)
5. support for conditional statements in all input using :RunMode and :IfModeEquals-:EndIfModeEquals commands.
6. Specified concentration/temperature time series, specified mass inflow time series (via new :MassLoading command), can now link sources to HRU groups
7. data assimilation of lake levels using :AssimilateReservoirStage command, improved support for direct insertion streamflow assimilation
8. support for WITHIN_SBGROUP option in :PopulateHRUGroup command
9. support for routing using external scripts and live file with ROUTE_EXTERNAL option
10. new diagnostics: R4MS4E, RTRMSE, RABSERR, PERSINDEX, NSE4, KLING_GUPTA_DEVIATION
11. support for overriding specific HRU gauge index using :SpecifyGaugeForHRU command
12. new potential melt algorithm POTMELT_RILEY (Riley et al., 1972)
13. minor bug fixes and QA/QC improvement; universal in-line commenting

14. minor modifications to source to support Borland C++ compiller

The reservoir mass balance has been changed to handle precipitation differently. Reported reservoir inflow in the Hydrographs.csv file used include both precipitation on the lake/reservoir, plus inflow. Now only inflow from upstream is included in this term and precipitation is reported separately in the ReservoirMassBalance.csv output file. In addition, precipitation on lakes is no longer processed through the in-catchment routing routine and instantaneously impacts reservoir levels.

Backward compatibility issues:

- a bug was fixed in the calculation of outflow from a lake-type reservoir. The `:WeirCoefficient` was hard-coded at 0.666 and unaffected by inputs. Any earlier models with lake-type reservoirs will need to be re-calibrated.
- Net shortwave radiation now discriminates between above and below canopy conditions. This impacts energy-balance based snowmelt calculations in forested regions using the `POTMELT_EB`, `POTMELT_RESTRICTED`, and `POTMELT_CRHM_EBSM` snowmelt algorithms

2.9.6 Major Changes from v3.0.1 to v3.0.4 (Feb 2021)

The following features have been added:

1. optimization for reading small windows of very large NetCDF files; support for incomplete NetCDF coverage over disabled HRUs
2. fixed `:SBGroupPropertyMultiplier` for `MANNINGS_N`
3. support for in-stream specified concentration sources
4. competitive ET supported by `:OpenWaterEvaporation` routines
5. improved sublimation routines, added vertical wind profile options
6. minor bug fixes and QA/QC improvement

2.9.7 Major Changes from v3.0 to v3.0.1 (Oct 2020)

The following features have been added:

1. addition of subbasin groups and subbasin group property specification
2. improved support for gridded PET and open water PET
3. minor bug fixes and QA/QC improvement

2.9.8 Major Changes from v2.9.1 to v3.0 (May 2020)

The following features have been added:

1. addition of `SW_CLOUD_CORR_ANNANDALE` algorithm for shortwave cloud cover correction
2. competitive ET now supported by all ET algorithms (can suppress for backward compatibility with `:SuppressCompetitiveET` command. AET magnitudes are now directly accessible as state variable in custom outputs. Additional option `:SnowSuppressesPET` can be used to suppress ET when snow is on the ground. New `PET_LINACRE` algorithm for PET estimation.
3. support for simple insertion data assimilation of streamflow with `:AssimilateStreamflow` command

4. new reservoir regulatory support:
    - `:ReservoirMinFlow`
    - `:ReservoirMaxFlow`
    - `:ReservoirDownstreamFlow`
    - `:ReservoirMaxQDecrease`
    - `:ReservoirOverflowMode`
5. support for irrigation demand and flow diversions:
    - `:IrrigationDemand`
    - `:ReservoirDownstreamDemand`
    - `:FlowDiversion`
    - `:FlowDiversionLookupTable`
    - `:DemandMultiplier`
    - `:UnusableFlowPercentage`
6. support for stage-volume and stage-area curves for lake-type reservoirs and groundwater seepage from reservoirs
7. the `:LatFlush` command now supports inter-basin lateral water transfer
8. support for period-ending NetCDF inputs using `:PeriodEndingNC` command and proper handling of NetCDF time zones, offset, and scale attributes
9. support for reference elevations for gridded forcings
10. improved and optimized support for massive stream/lake networks
11. writing of interpolation weights to external file using `:WriteInterpolationWeights` command
12. major bug fixes: correct handling of diffusive wave hydrograph for basins with very small travel times; using open water ET for reservoir mass balance; glitch in the determination of day-switching which impacts some sub-daily models using daily min/max temperature for estimation of melt energy/PET
13. minor bug fixes; improved QA/QC of inputs

The following backwards compatibility issues were introduced:

1. None

2.9.9 Major Changes from v2.9 to v2.9.1 (May 2019)

The following features have been added:

1. support of user-specified NetCDF attributes
2. support for non-standard calendars
3. support for non-midnight start time with NetCDF forcing
4. minor bug fixes; improved QA/QC of inputs

2.9.10 Major Changes from v2.8.1 to v2.9 (Feb 2019)

The following features have been added:

1. Support for level 1 (exact) emulation of the MOHYSE model (Fortin and Turcotte, 2006); multiple processes added.
2. Support for level 1 emulation of the HMETS model (Martel et al., 2017); multiple processes added.
3. Support for the PAVICs platform
4. Added `PET_OUDIN` PET estimation method and `:DirectEvaporation` support.
5. Added `:AnnualCycle` method for inputting cyclical time series.
6. Integrated multiple algorithms from the Cold Regions Hydrological Model (CRHM) (Pomeroy et al., 2007), including the PET approach of Granger and Gray (1989), the rain snow partitioning approach of Harder and Pomeroy (2013), two snow albedo evolution algorithms, a energy balance potential melt routine and a simple snow balance approach.
7. Support for model simulation start times other than midnight

The following backwards compatibility issues were introduced:

1. Due to a bug in the calculation of UTM zone from HRU latitudes and longitudes, the interpolation schemes for large models with multiple meteorologic gauges had an anisotropic bias (e.g., long, thin nearest-neighbor zones). This has been fixed, leading to a discrepancy between old and new model precipitation and temperature interpolation. This will not impact Raven models using single gauges, NetCDF gridded inputs, or user-specified gauge weights, only those using inverse distance or nearest neighbor interpolation.

2.9.11 Major Changes from v2.8 to v2.8.1 (Jul 2018)

The following features have been added:

1. FEWS-compliant NetCDF custom and standard output (Hydrographs.nc and WatershedStorage.nc)
2. support of deaccumulation of NetCDF input data
3. `RAINSNOW_HARDER` Rain/snow discrimination
4. fixes to relative file path handling, :VegetationChange/:LandUseChange/lake crest height/target stage bugs introduced in v2.8,

The following backwards compatibility issues were introduced:

1. relative file paths are now (correctly and consistently) with reference to the file specified rather than the model working directory

2.9.12 Major Changes from v2.7 to v2.8

The following features have been added:

1. Documentation improvements/Bug fixes/Improved QA/QC on model inputs
2. Significant speed improvements, particularly with NetCDF processing
3. Wetlands - new wetland HRU type; support for lateral flow to and from geographically-isolated and riparian wetlands; new depression flow and seepage routines for wetland depression storage

4. Lakes and Reservoirs - lake-type reservoirs for natural (unmanaged) run-of-river lakes; time-dependent weir control and rule curves (maximum, minimum, and target stages); spillway and underflow stage-discharge curve specification; advective transport of constituents and tracers through reservoirs; reservoir inflow and net inflow diagnostics; reservoir outflow override; reservoir mass balance reporting;
5. Inter-HRU Flow and Transport - generalized lateral flow support of water between HRUs and lateral advective transport of constituents;
6. Shortwave radiation on sloping surfaces - the default method now uses the robust analytical calculation approach of Allen et al. (2006) for estimating clear sky solar radiation
7. Improved Input/Output - custom flux reporting between/to/from any state variable, mixed gauge interpolation support (i.e., when temperature and precipitation reported at different gauges)
8. Other - HRU/subbasin disabling (only model a subset of the model); subbasin-specific Manning's n and slope; automated HRU group population; optimization and speed improvements (particularly for NetCDF input); running average NSE diagnostics; basin inflow hydrographs at downstream end of subbasin; vegetation-based PET correction; support of date-based net shortwave radiation input forcings;

The following backwards compatibility issues were introduced:

1. None

2.9.13 Major Changes from v2.6 to v2.7 (May 2017)

The following features have been added:

1. Documentation improvements/Bug fixes/Improved QA/QC on model inputs
2. Significantly improved support for flexible reservoir simulation and calibration - time-varying reservoir curves, unevenly spaced reservoir curves,
3. Support for gridded data in NetCDF format (see appendix A.4.7)
4. Improved place- and time-specific control over application of processes using the `:->Conditional` command, `:LandUseChange` command, and `:VegetationChange` command.
5. `:CreateRVPTemplate` command can be used to generate a template .rvp file from specified .rvi model configuration
6. Added a number of new diagnostics (`LOG_NASH`, `NASH_DERIV`, `KLING_GUPTA`)
7. Addition of the GAWSER-style snow balance and consolidation routine
8. Addition of US Army Corps snowmelt model
9. `RunName` can be specified from the command line

The following backwards compatibility issues were introduced:

1. None

2.9.14 Major Changes from v2.5 to v2.6 (May 2016)

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 `:SnowCorrection` and `:RainCorrection` parameters, 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

The Hydrologic Process Library

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

3.1 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 3.1. 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.

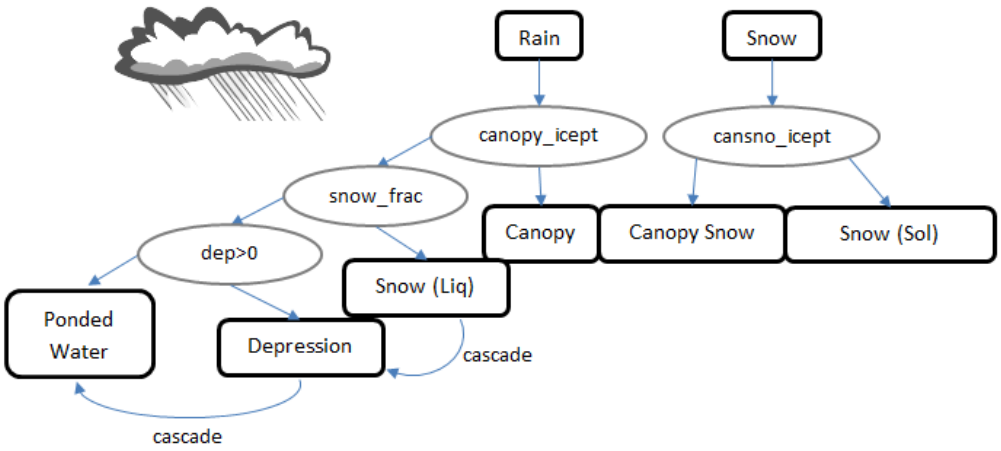

Figure 3.1: Partitioning of rainfall/snowfall to the appropriate surface storage compartments

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 full. 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 simulated,

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 the subsequent application of an infiltration or abstraction algorithm.

Special HRU types for open water, exposed rock, glaciers, and wetlands (determined by WATER, ROCK, GLACIER, and WETLAND prefixes on HRU soil profiles) are treated a bit differently than the default land HRU. In these HRUs, top soil is not active; therefore precipitation partitioning works a bit differently and (e.g.,) infiltration and soil evaporation routines are inactive.

For lake HRUs, all snow and rain is converted to liquid water and added directly to the SURFACE_WATER store ready to be routed downstream via in-catchment routing. Alternately, water can be sent to the LAKE_STORAGE store (if specified using the :LakeStorage command), where water release is delayed to the surface water network as controlled using (e.g.,) the :LakeRelease process (section 3.14). This latter approach is likely preferred for systems dominated by small lake features. Lake HRUs are defined as those with a zero-layer soil profile whose name begins with LAKE.

For glacier HRUs, glacier processes are enabled (they will not be applied in any non-glacier HRU). No soil processes (e.g., infiltration or percolation) are simulated beneath glaciers. Glacier HRUs are defined as those with a zero-layer soil profile whose name begins with GLACIER.

For wetland HRUs, all rain is converted to liquid water and added directly to the DEPRESSION store. Wetland HRUs are defined as those with a soil profile whose name begins with WETLAND. Snow which falls on a wetland is allowed to accumulate, assuming that the wetland is frozen. When it melts, it turns to ponded water and must be flushed to depression storage via proper commands in the :HydrologicProcesses block within the .rvi file.

For exposed rock HRUs, all throughfall and snowmelt is stored as PONDED_WATER. Since infiltration schemes don't function with rock-type HRUs, the user must provide an alternate mechanism to reach SURFACE_WATER (usually via a conditional :Flush process).

Example usage in the .rvi file:

```
:Precipitation RAVEN_DEFAULT ATMOS_PRECIP MULTIPLE
```

3.1.1 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:

$$\begin{aligned} \mathrm{LAI} &= (1-s) \cdot \mathrm{LAI_{max}} \cdot \mathrm{f_{LAI}(m)} \\ \mathrm{SAI} &= (1-s) \cdot \beta \cdot h_{veg} \end{aligned} \tag{3.1}$$

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 :SeasonalCanopyLAI command for each vegetation type, and β is the vegetation parameter SAI_HT_RATIO. Note that FOREST_COVERAG should be interpreted as the percentage of land covered in representative vegetation, and FOREST_SPARSENESS should be interpreted as a land use-based correction factor for vegetation density. 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_HEIGHT) and $f_{veg}(m)$ is the relative vegetation height correction by month m, specified using the :SeasonalCanopyHeight command in the .rvp file. In all cases, canopy interception of both snow and rain is limited by the capacity of the vegetation, determined by vegetation properties MAX_CAPACITY [mm] and MAX_SNOW_CAPACITY [mm]. The capacity at any given time varies with LAI:

$$C = C_{max} \cdot \frac{\mathrm{LAI}}{\mathrm{LAI_{max}}}$$

where C is the current capacity, C_{max} is the maximum capacity (e.g., MAX_CAPACITY). The total amount of intercepted snow or rain is not allowed to exceed this capacity for any of the interception algorithms.

Example usage in the .rvi file (above the verb|:HydrologicProcesses| block:

```
:PrecipIceptFract PRECIP_ICEPT_USER
```

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 default method used in Raven. The interception percentages are directly specified by the user, where θ_{rain} is the vegetation parameter RAIN_ICEPT_PCT and θ_{snow} 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:

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

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:

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

Hedstrom-Pomeroy method for snow (`PRECIP_ICEPT_HEDSTROM`)

If this method is chosen, the rain interception is the same as for `PRECIP_ICEPT_EXPLAI`, but the snow interception is handled as documented in Hedstrom and Pomeroy (1998).

No interception (`PRECIP_ICEPT_NONE`)

Interception does not occur/is not represented.

3.2 Infiltration / Runoff partitioning

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).

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 uninfiltrated water is typically treated as runoff and moved to `SURFACE_WATER`.

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

$$\begin{aligned} \phi_{max} &= Hn(1 - SF) \\ \phi_{tens} &= \phi_{max}(S_{fc} - S_{wilt}) \\ \phi_{fc} &= \phi_{max} S_{fc} \end{aligned} \tag{3.2}$$

where H is the soil layer thickness [mm], n is the porosity (soil property `POROSITY`), SF is the stone fraction (soil property `STONE_FRAC`, typically zero), 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`). These parameters may be user-specified or generated from percent sand/silt/clay.

Example usage in the .rvi file (within the `:HydrologicProcesses` command block):

```
:Infiltration INF_GREEN_AMPT PONDED_WATER MULTIPLE
```

Infiltration/Runoff Algorithms

Rational method (`INF_RATIONAL`)

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

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

where M_{inf} is the infiltration rate [mm/d], R is the rainfall/snowmelt rate [mm/d] (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:

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

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

$$S = 25400/CN - 254$$

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

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

where R is the rainfall/snowmelt rate [mm/d], 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 3.2), and soil moisture at the start of the time step, a state variable. k_{sat} is the saturated conductivity of the soil [mm/d], soil parameter HYDRAUL_COND. 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:

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

where R is the rainfall/snowmelt rate [mm/d]. F [mm], 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 3.2), and soil moisture at the start of the time step. k_{sat} is the saturated conductivity of the soil [mm/d], soil parameter HYDRAUL_COND. All parameters used are those associated with the top soil.

VIC method (INF_VIC)

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

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

where R is the rainfall/snowmelt rate [mm/d], ϕ_{soil} [mm] is the soil moisture content, ϕ_{max} is the maximum soil storage capacity as defined using equation 3.2, 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:

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

VIC/ARNO method (INF_VIC_ARNO)

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

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

where R is the rainfall/snowmelt rate [mm/d], 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] calculated using equation 3.2.

HBV method (INF_HBV)

The standard HBV model approach (Bergstrom, 1995; Lindström et al., 1997).

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

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] calculated using equation 3.2.

PRMS method (INF_PRMS)

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

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

where ϕ_{soil} is the soil layer water content [mm], ϕ_{tens} is the maximum tension storage [mm] calculated using equation 3.2, and F_{sat}^{max} is the maximum saturated area fraction (land use parameter MAX_SAT_AREA_FRAC).

UBC Watershed Model method (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

$$M_{inf} = R \cdot (1 - b_2)$$

where M_{inf} is limited by the soil storage deficit and 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):

$$b_2 = b_1 + (1 - b_1) \cdot FF$$

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

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

where ϕ_{soil} and ϕ_{max} are as defined in equation 3.2 and FF, the flash factor (which is constrained to vary between 0 and 1) is calculated as:

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

here, F_{imp} [-] is the land use parameter IMPERMEABLE_FRAC, V0FLAX [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 rainfall/snowmelt is distributed to groundwater (at rate M_{perc}), interflow (at rate M_{int}, and runoff M_{run} using the following expressions

$$\begin{aligned} M_{perc} &= \min\left(M_{max}^{perc}, R - M_{inf}\right) \cdot (1 - b_2) \\ M_{int} &= (R - M_{inf} - M_{perc}) \cdot (1 - b_2) \\ M_{run} &= b_2 \cdot R \end{aligned}$$

To summarize, 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} [mm/d], 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`, 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):

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

where $\alpha = \tanh(\phi_{pond}/\phi_{max})$, ϕ_{pond} [mm] is the ponded water storage after rainfall/snowmelt, ϕ_{soil} is the top soil layer water content [mm], and ϕ_{max} is the maximum topsoil storage [mm] calculated using equation 3.2.

HMETS infiltration method (`INF_HMETS`)

From the HMETS model (Martel et al., 2017):

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

where R is the rainfall/snowmelt rate [mm/d], α is the unitless land use parameter `HMETS_RUNOFF_COEFF`, ϕ_{soil} is the top soil layer water content [mm], and ϕ_{max} is the maximum soil storage [mm] calculated using equation 3.2.

AWBM infiltration method (`INF_AWBM`)

From the Australian Water Balance Model (AWBM) (Boughton, 2004). Not very compatible with other model structures. The surface is divided into three zones of different storage capacity (defined by the thickness and porosity of `SOIL[0]`, `SOIL[1]`, and `SOIL[2]`). The zones have an areal percentage coverage of a_1 (land surface parameter `AWBM_AREAFRAC1` [0..1]), a_2 (land surface parameter `AWBM_AREAFRAC2` [0..1]), and $a_3 = 1 - a_1 - a_2$. Each zone is filled to capacity (i.e., all water will infiltrate) until filled, in which case each zone releases excess water, which is partitioned into direct runoff and groundwater storage (where `SOIL[3]` represents this long term storage).

$$\begin{aligned} M_{excess} &= \max(a_1 R - \max(a_1\phi_1^{max} - \phi_1, 0)/\Delta t, 0) + \\ &\quad \max(a_2 R - \max(a_2\phi_2^{max} - \phi_2, 0)/\Delta t, 0) + \\ &\quad \max(a_3 R - \max(a_3\phi_3^{max} - \phi_3, 0)/\Delta t, 0) \\ M_{inf} &= R - M_{excess} \end{aligned}$$

$$
\begin{aligned}
M_{runoff} &= (1 - \text{BFI}) \cdot M_{excess} \\
M_{toGW} &= (\text{BFI}) \cdot M_{excess}
\end{aligned}
$$

3.3 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.

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`. Baseflow is rate-limited by the availability of soil/aquifer storage. Example usage in the .rvi file (within the `:HydrologicProcesses` command block):

```
:Baseflow BASE_LINEAR SOIL[4] SURFACE_WATER
```

Available Algorithms

Constant baseflow (`BASE_CONSTANT`)

A constant, specified rate of baseflow:

$$M_{base} = M_{max}$$

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

Linear storage (`BASE_LINEAR` or `BASE_LINEAR_ANALYTIC`)

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

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

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 time step (Δt):

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

The two methods are effectively equivalent for sufficiently small time steps, but the second is preferred for larger values of k ($k \cdot \Delta t > 0.1$).

Non-linear storage (`BASE_POWER_LAW`)

A very common approach used in a variety of conceptual models, including HBV Bergstrom (1995); Lindström et al. (1997). Here, the baseflow rate is non-linearly proportional to storage:

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

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):

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

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):

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

where ϕ_{ref} [mm] is the reference soil storage, the user-specified soil parameter GR4J_X3, which can be interpreted as a baseflow reference storage, ϕ_{soil} is the water storage [mm] in the soil or aquifer layer.

Topmodel baseflow method (BASE_TOPMODEL)

From TOPMODEL (Beven and Kirkby, 1979) as interpreted by (Clark et al., 2008):

$$M_{base} = M_{max} \cdot \frac{\phi_{max}}{n} \cdot \frac{1}{\lambda^n} \cdot \left(\frac{\phi_{soil}}{\phi_{max}}\right)^n$$

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 layer, ϕ_{max} is the maximum soil storage capacity, λ is the mean of the power-transformed topographic index [m] (terrain parameter TOPMODEL_LAMBDA), and n is the user-specified soil parameter BASEFLOW_N.

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.

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

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.

Threshold-based baseflow method (storage) (BASE_THRESH_STOR)

Here, baseflow doesn't commence until a threshold storage amount of the soil layer is met. Above the threshold, the outflow rate is linearly related to storage excess above the threshold.

$$M_{base} = K_2 \cdot \max\left(\phi_{soil} - \phi_{thresh}, 0\right)$$

where ϕ_{th} [mm] is the threshold soil storage at which baseflow begins (soil parameter STORAGE_THRESHOLD), K_2 is the soil parameter BASEFLOW_COEFF2 [1/d].

3.4 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.

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. Percolation is rate-limited by the availability of soil/aquifer storage and by the capacity of the receptor 'to' compartment. Example usage in the .rvi file Example usage in the .rvi file (within the `:HydrologicProcesses` command block):

```
:Percolation PERC_LINEAR SOIL[0] SOIL[1]
:Percolation PERC_LINEAR SOIL[1] SOIL[2]
```

Available Algorithms

Constant percolation (`PERC_CONSTANT`)

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

$$M_{perc} = M_{max}$$

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

Corrected linear percolation (`PERC_GAWSER`)

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

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

where M_{max} is the soil parameter `MAX_PERC_RATE`, ϕ_{soil} [mm] is the moisture content of the soil layer, and the other moisture contents are defined in equation 3.2. All parameters refer to that of the 'from' soil compartment.

Linear percolation (`PERC_LINEAR`)

Percolation is proportional to soil water content:

$$M_{perc} = k\dot{\phi}_{soil}$$

where k [1/d] is the soil parameter `PERC_COEFF` and ϕ_{soil} [mm] is defined in equation 3.2. 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:

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

where M_{max} [mm/d] is the soil parameter `MAX_PERC_RATE`, n is the soil parameter `PERC_N` and ϕ_{soil} [mm] and ϕ_{max} [mm] are defined in equation 3.2. All parameters refer to that of the 'from' soil compartment.

PRMS percolation method (`PERC_PRMS`)

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

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

where M_{max} [mm/d] is the soil parameter `MAX_PERC_RATE`, n is the soil parameter `PERC_N` and ϕ_{soil}, ϕ_{tens}, and ϕ_{max} [mm] are defined in equation 3.2. All parameters refer to that of the 'from' soil compartment.

Sacramento percolation method (`PERC_SACRAMENTO`)

Percolation is given by the following expression:

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

where M_{max}^{base} is the saturated baseflow rate (soil parameter `MAX_BASEFLOW_RATE`), α is soil parameter `SAC_PERC_ALPHA`, γ is the soil parameter `SAC_PERC_EXPON`, and ϕ_{soil} and ϕ_{max} are defined in equation 3.2. All parameters refer to that of the 'from' soil compartment, unless they have the ^{to} superscript.

GR4J percolation method (`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 (Perrin et al., 2003):

$$M_{perc} = -x_2 * (\min(\phi_{soil}/x_3, 1.0))^{3.5}$$

where x_2 is the soil parameter `GR4J_X2` and x_3 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.

3.5 Interflow

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

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. Interflow is rate-limited by the availability of soil/aquifer storage. Example usage in the .rvi file Example usage in the .rvi file (within the :HydrologicProcesses command block):

```
:Interflow INTERFLOW_PRMS SOIL[1] SURFACE_WATER
```

Available Algorithms

PRMS interflow method (INTERFLOW_PRMS)

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

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

where M_{max} is the maximum interflow rate (soil parameter MAX_INTERFLOW_RATE), ϕ_{soil} is the moisture content (in mm) of the draining soil, and ϕ_{tens}, and ϕ_{max} are defined in equation 3.2. All parameters refer to that of the 'from' soil compartment.

3.6 Soil Evapotranspiration

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

Soil evaporation always moves water between `SOIL[m]` and `ATMOSPHERE` units. Which soil layers are subjected to evapotranspiration depends on the soil structure model specified by the `:SoilModel` command and the particular evapotranspiration algorithm. It is rate-limited by the availability of soil/aquifer storage and by the capacity of the atmosphere to absorb water vapour. All of these algorithms update the total actual evapotranspiration (AET) from the HRU, and competitive ET is supported such that this algorithm competes with canopy and open water evaporation - the PET rate is reduced by the AET after application.

Example usage in the .rvi file (within the `:HydrologicProcesses` command block):

```
:SoilEvaporation SOILEVAP_HBV SOIL[0] ATMOSPHERE
```

In all notation below, PET refers to the potential evapotranspiration determined by one of the forcing function estimators of section 5.4. In all cases, this PET may be modified by the soil parameter `PET_CORRECTION`, which only modifies PET in these algorithms.

Available Algorithms

Uncorrected evaporation algorithm (`SOILEVAP_ALL`)

Water is removed from soil at the maximum rate until there is no water remaining:

$$M_{evap} = \text{PET}$$

where PET is the potential evapotranspiration rate [mm/d].

Linear evaporation-saturation (`SOILEVAP_HBV` or `SOILEVAP_TOPMODEL`)

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

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

where PET is the potential evapotranspiration rate [mm/d], and ϕ_{soil} [mm] and ϕ_{tens} [mm] are defined in equation 3.2. The HBV model uses an additional snow correction, such that ET is zero in non-forested areas if snow depth is non-zero.

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):

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

where PET is the potential evapotranspiration rate [mm/d], γ is the soil parameter `VIC_EVAP_GAMMA`, and ϕ_{soil} [mm] and ϕ_{max} [mm] are defined in equation 3.2.

Linear evaporation-storage (`SOILEVAP_LINEAR`)

Actual evapotranspiration is linearly proportional to topsoil storage, up to a maximum of PET:

$$M_{evap} = \min\left(\alpha \cdot \phi_{soil}, \text{PET}\right)$$

where PET is the potential evapotranspiration rate [mm/d], and ϕ_{soil} [mm] is defined in equation 3.2, and α [1/d] is the land use parameter `AET_COEFF`. This is used in the MOHYSE model Fortin and Turcotte (2006).

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

Soil ET [mm/d] 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.

$$M^U_{evap} = \text{PET} \cdot \xi_U \cdot \min\left(\frac{\phi^U_{soil}}{\phi^U_{tens}}, 1\right) \tag{3.3}$$

$$M^L_{evap} = \text{PET} \cdot \xi_L \cdot \min\left(\frac{\phi^L_{soil}}{\phi^L_{tens}}, 1\right) \tag{3.4}$$

where U and L refer to the upper and lower layers, respectively, and ϕ_{soil} [mm] and ϕ_{tens} [mm] are defined in equation 3.2. Currently, ξ_L and ξ_U are hardcoded as 0.3 and 0.7, respectively.

Sequential 2-layer evaporation (`SOILEVAP_SEQUEN`)

Daily soil ET [mm/d] is linearly proportional to the soil saturation; the top layer storage is exhausted first, then ET can be withdrawn from the lower layer.

$$M^U_{evap} = \text{PET} \cdot \min\left(\frac{\phi^U_{soil}}{\phi^U_{tens}}, 1\right) \tag{3.5}$$

$$M^L_{evap} = (\text{PET} - M^U_{evap}) \cdot \min\left(\frac{\phi^L_{soil}}{\phi^L_{tens}}, 1\right) \tag{3.6}$$

where U and L refer to the upper and lower layers, respectively, and ϕ_{soil} [mm] and ϕ_{tens} [mm] are defined in equation 3.2.

UBCWM approach (`SOILEVAP_UBC`)

Evaporation is controlled by the soil moisture deficit, $\phi_{max} - \phi_{soil}$, where ϕ_{max} is defined in equation 3.2, and is corrected for effective saturated area.

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

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

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

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):

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

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.

Ontario Crop Heat Unit Method (`SOILEVAP_CHU`)

From the Ontario crop heat unit method (Brown and Bootsma, 1993):

$$M_{evap} = \frac{\text{CHU}}{\text{CHU}_{mat}} \cdot \text{PET}$$

where CHU is the crop heat unit value (calculated using the `CHU_ONTARIO` hydrologic crop evolution process), and CHU_{mat} is the vegetation property `CHU_MATURITY`.

HYPR soil/wetland evaporation method (`SOILEVAP_HYPR`)

From the HYPR model for representing prairie landscapes (Ahmed et al., 2020), intended to be used in conjunction with the `ABST_PDMROF` abstraction routine, which represents depression storage as a probability distribution on the landscape. This process algorithm is unique in that it handles both evaporative losses from the soil and the losses from the depression storage. The soil evaporation rate is calculated in the same manner as the HBV model, where soil ET is at PET if storage exceeds the tension storage, then is linearly proportional to the soil saturation:

$$M^*_{evap} = \text{PET} \cdot \min\left(\frac{\phi_{soil}}{\phi_{tens}}, 1\right)$$

where PET is the potential evapotranspiration rate [mm/d], and ϕ_{soil} [mm] and ϕ_{tens} [mm] are defined in equation 3.2. The percentage of the landscape covered by depression storage is calculated as in Mekonnen et al. (2014):

$$F_p = F_{max} \cdot \left(\frac{\phi_{dep}}{\phi_{dmax}}\right)^n$$

where F_{max} is the land use parameter `MAX_DEP_AREA_FRAC`, n is the land use parameter `PONDED_EXP`, ϕ_{dep} is the depression storage in mm, and ϕ_{dmax} is the maximum depression storage on the landscape, `DEP_MAX` (mm).

$$\begin{aligned} M^d_{evap} &= (1 - F_p) \cdot M^*_{evap} \\ M^s_{evap} &= F_p \cdot \text{PET}_{OW} \end{aligned}$$

where PET_{OW} is the open water evaporation rate determined from the `:OW_Evaporation`-specified method. The first term is evaporation from the soil, the second term is evaporation from depression storage.

AWBM evaporation method (`SOILEVAP_AWBM`)

From the Australian Water Balance Model (AWBM) (Boughton, 2004). Not very compatible with other model structures. The surface is divided into three zones of different storage capacity (defined by the thickness and porosity of `SOIL[0]`, `SOIL[1]`, and `SOIL[2]`). The zones have an areal

percentage coverage of a_1 (land surface parameter `AWBM_AREAFRAC1` [0..1]), a_2 (land surface parameter `AWBM_AREAFRAC2` [0..1]), and $a_3 = 1 - a_1 - a_2$. Each zone is subject to evaporation until it is emptied.

$$\begin{aligned} M_{evap}^1 &= a_1 \text{PET} \\ M_{evap}^2 &= a_2 \text{PET} \\ M_{evap}^3 &= a_3 \text{PET} \end{aligned}$$

HYMOD2 evaporation method (`SOILEVAP_HYMOD2`)

From the revised HYMOD2 model of Roy et al. (2017). Intended to be used in conjunction with `INF_PDM` infiltration algorithm, also used by HYMOD2.

$$M_{evap} = K \cdot \text{PET} \tag{3.7}$$

where

$$K = K_{max} \cdot \left(G + (1 - G) \cdot \left(\frac{c^*}{c_{max}} \right)^c \right) \tag{3.8}$$

and c^* is calculated as done in the PDM method of Moore (2007):

$$c^* = c_{max} * \left(1.0 - \left(1.0 - \frac{\phi}{\phi_{max}} \right)^{1.0/(b+1.0)} \right) \tag{3.9}$$

where b is the land use parameter `PDM_B`, c is the land use parameter `HYMOD2_EXP`, G is the land use parameter `HYMOD2_G` between 0 and 1 determining the minimum resistance to ET, and K_{max} is the land use parameter `HYMOD2_Kmax`, an ET resistance parameter values between 0 and 1. Here, c_{max} is calculated as $(b + 1)\phi_{max}$ and ϕ and ϕ_{max} are the actual and maximum topsoil storage, respectively.

3.7 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, which may also implicitly include lateral groundwater flow processes in a sloping domain.

Capillary rise occurs between `SOIL` units, depending upon the soil structure model specified by the `:SoilModel` command. The user typically has to specify the 'to' and 'from' storage compartments. Capillary rise is rate-limited by the availability of soil/aquifer storage and by the capacity of the receptor 'to' compartment. Example usage in the .rvi file (within the `:HydrologicProcesses` command block):

```
:CapillaryRise CRISE_HBV SOIL[1] SOIL[0]
```

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; Lindström et al., 1997):

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

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 3.2. All parameters refer to that of the 'to' soil compartment.

3.8 Soil Balance

Some algorithms solve the entire subsurface mass balance problem (infiltration/percolation/baseflow, etc.) collectively, and these individual processes cannot be separated from the whole. The soil balance process therefore represents the collective simulation of all of these processes redistributing water in the soils.

Soil balance will move water between multiple SOIL units, but may also simulate runoff (moving water to SURFACE_WATER) and/or evaporation (moving water to ATMOSPHERE). The number of soils represented is particular to the algorithm chosen. Example usage in the .rvi file (within the :HydrologicProcesses command block):

```
:SoilBalance SOILBAL_SACSMA MULTIPLE MULTIPLE
```

Available Algorithms

Sacramento Soil Moisture Accounting (SOILBAL_SACSMA)

> This algorithm collectively represents infiltration, redistribution of water between conceptual soil units, and runoff/quickflow/baseflow release to the surface water, based upon the Sacramento model Burnash et al. (1973). This is a near-exact emulation of the original SAC-SMA code, and designed to be used in conjunction with the SOILEVAP_SACSMA soil evaporation algorithm. The algorithm is rather complicated, and therefore only the basic functioning of the model is documented here. Users are referred to either the source code or https://www.nws.noaa.gov/ohd/hrl/nwsrfs/users_manual/part2/_pdf/23sacsma.pdf for details.

The SOILBAL_SACSMA model only works with a 7 'layer' soil model. SOIL[0] and SOIL[1] represent the upper zone tension and free storage compartments, respectively. SOIL[2], SOIL[3], and SOIL[4] represent (respectively) the tension storage, primary free storage, and secondary free storage in the lower zone of the SAC-SMA conceptual model. Lastly, SOIL[5] represents a special storage corresponding to the saturated region of soil near a surface water body (referred to as ADIMC in the original SAC-SMA documentation) and SOIL[6] represents deep groundwater storage, which is never released to the surface.

The SAC-SMA algorithm requires 9 parameters. Land use parameters include MAX_SAT_AREA_FRAC, which determines the maximum percentage of ground which is regularly saturated, IMPERMEABLE_FRAC which determines the degree of direct runoff (no infiltration occurs beneath impermeable cover), and BF_LOSS_FRACTION, the percentage of baseflow redirected to deep groundwater. Soil parameters include three percolation coefficients associated with SOIL[1] (upper zone free storage): SAC_PERC_ALPHA, SAC_PERC_EXPON, and SAC_PERC_PFREE (ZPERC, REXP, and PFREE in original SAC-SMA parlance). Also used are three different linear baseflow coefficients (BASEFLOW_COEFF) associated with SOIL[1] (UZK, quickflow from the upper zone), SOIL[3] (LZPK, baseflow from primary lower zone storage), and SOIL[4] (LZSK, baseflow from secondary lower zone storage).

3.9 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. Canopy evaporation always occurs between `CANOPY` and `ATMOSPHERE` units. Canopy evaporation is rate-limited by the availability of canopy storage. All of these algorithms update the total actual evapotranspiration (AET) from the HRU, and competitive ET is supported such that this algorithm competes with soil and open water evaporation - the PET rate is reduced by the AET after application.

Example usage in the .rvi file (within the `:HydrologicProcesses` command block):

```
:CanopyEvaporation CANEVP_RUTTER CANOPY ATMOSPHERE
```

Available Algorithms

Maximum canopy evaporation (`CANEVP_MAXIMUM`)

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

$$M_{evap} = \text{PET} \cdot F_c \cdot (1 - f_s)$$

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

Complete canopy evaporation (`CANEVP_ALL`)

All moisture on the canopy evaporates instantaneously, i.e., all intercepted precipitation is sent back to the atmosphere. This is also the default behaviour if no canopy is present.

Rutter canopy evaporation (`CANEVP_RUTTER`)

From (Rutter et al., 1971):

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

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.

3.10 Canopy Drip

Canopy drip is the loss of liquid water from canopy to land surface, typically due to the impacts of wind. Canopy drip always occurs between CANOPY and PONDED_WATER units and is rate-limited by the availability of canopy storage. Example usage in the .rvi file (within the :HydrologicProcesses command block):

```
:CanopyDrip CANDRIP_RUTTER CANOPY PONDED_WATER
```

Available Algorithms

Rutter canopy drip (CANDRIP_RUTTER)

Moisture on the canopy which exceeds storage (given by vegetation parameter MAX_CAPACITY, mm) falls instantaneously to the ground.

Slowdrain canopy drip (CANDRIP_SLOWDRAIN)

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

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

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

3.11 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.

Abstraction always moves water from the PONDED_WATER state variable to the DEPRESSION storage state variable, but in some cases may also generate runoff (e.g., the ABST_PDMROF algorithm. Example usage in the .rvi file:

```
:Abstraction ABST_PERCENTAGE PONDED_WATER DEPRESSION
#or
:Abstraction ABST_PDMROF PONDED_WATER MULTIPLE
```

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

PDMROF method (ABST_PDMROF)

This approach, which has been shown to be successful in prairie and wetland systems, is documented in (Mekonnen et al., 2014), and assumes a probability distribution of storage capacity within the HRU represented with the Pareto distribution parameter b (land use parameter PDMROF_B). A maximum local depression storage capacity, c_{max} is calculated as

$$c_{max} = \phi_{max} \cdot (b + 1)$$

where ϕ_{max} is the maximum total average depression storage in the HRU, land use parameter DEP_MAX [mm]. From this, the current critical capacity c^* may be calculated from the depression

storage ϕ_{dep}:

$$c^* = c_{max}\left(1 - \left(1 - \frac{\phi_{dep}}{\phi_{max}}\right)^{\frac{1}{b+1}}\right)$$

The total amount of abstraction during a time step may be determined from ϕ_{pond}, the available ponded water ready to be abstracted:

$$M_{abst} = \frac{\phi_{max}}{\Delta t}\left[\left(1 - \frac{c^*}{c_max}\right)^{b+1} - \left(1 - \frac{c^* + \phi_{pond}}{c_{max}}\right)^{b+1}\right]$$

where the term $c^* + \phi_{pond}$ in the square brackets is constrained to be less than c_{max} (corresponding to the entire landscape shedding water). The amount of water not abstracted is moved to SURFACE_WATER storage as runoff.

UWFS method (ABST_UWFS)

This approach (the 'Upscaled Wetland Fill and Spill' approach) is another abstraction method shown to be successful in prairie regions and accounts for lateral flow effects in runoff generation from contributing area of multiple wetlands with characteristics described by a probability distribution. The heterogeneity of depressions properties i.e., deficit depth and concentrating factor are defined by exponential distribution function with the required land use parameters of minimum concentrating factor, UWFS_BETAMIN [-] and shape factor, UWFS_B (b).

The amount of abstraction during each time step is controlled by the difference between minimum available water to each depressions and minimum deficit as:

$$M_{abst} = P - \frac{1}{<\beta>}\begin{cases}\frac{ad}{a+d}(1-P_f)\left[\frac{dP^*-1+e^{-dP^*}}{d^2} + \frac{aP^*+1}{a^2}\right] + P_f\left(P^* + \frac{1}{a}\right) & \text{for } P^* > 0 \\ \frac{ad}{a+d}\left[\frac{e^{aP^*}}{a^2}\right] & \text{for } P^* < 0\end{cases}$$

where P is the available runoff (after infiltration of precipitation and snowmelt) on the landscape, P^* is calculated based on land use parameter UWFS_BETAMIN (β_{min}) and precipitation and state variable MIN_DEP_DEFICIT [mm], P_f is the percentage of full depressions which is calculated by the model, a is b/P, $<\beta> = (\beta_{min}+1)/b$, and d is deficit distribution shape factor which is calculated from land use parameters of DEP_MAX [mm], MAX_DEP_AREA_FRAC [0..1] and the current amount of depression storage [mm].

The minimum deficit in depressions is updated at each time step based on the generated runoff. If runoff is generated from the basin, the new minimum deficit depth (D_{min}) is zero. However, if there is no runoff generated form the basin the new minimum deficit depth will be updated as:

$$D_{min}^{new} = D_{min} - \min(\beta_{ave}P, D_{min})$$

where $\beta_{ave} = \beta_{min} + 1/b$ is the average concentrating factor in the HRU.

3.12 Depression/Wetland Storage Overflow

Depression overflow refers to water lost from ponds and wetlands to the main surface water network. Depression overflow moves water from the DEPRESSION storage variable and is always moved to SURFACE_WATER. Depression overflow is rate-limited by the availability of water in depression storage. Usage in .rvi file:

```
:DepressionOverflow DFLOW_THRESHPOW DEPRESSION SURFACE_WATER
```

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:

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

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

Linear depression overflow(DFLOW_LINEAR)

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

$$M_{dflow} = k_d \cdot (\phi_{dep} - \phi_{th})$$

where ϕ_{dep} is the current depression storage [mm], ϕ_{th} is the given threshold storage level [mm] (land use parameter DEP_THRESHOLD), and k_d is the linear storage coefficient [1/d] (land use parameter DEP_K). If $\phi_{dep} < \phi_{th}$, $M_{dflow} = 0$.

Weir-like depression overflow(DFLOW_WEIR)

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

$$M_{dflow} = 0.666 * c * \sqrt{2g} * pow(max(\phi_{dep} - \phi_{th}, 0.0), 1.5);$$

where ϕ_{dep} is the current depression storage [mm], ϕ_{th} is the given threshold storage level [mm] (land use parameter DEP_THRESHOLD), and , and g is gravity. Here the crestwidth c is estimated as:

$$c = r_d \frac{\sqrt{A}}{A}$$

where r_d is the ratio of the HRU area A to the length of the HRU boundary over which seepage occurs, estimated as $\sqrt{A}$ (land use parameter DEP_CRESTRATIO)

3.13 Seepage from Depressions/Wetlands

Seepage overflow refers to water lost from ponds and wetlands to lower soil units, including groundwater.

Seepage moves water from the DEPRESSION storage variable and is always moved to one of the SOIL units, subject to the availability of water in depression storage and remaining room in the soil. Seepage is rate-limited by the availability of water in depression storage. Example usage in the .rvi file:

```
:Seepage SEEP_LINEAR DEPRESSION SOIL[1]
```

Available Algorithms

Linear seepage (**SEEP_LINEAR**)

The seepage to surface water is controlled by the amount of water in depression storage:

$$M_{dflow} = k_{seep} \cdot \phi_{dep}$$

where ϕ_{dep} is the current depression storage [mm], and k_{seep} is the linear seepage coefficient [1/d] (land use parameter DEP_SEEP_K).

3.14 Lake Release

Lake release refers to delayed release of water from lake storage, and is only used when a specific LAKE_STORAGE storage compartment is specified using the :LakeStorage command (alternately, all water falling on LAKE-type HRUs will be directed directly to the surface water network after accounting for open water evaporation, which may lead to a flashier-than-expected hydrograph). Lake storage is intended to represent lakes which are connected to the surface water network either directly or indirectly via groundwater. Exchange can be bidirectional such that low lake levels may extract water from surface water storage. Lake release is disabled for LAKE-type HRUs when they are linked to surface water reservoirs; for these (usually larger) lakes, the delayed release from storage is completely controlled by the reservoir/lake outflow structure.

Lake release typically moves water from the LAKE_STORAGE storage variable to SURFACE_WATER; Lake storage may go below zero, which corresponds to a disequilibrium with the surface water network such that lakes will extract water. Lake release is not rate-limited except for when excess negative lake storage will dry out the surface water. Example usage in the .rvi file:

```
:LakeRelease LAKEREL_LINEAR LAKE_STORAGE SURFACE_WATER
```

Available Algorithms

Linear release (LAKEREL_LINEAR)

The rate of release to/from surface water is controlled by the amount of water in lake storage:

$$M_{lrel} = k_{lrel} \cdot \phi_{lake}$$

where ϕ_{lake} is the current (positive or negative) net lake storage [mm], and k_{lrel} is the linear storage coefficient [1/d] (land use parameter LAKE_REL_COEFF). Note that lake seepage can be negative (increasing lake storage) if the net lake storage is negative.

3.15 Open Water Evaporation

Open water evaporation moves water from DEPRESSION, SURFACE_WATER, PONDED_WATER, or LAKE_STORAGE to the atmosphere. This calculation is independent of the evaporation from reservoirs, which is calculated as equal to the open water PET determined by the :OW_Evaporation algorithm. All algorithms below use the PET over open water, PET_{ow}, corrected by the land use parameter OW_PET_CORR, also using the :OW_Evaporation algorithm. All of these algorithms update the total actual evapotranspiration (AET) from the HRU, and competitive ET is supported such that this algorithm competes with soil and canopy evaporation - the PET rate is reduced by the AET after application.

Example Usage in the .rvi file is:

```
:OpenWaterEvaporation OPEN_WATER_EVAP DEPRESSION ATMOSPHERE
```

The recipient 'to' compartment is always ATMOSPHERE, but the user specifies the source ('from') compartment.

Available Algorithms

Basic (OPEN_WATER_EVAP) (default)

The rate of loss is equal to the PET_{ow}, modified by the correction factor:

$$M_{evap} = C \cdot \text{PET}_{ow}$$

where C is the land use parameter OW_PET_CORR.

Riparian (OPEN_WATER_RIPARIAN)

As used in the SAC-SMA model emulation. The rate of loss is equal to the PET_{ow}, modified by the correction factor, and the percentage of land cover covered in streams/riparian landcover:

$$M_{evap} = C \cdot f_s \cdot \text{PET}_{ow}$$

where C is the land use parameter OW_PET_CORR and f_s is the land use parameter STREAM_FRACTION [0..1].

Upscaled wetland fill and spill algorithm (OPEN_WATER_UWFS)

The UWFS method, a algorithm for simulating prairie pothole regions and other wetland-dominated landscapes, calculates the open water evaporation rate same as in the OPEN_WATER_EVAP method, but adjusted byt the land use parameter MAX_DEP_AREA_FRAC [0..1] which represents the percentage of landscape covered by depressions.

$$M_{evap} = C \cdot f_d \cdot \text{PET}_{ow}$$

where C is the land use parameter OW_PET_CORR and f_d is the land use parameter MAX_DEP_AREA_FRAC [0..1].

This algorithm should typically be used in UWFS model in conjunction with ABST_UWFS abstraction method. This method therefore also calculates changes in the minimum deficit depth of wetlands, MIN_DEP_DEFICIT. The minimum deficit depth after evaporation is updated as:

$$D_{min}^{new} = D_{min}^{old} + M_{evap} \cdot \Delta t$$

3.16 Process Group

Process groups are used to configure so-called blended models, in which the flux resulting from a given hydrologic process is determined by a weighted average of two or more process options.

This option requires that all process options within the blended group use the same 'to' and 'from' compartments, but can otherwise be applied to any type of hydrologic process. This approach was introduced in the work by Mai et al. (2020). In Raven implementation, the process group is treated as a hydrologic process, and can be used in combination with conditional or other commands in the .rvi file.

The weights corresponding to the N hydrologic processes within the process group can be specified in one of two ways. The weights can either be specified directly for the process group, or with $N-1$ weight-generating parameters that are independent uniform numbers, and are used to calculate N weights that sum to 1.0. This latter method of determining N weights from $N-1$ uniform numbers is referred to as the 'pie-sharing' approach, and is described in Mai et al. (2022).

Example configuration of a blended infiltration process group with 3 process options,and specifying weights explicitly in the .rvi file:

```
:ProcessGroup
  :Infiltration  INF_HMETS     PONDED_WATER  MULTIPLE
  :Infiltration  INF_VIC_ARNO  PONDED_WATER  MULTIPLE
  :Infiltration  INF_HBV       PONDED_WATER  MULTIPLE
:EndProcessGroup 0.333 0.333 0.334
```

or, equivalently using only 2 weight-generating parameters and the `CALCULATE_WTS` command to calculate a set of 3 weights:

```
:ProcessGroup
  :Infiltration  INF_HMETS     PONDED_WATER  MULTIPLE
  :Infiltration  INF_VIC_ARNO  PONDED_WATER  MULTIPLE
  :Infiltration  INF_HBV       PONDED_WATER  MULTIPLE
:EndProcessGroup CALCULATE_WTS 0.55556 0.5
```

3.17 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.

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. Snow balance is rate-limited by the storage in 'from' and 'to' compartments. Example usage in the .rvi file (note that most snow balance models are manipulating multiple storage compartments):

```
:SnowBalance SNOBAL_SIMPLE_MELT SNOW PONDED_WATER
```

or

```
:SnowBalance SNOBAL_TWO_LAYER MULTIPLE MULTIPLE
```

Most of the snowmelt algorithms that explicitly simulate liquid water content within the snowpack use the global parameter SNOW_SWI to determine the maximum possible liquid water storage of the snowpack:

$$\phi_{max}^{sl} = SWE \cdot SWI$$

where ϕ_{max}^{sl} [mm] is the maximum liquid water storage of the snowpack, SWE is the snow water equivalent of the snowpack [mm], and SWI is the global parameter SNOW_SWI, which defaults to 0.05 if not specified.

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.

$$M_{melt} = M'_{melt}$$

where the potential melt rate, M'_{melt} [mm/d], is calculated using one of the methods described in section 5.8.1. The melt rate is constrained such that only available snow will melt (i.e., the maximum melt rate is $SWE/\Delta t$) . This is the same as using :SnowMelt MELT_POTENTIAL.

HBV snow balance (SNOBAL_HBV)

The HBV snow balance (Bergstrom, 1995; Lindström et al., 1997) represents both melt and liquid water storage in the pore space of the snow. The melt rate is determined by the potential melt rate algorithm (POTMELT_HBV for true HBV emulation), while refreeze is calculated using:

$$M_{refreeze} = K_a \cdot \max(T_f - T, 0)$$

where K_a is the land use parameter REFREEZE_FACTOR [mm/d/ °C], and T_f is the melt temperature [°C] (zero by default, but can be set with the land use parameter DD_MELT_TEMP).

Meltwater fills the snow pore space first (with the maximum fillable pore space determined by the global parameter SNOW_SWI), then is allowed to overflow. All overflow percolates into SOIL[0] by default, but may be redirected to PONDED_WATER using the :Redirect command if desired.

UBCWM snow balance (SNOBAL_UBCWM)

As described in the UBC Watershed model documentation (Quick, 1995). Potential melt is typically calculated using the POTMELT_UBCWM method described in section 5.8.1. 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 Neige snow balance (SNOBAL_CEMA_NEIGE)

Often used with the GR4J model configuration, the Cema Neige snow balance uses the potential melt rate calculated using the methods of section 5.8.1, but corrected with a snow cover factor,

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

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 AVG_ANNUAL_SNOW.

Two-layer snow balance (SNOBAL_TWO_LAYER)

A two-layer snowmelt model that simulates accumulation of cold content, changes in surface snow temperature, and evolution both liquid and solid snow stores. Available energy (supplied as potential melt) is first used to bring the temperature of the surface snowpack to freezing, then the remainder is used to melt the frozen snow, which is allocated to liquid snow until the pack is ripe, at which point it then drains into PONDED_WATER storage. Ripeness is controlled by the global parameter SNOW_SWI, which represents the maximum liquid snow storage capacity as a fraction of snowpack SWE. The second (bottom) layer is only applied when the snow as SWE exceeds the global parameter MAX_SWE_SURFACE, in mm.

HMETS snow balance (SNOBAL_HMETS)

A snowmelt model documented in Martel et al. (2017). This is a simple single layer snowmelt model with degree day freezing, which tracks liquid water content in the snowpack in addition to SWE. The refreeze rate (constrained by water availability) is given by:

$$M_{rf} = K_f \cdot (T_{rf} - T_{di})^f$$

where K_f is the land use property REFREEZE_FACTOR, T_rf is the degree day refreeze factor (land use property DD_REFREEZE_TEMP, T_{di} is the average of the average daily temperature and the minimum daily temperature, and f is the land use parameter REFREEZE_EXP. The water retention capacity (upper limit of liquid water content in snow) varies over the course of the year based upon cumulative snowmelt:

$$\text{SWI} = \max\left(\text{SWI}_{min}, \text{SWI}_{max} \cdot (1 - \alpha \cdot M_{cumul})\right)$$

where SWI_{min} and SWI_{max} are the land use parameters SNOW_SWI_MIN and SNOW_SWI_MAX, α is the land use parameter SWI_REDUCT_COEFF, and M_{cumul} is the cumulative melt since the last period of zero snow depth (a state variable tracked by this algorithm).

CRHM EBSM snow balance (`SNOBAL_CRHM_EBSM`)

A snowmelt model based upon that of Marks and Dozier (1992) as implemented within the CRHM hydrological model Pomeroy et al. (2007). The single-layer energy-balance-based snow model tracks liquid water content, SWE, and energy content of the snowpack.

3.18 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 content is low.

Sublimation always occurs between SNOW and ATMOSPHERE units and is limited by the availability of snow. When sublimation is negative, it behaves as deposition, enabling water vapor to solidify on the snowpack.

Example usage in the .rvi file:

```
:Sublimation SUBLIM_KUZMIN SNOW ATMOSPHERE
```

Available Algorithms

Kuzmin (1972) method (SUBLIM_KUZMIN)

The sublimation rate (in [mm/d]) is calculated using the following empirical relationship (Kuzmin, 1972; Kutchment and Gelfan, 1996):

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

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, where the vapour pressure is calculated from the relative humidity and saturated vapour pressure is estimated from average daily air temperature.

Kuchment-Gelfan method (SUBLIM_KUCHMENT_GELFAN)

Identical to the SUBLIM_KUZMIN method, but uses time step mean temperature rather than daily mean temperature to estimate vapour pressures. Preferred for sub-daily models.

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):

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

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]. Wind velocity and vapour pressures are estimated by Raven from the specified forcing algorithms for wind speed, temperature, and relative humidity.

Bulk Aerodynamic method (SUBLIM_BULK_AERO)

The latent heat flux is calculated using the following relationship Hock and Holmgren (2005):

$$Q_e = \rho_a \lambda_s \cdot C_E \cdot v_{ave} * \frac{0.622(P_{sat}^{snow} - P_{ave})}{P_a}$$

where v_{ave} [m/s] is the wind velocity at reference height, P_a, P_{sat}^{snow} and P_{ave} [kPa] are the air pressure, saturated vapour pressure at the snow surface, and vapour pressure, respectively, λ_s is

the latent heat of sublimation, and ρ_a is air density. The transfer coefficient, C_E, is calculated as:

$$C_E = \frac{\kappa^2}{\ln(z_{ref}/z_0)\ln(z_{ref}/z_0^e)}$$

where z_{ref} is the reference height of 2m, z_0 is the aerodynamic roughness length of snow, z_0^e is the aerodynamic roughness length for temperature, and κ is the von Karman constant (=0.42). Lastly, sublimation rate is calculated as:

$$M_{subl} = \frac{Q_E}{\lambda_s \rho_w}$$

where ρ_w is the density of water [kg/m^3]. This energy-budget-based methodology does not have any free parameters.

3.19 Snow Refreeze

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

Snow refreeze always occurs between `SNOW_LIQ` and `SNOW` units. Snow refreeze is limited by the availability of liquid water in the snowpack. Refreeze rates must be positive. In most cases, snow refreeze should be handled using the `:SnowBalance` routines. Example usage in the .rvi file:

```
:SnowRefreeze FREEZE_DEGREE_DAY SNOW_LIQ SNOW
```

Available Algorithms

Degree day method (`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):

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

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.

3.20 Snow Albedo Evolution

Snow albedo evolution is the process through which snow albedo changes due to snow compaction, snowpack aging, or fresh snow accumulation. The snow albedo evolution algorithms have no sources or sinks, it simply models the rate of change of albedo over time. Snow albedo is constrained to be in the range 0-1. Example usage in the .rvi file (note that there is no 'to' and 'from' state variable, since this is not changing the water/energy balance):

```
:SnowAlbedoEvolve SNOALB_UBC
```

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 and MAX_SNOW_ALBEDO, defined in the :UBCSnowParams command in the .rvp file.

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

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

where α_{max} is the global parameter MAX_SNOW_ALBEDO, α_b is a threshold albedo value (UBC_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 UBC_ALBSNW), K is the global parameter UBC_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 (UBC_MAX_CUM_MELT). All of these global parameters are specified using the command :UBCSnowParams in the .rvp file.

CRHM Essery Approach (SNOALB_CRHM_ESSERY)

A simple albedo evolution algorithm developed by Richard Essery, now at the University of Edinburgh. The albedo decays as follows:

$$M_{snalb} = -\beta \qquad \text{if } T_{snow} < 0$$

$$M_{snalb} = -\beta_2 \cdot (\alpha - \alpha_{min}) \qquad \text{if } \alpha < \alpha_b$$

where β is he global parameter ALB_DECAY_COLD [1/d], β_2 is the global parameter ALB_DECAY_MELT [1/d], and α_{min} is the global parameter MIN_SNOW_ALBEDO [-]. The albedo also increases due to fresh snow using the following

$$M_{snalb} = (\alpha_{max} - \alpha) \cdot \min(S/S_{thresh}, 1.0) \cdot \Delta t$$

where α_{max} is the global parameter MAX_SNOW_ALBEDO [-], S is the snowfall rate [mm/d], and S_{thresh} is the global parameter SNOWFALL_ALBTHRESH [mm/d].

Baker Approach (SNOALB_BAKER)

A simple albedo evolution algorithm from Baker et al. (1990), as ported from CRHM (Pomeroy et al., 2007). The albedo decays as follows:

$$\alpha = 0.9 - 0.0473 \cdot A_{snow}^{0.1}$$

where A_{snow} is the age of the snow in days. The snow age reboots to zero if the snowfall rate exceeds the global parameter SNOWFALL_ALBTHRESH [mm/d].

3.21 Glacier Melt

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

Glacier 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. Glacial melt is not limited by the available glacier ice, which is assumed to be abundant. Example usage in the .rvi file:

```
:GlacierMelt GMELT_SIMPLE_MELT GLACIER_ICE SURFACE_WATER
```

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 5.8.1.

HBV approach (GMELT_HBV)

The melt rate is equal to the potential melt rate, calculated using the methods described in section 5.8.1. 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).

3.22 Glacier Release

Glacier release refers to the release of meltwater stored within a glacier to surface water, and is typically used in conjunction with the glacier melt process, i.e., melt is released from the surface and is temporarily stored or delayed before reaching the surface water network.

Glacier release algorithms move water from GLACIER to SURFACE_WATER. Glacial release is limited by the available glacier liquid water storage. Example usage in the .rvi file:

```
:GlacierRelease GRELEASE_LINEAR_STORAGE GLACIER SURFACE_WATER
```

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 (Hamilton et al., 2000):

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

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.

3.23 Glacier Infiltration

Glacier infiltration release refers to the movement of meltwater to other storage compartments, either within the glacier or in underlying soil layers. This is typically used in conjunction with the glacier melt process, i.e. melt is released from the surface and then directed via infiltration into one of several plausible storage units.

Glacier infiltration algorithms move water from `PONDED_WATER` to `MULTIPLE`, including `GLACIER` and/or soil stores. Example usage in the .rvi file:

```
:GlacierInfiltration GINFIL_UBCWM PONDED_WATER MULTIPLE
```

Available Algorithms

UBC Watershed Model approach (`GINFIL_UBCWM`)

The runoff is determined based on the capacity of underlying soils to accept rain and melt water, and moves to `GLACIER` storage. Of the remaining ponded water (i.e. for which soils have capacity), a proportion equal to 1-`UBCGroundwaterSplit` moves to the upper groundwater store and a proportion equal to `UBCGroundwaterSplit` moves to the lower groundwater store.

3.24 Lake Freezing

Lake freezing tracks the thickness of a frozen ice surface on simulated lakes and allows snow to accumulate atop the lake surface. When ice is not present, any snowfall on a lake is immediately melted and added to the lake water budget. This process is only applied to HRUs with a LAKE-type soil profile.

Lake freezing algorithms do not move water between compartments, but rather adjust the `ICE_THICKNESS` state variable; this influences the `SNOW`, `SNOW_LIQ`, and `COLD_CONTENT` state variables in the partitioning of precipitation. Example usage in the .rvi file:

```
:LakeFreeze LFREEZE_BASIC
```

Available Algorithms

Basic approach (`LFREEZE_BASIC`)

The ice thickness is determined from the potential melt rate, with a buffering effect applied due to the presence of snow.

$$\frac{dT}{dt} = -\left(1 - \min\left(\frac{\phi_{SWE}}{B}, 1\right)\right) M_{melt}$$

where M_{melt} [mm/d] is the potential melt rate (positive for melt, negative for freezing), ϕ_{SWE} [mm] is the snow water equivalent atop the ice, and B is the land surface parameter `LAKESNOW_BUFFER_HT` [mm].

3.25 Crop Heat Unit Evolution

Crop heat units (CHUs) are used by some organizations 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.

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. Crop heat units are zero outside of the growing season. Example usage in the .rvi file:

```
:CropHeatUnitEvolve CHU_ONTARIO
```

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 $^{\circ}\mathrm{C}$, at which time the crop heat units are set to zero. It ends when the temperature dips below -2 $^{\circ}\mathrm{C}$, or after September 30th. During the growing season, CHUs are incremented using the following expressions (Brown and Bootsma, 1993):

$$\mathrm{CHU}_d = 3.33 \cdot (T_{max} - 10) - 0.084 \cdot (T_{max} - 10)^2$$
$$\mathrm{CHU}_n = 1.8 \cdot (T_{min} - 4.4)$$
$$\mathrm{CHU}_{new} = \mathrm{CHU}_{old} + 0.5 \cdot (\mathrm{CHU}_d + \mathrm{CHU}_n)$$

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

3.26 Special Processes

The flush, lateral flush, split, redirect flow, 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 some or all of the water storage from one storage compartment to another.
- The Lateral Flush process instantaneously moves all of the water storage from one storage compartment in one or more HRUs in a basin to another storage unit in another HRU within the basin.
- The Lateral Equilibrate process exchanges storage in adjacent HRUs within a basin until their storages are equal in depth (and also mixes constituents)
- The Overflow process moves the excess water storage (more than the maximum capacity of the water storage unit) to another compartment. The .rvi command `:AllowSoilOverfill` can be used with the overflow process to ensure that soil compartments may be filled beyond their maximum capacity (which by default is not the case).
- The RedirectFlow process overrides the default storage targets of the previous algorithm
- 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.

The flush, lateral 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 by any other process to any storage compartment. Example usage in the .rvi file:

```
# moves all ponded water to surface water
:Flush RAVEN_DEFAULT PONDED_WATER SURFACE_WATER

# moves 70\% of ponded water to surface water
:Flush RAVEN_DEFAULT PONDED_WATER SURFACE_WATER 0.7

# moves liquid snow in excess of maximum liquid snow storage
# to surface water
:SnowBalance SNOBAL_SIMPLE_MELT SNOW SNOW_LIQ
   :-->Overflow RAVEN_DEFAULT SNOW_LIQ SURFACE_WATER

# redirects runoff from infiltration algorithm to a conceptual store
:Infiltration INF_GREEN_AMPT PONDED_WATER MULTIPLE
   :-->RedirectFlow SURFACE_WATER SOIL[4]

# moves 60% of ponded water to surface water, the rest infiltrates
:Split RAVEN_DEFAULT PONDED_WATER SURFACE_WATER SOIL[0] 0.6
```

```
# delays release of surface water to outlet through convolution
:Flush RAVEN_DEFAULT      SURFACE_WATER   CONVOLUTION[0]
:Convolve CONVOL_GR4J_1   CONVOLUTION[0]  SURFACE_WATER

# moves all runoff from upland HRUs to wetlands
# requires definition of Uplands and Wetlands HRU groups
:LateralFlush RAVEN_DEFAULT Uplands SURFACE_WATER To Wetlands DEPRESSION

# instantaneously equilibrates groundwater storage in basin HRUs
# (equivalent to deprecated :AggregatedVariable SOIL[3] AllHRUs
:LateralEquilibrate RAVEN_DEFAULT AllHRUs SOIL[3] 1.0
```

Available Algorithms (Convolution)

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.

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 2 (`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`.

Gamma transfer function 1 (`CONVOL_GAMMA`)

The transfer function used is

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

where a and β are the land use parameters `GAMMA_SHAPE` and `GAMMA_SCALE`, respectively.

Gamma transfer function 2 (CONVOL_GAMMA2)

The transfer function used is

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

where a and β are the land use parameters GAMMA_SHAPE2 and GAMMA_SCALE2, respectively. The purpose of this is to be able to support two convolution processes in serial or parallel.

Chapter 4

Routing and Water Management

The following chapter outlines the routing algorithms available for modelling the downstream migration of water through a terrain/channel/reservoir 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 subbasin river reach, then again a delay before the water reaches the subbasin outlet. This total 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 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.

While this chapter addresses the primary catchment-channel-reservoir routing progression in Raven, Raven supports alternate means of influencing the timing characteristics of a basin. For instance, some lateral routing between HRUs may be performed prior to delivery to the stream network (e.g., see section 3.26 for discussion of the Lateral Flush process which can be used to route water laterally). This lateral transfer is separate from the landscape routing described in this chapter, but may therefore impact propagation of water downstream, for instance by sending landscape runoff to a riparian wetland. Other conceptual models (e.g., those in HBV) route water through conceptual routing stores; this is supported in Raven by using 'artificial' soil horizons as routing stores, as can be seen in the HBV-EC and GR4J model evaluations in appendix F.

4.1 In-Catchment Routing

4.1.1 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.,

$$Q(t+\Delta t) = \sum_{n=0}^{N} Q_{lat}(t - n\Delta t) \cdot UH_n \tag{4.1}$$

where $Q(t)$ [m^3/s] is the flow rate into the channel from the subbasin at time t, $Q_{lat}(t)$ [m^3/s] is the constant lateral release flow rate from the HRU surface over the time step 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 of 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:

$$Q(t) = \int_0^{\infty} Q_{lat}(t - \tau) \cdot UH(\tau) d\tau \tag{4.2}$$

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. There are numerous approaches in the literature for estimating unit hydrograph characteristics. While the unit hydrographs below are reported as continuous, they are internally converted to a discrete version via the following relation:

$$UH_n = \frac{1}{\Delta t} \int_{n\Delta t}^{(n+1)\Delta t} UH(t) dt \quad n = 0 ... \infty \tag{4.3}$$

Only non-zero $\vec{UH}$ vector elements are retained.

4.1.2 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. Note that all of the below algorithms describe the continuous form of the unit hydrograph for an event of one time step duration.

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 (i.e., as soon as it can given a finite 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.,

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

where $\Gamma(a)$ is the Gamma function and a and β are the subbasin parameters `GAMMA_SCALE` and `GAMMA_SHAPE`. If not provided, a defaults to a reasonable value of 3. if the `GAMMA_SCALE` parameter is not provided but the time to peak subbasin parameter `TIME_TO_PEAK` (t_p) is provided, then β is calculated as follows:

$$\beta = \frac{a - 1}{t_p}$$

Note that this automatic calculation can lead to issues when a<1.0, because the peak value occurs at $t = 0$ for this distribution when $a < 1.0$.

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. The property `TIME_TO_PEAK` must be less than `TIME_CONC`.

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

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 integer number of reservoirs (N) equal to `NUM_RESERVOIRS` (another subbasin property).

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

4.2 In-Channel Routing

4.2.1 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 (or themselves embody an entire 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

$$Q_{out}^{n+1} = F_{route}(Q_{out}^{n}, \vec{Q}^{in}, \vec{P}_s) + Q_{lat}^{n+1} \tag{4.4}$$

where F_{route} is the routing algorithm, $\vec{Q}^{in}$ is the recent time history of upstream (and upbasin) inflows to the channel, Q_{lat}^{n+1} is the lateral inputs generated by the in-catchment routing method for the time step, $\vec{P}_s$ is a vector of channel parameters, typically a number of channel rating curves, primary channel and bank roughness, and weir or reservoir relationships. Lastly, Q_{out}^{n+1} is the outflow from the stream reach at the end of the time step. 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.

4.2.2 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 nearly 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:

$$Q_{out}^{n+1} = \sum_{i=0}^{N} Q_{in}^{n-i+1} \cdot UH_i' + Q_{lat}^{n+1} \tag{4.5}$$

where Q_{out}^{n+1} [m^3/s] is the flow rate from the subbasin at the end of the time step, Q_{in}^{n} [m^3/s] 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:

$$c_{ref} = \left.\frac{dQ}{dA}\right|_{Q_{ref}} \tag{4.6}$$

c_{ref} is the reference celerity for the reach, the velocity corresponding to the reference flow, Q_{ref} [m^3/s] in the reach, usually specified as the bank full flow using the subbasin parameter `Q_REFERENCE`. If `Q_REFERENCE` is not provided explicitly, it is estimated from the `AVG_ANNUAL_RUNOFF` parameter. 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_{ref} 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 4.6.

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 4.6, 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. The diffusive wave model is currently the preferred routing method for transport simulation. If `Q_REFERENCE` is not provided explicitly, it is estimated from the `AVG_ANNUAL_RUNOFF` parameter.

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 Williams (1969):

$$Q_{out}^{n+1} = c_1 \cdot Q_{in}^{n+1} + c_2 \cdot Q_{in}^{n} + c_3 \cdot Q_{out}^{n} + Q_{lat}^{n+1} \tag{4.7}$$

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

$$k = \min\left(\frac{1}{\frac{K}{\Delta t} + 0.5}, 1\right) \tag{4.8}$$

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:

$$Q_{out}^{n+1} = c_1 \cdot Q_{in}^{n+1} + c_2 \cdot Q_{in}^{n} + c_3 \cdot Q_{out}^{n} + Q_{lat}^{n+1} \tag{4.9}$$

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

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

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

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

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

where c_{ref} is the reference celerity for the reach (calculated using equation 4.6), 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 for the routing algorithm. 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. If `Q_REFERENCE` is not provided explicitly, it is estimated from the `AVG_ANNUAL_RUNOFF` parameter.

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} (without the in-channel routing included). The final outflow is calculated by adding Q_{lat}^{n+1} to the resultant outflow from this algorithm. 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.

4.3 Lake and Reservoir Routing

4.3.1 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. Raven supports a range of methods for determining the outflow from a reservoir or lake using either stage-discharge relationships or operational constraints such as flow and stage targets. Each reservoir may have two stage-discharge curves to represent, for example, combined tunnel underflow and spillway overflow. For simple natural lakes, stage-discharge curves can be calculated by Raven, only the estimated crest width of the lake overflow is specified by the user. A schematic of two common reservoir configurations, one for a prismatic single-parameter lake and one for a general managed reservoir are shown in figure 4.1

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:

$$\begin{aligned} \frac{dV(h)}{dt} &= Q_{in} - Q(h) + P \cdot A^{*} - E \cdot A(h) - S(h) \qquad (4.10) \\ \frac{V(h^{n+1}) - V(h^{n})}{\Delta t} &= \frac{1}{2}(Q_{in}^{n} + Q_{in}^{n+1}) - \frac{1}{2}\left(Q(h^{n}) + Q(h^{n+1})\right) \\ &\quad + P \cdot A^{*} - \frac{E}{2}\left(A(h^{n}) + A(h^{n+1})\right) - \frac{1}{2}\left(S(h^{n}) + S(h^{n+1})\right) \end{aligned}$$

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 (appendix A.3.2). P is the direct precipitation, applied to the fixed area A^{*} of the HRU associated with the reservoir, E is the timestep-averaged open-water evaporation rate for the reservoir calculated as determined by the `:OWEvaporation` command for the reservoir-linked open water HRU. $S(h) = k \cdot (h - h_{gw})$ is the groundwater seepage rate of the reservoir, calculated from a seepage coefficient k [m^3/d] and groundwater reference head h_{gw} (specified using the `:SeepageParameters` command). Reservoir evaporation may be modified by the `LAKE_PET_CORR` land surface parameter. 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

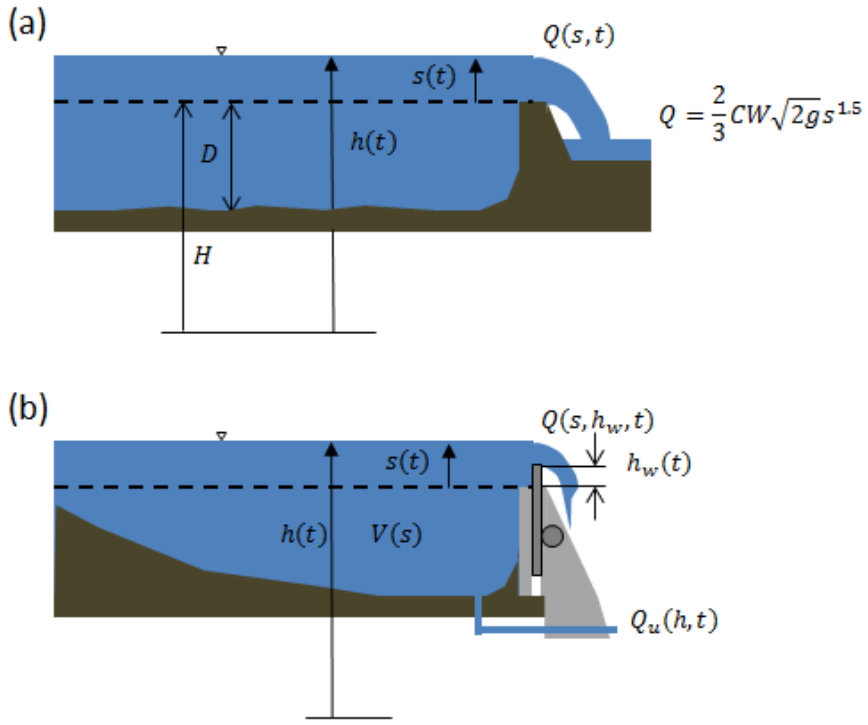

Figure 4.1: Example reservoir configurations in RAVEN. (a) a lake-type reservoir and (b) a general managed reservoir. $h(t)$ is the absolute stage height and $s(t)$ is the height of the water level above the minimum crest height. For the lake-type reservoir, D is the `:MaxDepth` parameter, H is the optional `:AbsoluteCrestHeight` parameter, C is the `:WeirCoefficient`, and W is the `:CrestWidth`. For operated reservoirs, typically the stage-volume ($V(h)$) and stage-discharge ($Q(h)$) curves are provided, with optional support for underflow $Q_u(h)$ if desired. The gate shift $h_w(t)$ may be specified to represent the operation of a stop-log weir or similar.

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 stage-discharge ($Q(h)$) and stage-volume ($V(h)$) curves. The outflow from the reservoir is determined solely from the stage-discharge curve unless overridden by operational rules.

Operational controls that can be applied to determine reservoir outflow include:

- Maximum Stage constraints - the maximum stage may be provided as a time series using the `:ReservoirMaxStage` command in an .rvt file. The maximum stage constraint overrides all other controls.
- Minimum Stage constraints - the minimum stage may be provide as a time series using the `:ReservoirMinStage` command in an .rvt file. The minimum stage must be less than the maximum stage; if the minimum stage constraint is hit, then the outflow is set to zero or to the minimum flow as proscribed in the `:ReservoirMinStageFlow` time series command.
- Target Stage constraints - the target stage may be provides as a time series using the `:ReservoirTargetStage` command in an .rvt file. The target stage must be between the minimum and maximum stage. When supplied, the required outflow at the end of the time step Q^{n+1} needed to maintain the target stage (i.e., such that $h^{n+1} = h^{n+1}_{target}$) will be determined. If there are no

maximum flow change constraints, this will be applied and the target stage will always be met. However, if a maximum flow change constraint is met, then the change in discharge over the time step ($Q^{n+1} - Q^n$) will be limited by the flow constraint, which is expressed as the maximum positive rate of change in outflow, in m^3/s. This maximum flow change constraint is supplied as a time series via the `:ReservoirMaxQDelta` command. There can also be a maximum negative rate of change in outflow specified using the `:ReservoirMaxQDecrease` command.

- Variable weir height - the datum of the stage-discharge curve may be shifted over time by specifying the relative weir height as a time series with the `:VariableWeirHeight` command. This can emulate the operation of a stop-log weir or similar weir structure where the crest height is controlled by operators.
- Outflow override - the outflow from the reservoir may be completely specified by the user if the `:OverrideReservoirFlow` time series command is supplied for the simulated reservoir.
- Minimum flow constraints - the minimum flow may be provided as a time series using the `:ReservoirMinFlow` command in an .rvt file. This specified minimum flow may increase to satisfy downstream target flows specified using the `:ReservoirDownstreamFlow` command, if present.
- Maximum flow constraints - the maximum flow may be provided as a time series using the `:ReservoirMaxFlow` command in an .rvt file.
- The DZTR (Dynamic Zoned Target Release) model of Yassin et al. (2019) - the discharge is calculated using a time-variable volume-discharge curve generated via matching historical observations occurring under unknown reservoir operational rules using the `:DZTRReservoirModel` command in the .rvh `:Reservoir` block.

Advice

If any of these reservoir constraints are constant in time or annually cyclical, use the `:AnnualCycle` time series command so you don't have to specify a long redundant time series in the .rvt file.

The combination of maximum, minimum, and target stage constraints may be used, for instance, to emulate historical application of rule curves during the model calibration/validation process. The override reservoir outflow control can be used to replace simulated outflows from a reservoir with observed outflows during model calibration/validation or it can be used in short-term forecasting to examine the influence of operational decisions on reservoir stage and downstream flows. Lastly, approximate rule curves may be used in forecasting for systems where actual operational rules are unknown. It is worth noting the order of priority of these different constraints when multiple operational controls are specified for a reservoir (from highest priority to lowest priority):

1. Maximum stage constraint has highest priority (`:ReservoirMaxStage`)
2. Minimum flow and maximum flow constraints (`:ReservoirMinFlow` and `:ReservoirMaxFlow` (and also downstream target flows from `:ReservoirDownstreamDemand` influencing minimum flow))
3. Overridden flow (`:OverrideReservoirFlow`)
4. Minimum stage with Min stage specified flow (which is zero if not specified) (`:ReservoirMinStage` and `:ReservoirMinStageFlow`)*
5. Qdelta constraints (`:ReservoirMaxQDelta` and `:ReservoirMaxQDecrease`)
6. Target stage (`:ReservoirTargetStage`)

7. Natural weir flows has lowest priority

*The minimum stage constraint can be moved to the top of this list (i.e., get highest priority) by using the `:MinStageConstraintDominant` command in the `:Reservoir` block.

Note that there are a number of different approaches available for discretizing and representing lakes and reservoirs in Raven. For small lakes, we may not wish to explicitly represent their outflow characteristics and only wish to represent their role in influencing the basin water balance. In this case, water HRUs would be included in the basin, but a reservoir would not be used. For smaller lakes or reservoirs that still have a notable influence on downstream flows, a single water HRU would be included in the subbasin and linked to the reservoir. For natural lakes, it is suggested to use the 'lake-type' reservoir input structure described in appendix A.3.2. For large reservoirs (especially those with multiple subbasins draining into them), it is suggested to treat the reservoir as its own single-HRU subbasin with zero reach length.

Known inflows or outflows from the reservoir (e.g., irrigation diversions) may be considered in the above mass balance using the `:ReservoirExtraction` time series command in the .rvt file.

If reservoirs are present in the model, the file `ReservoirStages.csv` file is automatically created (with the runname prefix if specified). A full reporting of reservoir mass balance for all gauged subbasins is provided in the file `ReservoirMassBalance.csv` if the `:WriteReservoirMBFile` command is included in the .rvi file.

Observations of reservoir inflow, reservoir net inflow, and/or reservoir stage may be supplied to Raven and be evaluated against simulated values using the full set of diagnostics indicated in section 8.2.

4.4 Water Demand and Flow Diversions

4.4.1 Overview

Raven supports user-specified time series of water demand (e.g., irrigation or water treatment withdrawals) and flow diversions from one part of the watershed to another. It also supports simplified management constraints upon reservoirs based upon downstream demand. The key tools for representing these management-driven influences on river discharge include the following commands, described in details in appendix A.4.4:

- `:WaterDemand` or `:IrrigationDemand` (.rvt): a time series of desired withdrawals from the outlet of a subbasin, constrained such that stream discharge must be greater than zero or the environmental minimum flow.
- `:EnvironmentalMinFlow` (.rvt): used to constrain irrigation demand to maintain environmental minimum flows. If flows are less than environmental minimum flows, irrigation demand goes unmet.
- `:FlowDiversion` (.rvt): moves water from the outlet of one subbasin to the inlet of another basin. A simple percentage of simulated discharge may be used, or a user-specified lookup table may be used with the `:FlowDiversionLookupTable` command.
- `:ReservoirDownstreamDemand` (.rvt): ensures that the minimum outflow from reservoirs satisfies irrigation demand downstream. This can be specified on a case-by-case basis where (e.g.) a single reservoir meets a percentage of demand at a single location. Alternately, multi-reservoir systems may support multiple downstream demands as is controlled by the `:ReservoirDemandAllocation` command in the .rvi file. Percentages of downstream demand may be determined by reservoir contributing area or by maximum reservoir capacity.

- :ReservoirExtraction (.rvt): supports both extraction from a reservoir or (if negative) injection of water into a given reservoir.
- :BasinInflowHydrograph (.rvt): can be used to add water to (or remove water from, if negative) a subbasin at its reach outlet. If negative, this command does not respect positivity constraints, so the use of :IrrigationDemand or :WaterDemand command is preferred for representing withdrawals.
- :BasinInflowHydrograph2 (.rvt): can be used to add water to a subbasin at its reach inlet.

Because extractions are applied at the inlet (upstream end of basin main reach) or outlet (downstream end of main reach), users may wish to consider discretizing the watershed into subbasins in such a way that demand or supply locations correspond to basin outlets. Unmet irrigation demand is reported in the output file demands.csv when the :WriteDemandFile command is included. The default approach in Raven is for environmental flow constraints to be applied locally, i.e., only the water demand in basin n is adjusted for environmental flow constraints in basin n - delivery of demanded water in upstream basins will not be adjusted to handle downstream flow constraints except explicitly via the :ReservoirDownstreamDemand command.

4.4.2 Reservoir Demand Allocation

Reservoirs are often managed in such a way to support downstream irrigation needs. Raven supports emulation of this management practice by modifying the minimum flow targets for individual reservoirs based upon the instantaneous flows at irrigation demand locations downstream. The :ReservoirDownstreamDemand command implements this functionality, and can support individual demands (e.g., Reservoir A is expected to meet 80% of irrigation demand at location 1) or multiple demands (e.g., the 3 reservoirs upstream of demand location 2 share the responsibility of satisfying the minimum flow at that location. Note that this local handling of downstream demand has been superseded by the global handling of the water management optimization functionality described in section 4.5.

The impact of multiple controls simultaneously applied are best demonstrated via example. Consider the watershed in figure 4.2.

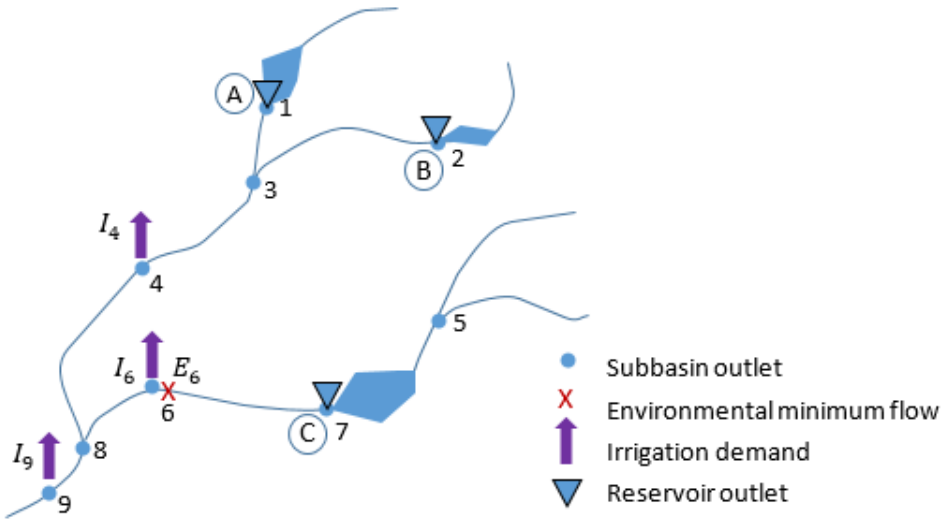

Figure 4.2: An example reservoir routing configuration. Numbers at subbasin outlets refer to the subbasin ID.

This system is represented using the following setup:

In the .rvi file:

```
:ReservoirDemandAllocation DEMANDBY_CONTRIB_AREA
```

In the .rvt file (note that the `:AnnualCycle` command is here used for brevity; daily or sub-daily time series can alternatively be provided):

```
:WaterDemand 4
  :AnnualCycle 0 0 0 0 20 20 20 20 20 10 0 0 #monthly flows [m3/s]
:EndWaterDemand
:WaterDemand 6
  :AnnualCycle 0 0 0 0 10 10 10 10 10 0 0 0
:EndWaterDemand
:WaterDemand 9
  :AnnualCycle 0 0 0 0 5 5 30 5 5 5 0 0
:EndWaterDemand
:EnvironmentalMinFlow 6
  :AnnualCycle 22 22 22 22 22 22 22 22 22 22 22 22
:EndEnvironmentalMinFlow

# 20% of irrigation demand in subbasin 4 is supported by
# reservoir A (in subbasin 1)
:ReservoirDownstreamDemand 4 1 0.2
# 90% of irrigation demand in subbasin 6 is supported by
# reservoir C (in subbasin 7)
:ReservoirDownstreamDemand 6 7 0.9
# 50% of irrigation demand in subbasin 9 is supported
# by upstream reservoirs
:ReservoirDownstreamDemand 9 _AUTO 0.5
```

In the .rvp file, the global demand multiplier, α_G is set with the command `:GlobalParameter RESERVOIR_DEMAND_MULT`. In the .rvh file, in the `:Reservoir` command blocks, the local demand multipliers α_i are set using the `:DemandMultiplier` property.

Because these reservoir demand constraints are additive, the minimum outflow from each reservoir can be calculated as follows:

$$
\begin{aligned}
Q^A_{min} &= \alpha_A \cdot \alpha_G \cdot \left((0.2 \cdot I_4) + \frac{A_1}{A_1 + A_2 + A_7} \cdot (0.5 \cdot I_9) \right) \\
Q^B_{min} &= \alpha_B \cdot \alpha_G \cdot \left(\frac{A_2}{A_1 + A_2 + A_7} \cdot (0.5 \cdot I_9) \right) \\
Q^C_{min} &= \alpha_C \cdot \alpha_G \cdot \left((0.9 \cdot I_6) + \frac{A_7}{A_1 + A_2 + A_7} \cdot (0.5 \cdot I_9) \right) + E_6
\end{aligned}
$$

where A_i, I_i and E_i are the contributing area, irrigation demand, and environmental minimum flow of basin i, respectively. Note that the α_i terms locally correct the influence of each dam; by setting to zero, the influence of this demand constraint can be turned off. The global modifier α_G can evaluate different management scenarios (e.g., determine the impact of not modifying reservoir flows to support downstream demand). It can be seen that if the local and global demand multipliers are equal to one, the

minimum flows of all three reservoirs will satisfy 50% of the demand in subbasin 9. It should also be noted that if an environmental minimum flow is specified at a downstream demand location, then it must be respected by the reservoir outflow, without any multiplier used.

4.5 Water Management Optimization

Raven (v4.0 and thereafter) supports simulation of basin water management as an optimization problem. With the `:ApplyManagementOptimization` option activated in the .rvi file, the management of reservoirs and the satisfaction of water demands is handled by optimally satisfying a set of management goals subject to user-specified management constraints in each time step. This command is typically supported by accompanied commands in the rvt file to specify water demands (`:WaterDemand`) and/or minimum environmental flows (`:EnvironmentalMinFlow` command), as well as the raven water management rvm file, which can be used to specify user-specified management goals and constraints. See Appendix A.6 for more on input file commands.

Note that water management can only support the linear routing modes `ROUTE_DIFFUSIVE_WAVE`, `ROUTE_PLUG_FLOW` and `ROUTE_NONE`.

With water management enabled, the following optimization problem is solved in each time step using the open-source linear programming library `lp_solve`:

$$\sum_{i=0}^{N_G} P_i(\vec{Q},\vec{h}) \rightarrow \min \tag{4.11}$$

where P_i is a set of penalties associated with not satisfying N_G management goals that are associated with water levels, $\vec{h}$, and discharges $\vec{Q}$, in the watershed. By default, any unmet water demand induces a penalty of the form.

$$P_i^{demand} = P_d \cdot (Q_d - D) \tag{4.12}$$

where P_d is the penalty coefficient associated with the demand (1.0 by default), Q_d is the water demand [m^3/s], and D is the delivered water [m^3/s].

Users can specify additional penalties associated with not satisfying other goals: for instance, a water level getting too low, a flow target not being met, or excessive ramping up of flow from a reservoir. Because management goals are often competing, we can weight the importance of each goal by changing the magnitude of penalties: important management goals will have larger penalties than less-important goals. This optimization is limited by a set of mass balance constraints and user-specified conditional management constraints, all of which *must* be satisfied. The variables adjusted during the optimization process are referred to here as decision variables, and include (by default) the outflow from each subbasin, the outflow from each lake/reservoir, and the stage in each lake/reservoir. These can be supplemented with additional user-defined decision variables such as cumulative withdrawal from a river segment or net hydropower production. All of the goals, constraints, and penalties are specified within the .rvm (management) input file.

The optimization process always enforces the following mass balance constraints in each time step. In each subbasin reach p, conservation of mass is imposed:

$$Q_p^{n+1} = U_0^p \sum_{i=0}^{N_p^{in}} Q_{p_i}^{n+1} + \sum_{j=1}^{N} U_j^p \sum_{i=0}^{N_p^{in}} Q_{p_i}^{n-j+1} + R_p^n - D_p^n + I_p^n \pm F_p^n \tag{4.13}$$

where Q_p^{n+1} is the subbasin outflow at the end of time step n, U_j^p are the transfer function terms for basin p, N_{in} is the number of inflowing basins to basin p and the term $Q_{p_i}^n$ is the inflow from basin i to

basin p at the end of time step n. The terms R_p^n, D_p^n, I_p^n, and F_p^n [m^3/s] represent the runoff to, delivered demand from, specified inflow to, and diversions to/from basin reach p. This is identical to the mass balance problem typically posed by RAVEN, but is here solved simultaneously rather than sequentially from upstream to downstream, such that flows and demand delivery can be a function of upstream and downstream management choices. Unlike many water resource optimization tools, it explicitly handles the travel time in each reach.

For each lake or reservoir, a linearized version of Equation 4.10 is solved, one which assumes the lake area doesn't vary significantly over the time step:

$$\begin{aligned} \frac{A^n}{\Delta t}(h^{n+1} - h^n) &= \frac{1}{2}(Q_{in}^n + Q_{in}^{n+1}) - Q_{out} \\ &\quad + P \cdot A^n - E \cdot A^n - \frac{1}{2}\left(S(h^n) + S(h^{n+1})\right) \\ Q_{out} &= \frac{1}{2}\left(Q(h^n) + Q(h^{n+1})\right) \end{aligned} \tag{4.14}$$

where $Q(h^{n+1})$ is approximated by:

$$Q(h^{n+1}) = Q(h^n) + \left(\frac{dQ}{dh}\right)_{h^n} \cdot \left(h^{n+1} - n^n\right) \tag{4.15}$$

This linearized water management optimization is solved iteratively, updating the unknown reservoir stages, streamflows, delivered demands, and optimization slack variables in each iteration. After the water management optimization problem is solved, the delivered water demands are used in the standard non-linear RAVEN routing algorithm. This ensures hydrograph results are the same when running with or without water management optimization.

Environmental minimum flows are handled by defining a slack variable e^+ which is minimized as part of the objective function:

$$e^+ \geq Q - Q_E \tag{4.16}$$

where Q_E is the environmental minimum flow, and Q is the outflow from the subbasin reach. Because it is minimized, e^+ will be positive and non-zero if $Q > Q_E$, but zero when the environmental minimum flow is violated. This triggers the following constraint within the optimization problem:

$$\sum upstream D_i = 0 \tag{4.17}$$

i.e., that all of the restricted upstream demands are set to zero if the environmental minimum flow is not satisfied. Unrestricted demands are not influenced by environmental minimum flows.

The management optimization tool converts all of the reservoir management constraints in section 4.3 into corresponding management goals. Maximum stage and minimum stage constraints from the `:ReservoirMaxStage` and `:ReservoirMinStage` commands are treated as goals, as are maximum and minimum flow constraints specified using the `:ReservoirMaxFlow` and `:ReservoirMinFlow`. The `:TargetStage` command is likewise interpreted as a management goal. The `:ReservoirDownstreamFlow` and `:ReservoirDownstreamDemand` commands are currently incompatible with the demand optimization functionality, and should be replaced with equivalent management goals in the .rvm file.

Lastly, user-specified management constraints and goals may be specified to handle arbitrarily complex custom management scenarios. The details of how to implement these are supplied in appendix A.6.

The demand optimization process generates two new output files by default:

- `DemandOptimization.csv` - reports a time series of all decision variables calculated via the optimization process
- `GoalSatisfaction.csv` - reports a time series of all slack variables used to handle goals as soft constraints. Non-zero slack variables correspond to unmet goals, and their magnitude corresponds to the degree to which the goal was unmet.

4.6 Strategies for Simulating Lakes/Wetlands in Raven

Raven has multiple mechanisms in place for simulating lakes and key lake processes. Since most of the template models don't explicitly represent lakes, a user may wish to augment these models to support lake simulation.

The primary options are as follows:

1. Lakes as reservoirs- the mass balance of the lake is explicitly simulated using a level pool routing approach, and the reservoir outflow characteristics must be defined. This is the preferred approach for large lakes with significant outflows, large surface areas, and significant impact on the timing of these outflows. It is also necessary if one wishes to simulate reservoir outflow management. For all such Lakes, a :Reservoir-:EndReservoir command block will be required to define the outflow characteristics, and therefore these large lakes require additional parameters. It is recommended that these large reservoir-like lakes be represented using a single subbasin comprised of a single LAKE HRU (one using the special LAKE soil profile) or a single subbasin with a single LAKE HRU and a set of land HRUs which drain to the lake. In these lakes, the `:OW_Evaporation` method (not the `:LakeEvaporation` process) determines how the PET from lakes is calculated; this ET can be modified using the `LAKE_PET_CORR` land use parameter. With this option, groundwater seepage to and from the lake may also be simulated.
2. Lakes as water HRUs (approach #1)- for smaller lakes which are along the channel and have minimal impacts on flows, it is recommended not to represent these as reservoirs. On these lakes, all precipitation will be directed directly to the channel, and there is no need to define outflow characteristics. Evaporation from the lake surface may be simulated using the :LakeEvaporation hydrologic process, which removes water from the `SURFACE_WATER` in the channel at a rate proportional to the lake HRU surface area. It may be desirable to address the hydraulic impacts of such lakes by increasing the `MANNINGS_N` parameter (or other relevant routing parameters) of the subbasin in which the lake resides.
3. Lakes as water HRUs (approach #2)- for medium sized lakes which may strongly dissipate the basin scale response, we can likewise choose not to represent these explicitly as reservoirs. On these lakes, all precipitation will be directed directly to `LAKE_STORAGE`, and there is no need to define outflow characteristics. Evaporation from the lake surface may be simulated using the `:LakeEvaporation` hydrologic process, which removes water from the `LAKE_STORAGE` in at a rate proportional to the water HRU surface area. Water is released to `SURFACE_WATER` via a simple storage-outflow relationship defined using the `:LakeRelease` hydrologic process. For such lakes, the `:LakeStorage` variable must be set to `LAKE_STORAGE`. This approach supports more user control over lake impact than option #2, controlled by the land use parameter `LAKE_REL_COEFF`.
4. Off-channel lakes - for lakes not connected to the primary stream network, the user must first assess whether it makes sense to include them and their contributing area in the model, as such lakes may be hydrologically isolated from the rest of the watershed. If no action is taken, they will function identically to the lakes as water HRUs (option #2 or #3, above).

5. Lakes as depression storage - for (potentially ephemeral) lakes, ponds, or wetlands, it may be useful to simulate the storage capacity of the landscape using depression storage. Depression storage withholds some or all of the runoff locally in depression stores via an :Abstraction algorithm. This depression storage can lose water to the subsurface (via :Seepage), leak to the main surface water network (via :DepressionOverflow) and evaporate to the atmosphere (via :OpenWaterEvaporation). Because depression storage can interact with the soil layers below, it is suggested that a LAKE profile not be used to represent local depression storage if :Seepage is enabled. Typically, depression storage is used to represent lake extents much smaller than the HRUs of interest, however, larger lakes or wetlands systems could be more explicitly represented by abstracting laterally to WETLAND HRUs via, e.g., the :LateralFlush process, and there conditionally applying seepage, evaporation, and overflow.

6. Lakes in HBV [special case] - the HBV hydrological model Bergstrom (1995); Lindström et al. (1997) and its variants (HBV-EC (Hamilton et al., 2000), HBV-Lite) uses a somewhat curious conceptual model for lakes that requires overriding of some normal lake functioning. In this model, the second routing store (the 'slow reservoir', SOIL[2]) is treated as a 'Lake' (of sorts), subject to evaporation via the :LakeEvaporation algorithm, but also used as a routing store and recipient of water from the 'fast reservoir'. It is likely best not to think about this as a lake, but rather as a (somewhat curious) conceptual representation of depression storage in a landscape.

All Lake HRUs should be assigned (1) a LAKE soil profile (except wetlands, see above), (2) a WATER/LAKE land use type and (3) an OPEN/BARREN (or similar) vegetation type with zero LAI. The latter is required to ensure that no canopy interception or radiation corrections are applied to the lake surface. LAKE profiles

Any HRU whose soil profile is set to LAKE* does not have soil layers underneath. Infiltration, percolation, and baseflow are therefore not simulated. All precipitation (snow and rain) which falls on the lake HRU will be sent to the lake storage unit defined by the :LakeStorage command. This is, by default, SURFACE_WATER and should only be overridden for (1) use of option #3 above to simulate lakes and (2) emulation of HBV and its variants. If the HRU represents a Reservoir, this water is applied directly to the reservoir mass balance. If the lake IS NOT explicitly represented by a reservoir, then this water goes right into the channel system, and the only way to represent the impacts of the lake are via an artificially high Manning's roughness coefficient or equivalent routing parameter.

Lakes in HBV

By default, the :LakeStorage state variable is SURFACE_WATER. For HBV-EC or HBV light emulation this is modified to SOIL[2]. HBV treats all landscapes as if they have some presence of lakes, but does not explicitly treat LAKE HRUs. If you would like to revise HBV to properly handle precipitation on a lake, an additional command is required in the :HydrologicProcesses block:

```
:Flush RAVEN_DEFAULT [lake storage SV] [SURFACE_WATER]
:-->Conditional HRU_TYPE IS LAKE
```

Without this, the precipitation in the 'lake' storage unit may steadily increase because standard mechanisms for drainage (e.g., baseflow) are disabled for LAKE type HRUs.

4.6.1 Lake Evaporation

Lake Evaporation is only applied to LAKE HRUs which are not linked to a Reservoir by their HRU ID. If a lake HRU is linked to a Reservoir, ET is handled using the specified :OW_Evaporation algorithm (which can be corrected by the LAKE_PET_CORR modifier, a land use parameter). Evaporation from depression storage should be handled via the :OpenWaterEvaporation algorithm.

4.6.2 Lakes and in-catchment routing

The reach length in a subbasin with a lake or reservoir refers to the main channel upstream of the lake, and is therefore typically smaller than that of a subbasin without a lake. Likewise, the in-catchment routing parameters are only applied on the land portion of the subbasin. As such, the time of concentration and time to peak (or other in-catchment routing parameters) should be determined solely by looking at the non-lake portion of the subbasin. In the cases where the lake is treated as a single-HRU subbasin, the reach length should be zero and the time of concentration should be a very small number.

Chapter 5

Forcing Functions

In Raven, forcing functions, such as rainfall or incoming 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 hydrologic response 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. Many forcing functions may be estimated by Raven if field data is unavailable.

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.

5.1 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 internally 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.1. 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.

As of v2.8, Raven supports two different sets of gauges interpolants, i.e., only a subset of gauges must have temperature or precipitation. If an automated interpolation technique (e.g., `INTERP_NEAREST_NEIGHBOR`) is used, only the gauges that have available temperature data will be used to interpolate temperature; all other gauges will be given a weight of 0.0; likewise with precipitation. However, if the interpolation is user-supplied (`INTERP_FROM_FILE`), all gauges must have precipitation and temperature data. If windspeed/radiation/humidity data is provided, it must be provided at the same gauges where precipitation is available.

Note that values are first spatially interpolated, then corrected for orography. Orographic corrections are based upon the interpolated gauge elevation. This ensures, for instance, that an HRU directly between two met stations at different elevations doesn't get doubly corrected for orography - the interpolation already handles this. The interpolated gauge reference elevation is calculated as:

$$\hat{z}_k = \sum_{g=1}^{NG} w_{kg} \cdot z_g \tag{5.1}$$

where z_g is the elevation of the g^{th} gauge. Note that for a nearest neighbor interpolation, this is equivalent to standard orographic corrections from the nearest meteorological gauge.

5.1.1 Interpolation for Gridded Data

If gridded data in NetCDF format is supplied using the `:GriddedForcing` command, then the user must provide a means of mapping grid cell meteorological data to HRUs. This is done in a similar fashion to the `INTERP_FROM_FILE` option for meteorological gauges. Usually, one would generate the relative area coverage of each cell i in each HRU k using a GIS tool. The proper weighting for each grid cell in each HRU is then $w_{ki} = A_{ki}/A_k$ where A_{ik} is the area of the HRU and A_{ik} is the area of NetCDF grid cell i which overlaps HRU k. With this (specified using the `:GridWeights` command structure explained in appendix A.4.7), the forcings for each HRU may be calculated as:

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

$$V_k = \sum_{i=1}^{NC} w_{ki} \cdot V_i$$

i.e., any forcing (precip/temp/etc.) value V_k for HRU k is generated by weighting the values from all NC cells V_i, using an HRU-specific weighting factor w_{ki}. Note that $\sum_{i=1}^{NC} w_{ki} = 1$ 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.

If cell representative elevations are provided, the representative elevation of the meteorological data in each cell is given as

$$\hat{z}_k = \sum_{i=1}^{NC} w_{ki} \cdot \hat{z}_i$$

where $\hat{z}_i$ is the representative cell elevation (usually the mean ground elevation in each climate model cell). If not provided, it is assumed that no orographic corrections are applied, i.e., $\hat{z}_k = z_k$, where z_k is the elevation of the HRU. This is fairly reasonable approximation if the grid cells are small enough such that there is relatively small variability of elevation within each cell.

5.2 Temperature

Daily average, sub-daily, and daily minimum and maximum temperatures are required for many hydrologic 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 calculated directly. 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 `:SubdailyMethod` command. Lastly, in the worst case scenario where only daily average temperature is provided, the daily min, max, and sub-daily temperatures are also generated using the approach specified in the `:SubdailyMethod` command, but with the max and min calculated as the mean daily temperature plus or minus 4 degrees.

5.2.1 Orographic Temperature Effects

Orographic effects may be applied to correct temperature estimates at each HRU based on the specified elevation of the HRU relative to the local meteorological gauge. 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. Orographic corrections would typically not be applied when gridded temperature data is provided.

> **Advice**
>
> The `OROCORR_SIMPLELAPSE` is recommended for most applications, unless trying to emulate a specific model configuration (i.e., HBV or UBCWM). The UBCWM algorithm performs well on the western face of the Rockies.

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:

$$T = T_k - \alpha(z - \hat{z}_k) \tag{5.2}$$

where T is the estimated HRU temperature after correction, T_k is the temperature in the HRU interpolated from the gauge data, z and $\hat{z}_k$ are the elevation of the HRU and reference elevation at the gauge respectively, where $\hat{z}_k$ is calculated from equation 5.1, and α is the specified adiabatic lapse rate. Equation 5.2 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 global parameter `ADIABATIC_LAPSE` in the RVP file. Note that this correction is equivalent to the standard interpretation of lapse rates for nearest neighbor interpolation, i.e., 5.2 simplifies to:

$$T = T_g - \alpha(z - z_g) \tag{5.3}$$

where z_g is the elevation of the nearest gauge and T_g is the temperature at the nearest gauge.

HBV method (`OROCORR_HBV`)

The HBV model method from Bergstrom (1995); Lindström et al. (1997) employs the simple orographic temperature correction method described above employing Equation 5.2, except that the

monthly average temperatures are not lapsed to be consistent with their treatment in the standard HBV model.

UBCWM 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 keyword and parameter sequence:

```
:UBCTempLapseRates A0TLXM A0TLNM A0TLXH A0TLNH P0TEDL P0TEDU
```

The parameters listed above are described in Table 5.1.

Table 5.1: 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

$$V = \begin{cases} \min\left(\frac{P}{\text{A0PPTP}}, 1.0\right), & \text{if A0PPTP} > 0 \\ 0, & \text{if A0PPTP} \leq 0 \end{cases} \tag{5.4}$$

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 5.5. The wet and dry adiabatic lapse rates are specified in the RVP file using the `:WetAdiabaticLapse` and `:AdiabaticLapseRate` respectively.

$$\alpha_c = V\alpha_w + (1 - V)\alpha_d \tag{5.5}$$

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 5.6.

$$w_t = \frac{T_{max} - T_{min}}{\text{A0TERM}} \tag{5.6}$$

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 5.7 and 5.8 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.

$$\alpha_{max} = \begin{cases} w_t \text{A0TLXM} + (1 - w_t)\alpha_c, & \text{if elevation} \geq 2000 \text{ m} \\ w_t \text{A0TLXH} + (1 - w_t)\alpha_c, & \text{if elevation} < 2000 \text{ m} \end{cases} \tag{5.7}$$

$$\alpha_{min} = \begin{cases} w_t \text{A0TLNM} + (1 - w_t)\alpha_c, & \text{if elevation} \geq 2000 \text{ m} \\ w_t \text{A0TLNH} + (1 - w_t)\alpha_c, & \text{if elevation} < 2000 \text{ m} \end{cases} \tag{5.8}$$

UBCWM 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 5.9 and the relationship for the minimum daily temperature lapse rate is shown in Equation 5.10.

$$\alpha_{max} = \frac{T_{max2} - T_{max1}}{z_2 - z_1} \tag{5.9}$$

$$\alpha_{min} = \frac{T_{min2} - T_{min1}}{z_2 - z_1} \tag{5.10}$$

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.

5.3 Precipitation

Precipitation forcings (rainfall and snowfall) 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. Rainfall and snowfall may further be corrected for bias on a subbasin-by-subbasin basis using the subbasin properties `RAIN_CORR` and `SNOW_CORR`.

5.3.1 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 `:RainSnowFraction` command. All of the provided algorithms calculate the snow fraction α_s, and rain and snow are determined from:

$$\begin{aligned} R &= (1-\alpha_s)P \\ S &= \alpha_s P \end{aligned}$$

where R [mm/d], S [mm/d], and P are rainfall, snowfall, and total precipitation rates, respectively. The following algorithms for α_s are available:

Determine From Data (`RAINSNOW_DATA`)

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

Temperature range approach (`RAINSNOW_DINGMAN`)

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

$$\alpha_s = \frac{T_{trans} - T_{min}}{T_{max} - T_{min}} \tag{5.11}$$

where T_{trans} is the rain/snow transition temperature (global parameter `RAINSNOW_TEMP`) [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 ($\alpha_s = 1$) or all rain ($\alpha_s = 0$), accordingly. This snow fraction is applied for the entire day.

Temperature threshold approach (`RAINSNOW_THRESHOLD`)

In the temperature range approach, all precipitation is assumed to be snow when $T_{ave} < T_{trans}$, and rain otherwise, where T_{trans} is the rain/snow transition temperature (global parameter `RAINSNOW_TEMP`) [default: 0 °C].

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, T_{ave}:

$$\alpha_s = 0.5 + \frac{T_{trans} - T_{ave}}{\Delta T} \tag{5.12}$$

in the range from $T_{trans} - \Delta T/2$ to $T_{trans} + \Delta T/2$, where T_{trans} is the rain/snow transition temperature (global parameter `RAINSNOW_TEMP`, [°C]) and ΔT is the global parameter `RAINSNOW_DELTA` [°C]. If T_{ave} is outside of this temperature range, the precipitation is either all snow ($\alpha_s = 1$) or all rain ($\alpha_s = 0$), accordingly. This snow fraction is applied for the entire day.

Harder and Pomeroy Approach(`RAINSNOW_HARDER`)

Using the method of (Harder and Pomeroy, 2013) as implemented in CRHM Pomeroy et al. (2007). The fraction of snow is given by:

$$\alpha_s = 1 - \frac{1}{1 + 2.503 \cdot 8^{-T_{ib}}} \tag{5.13}$$

where T_{ib} is the ice bulb temperature, as determined from the relative humidity and air temperature.

HSPF Approach(`RAINSNOW_HSPF`)

Using the empirical formula as documented in the HSPF manual Bicknell et al. (1997) (here converted to Celsius). A reference temperature T^* is determined using

$$T^* = T_{trans} + (T_{ave} - T_{dp})(0.5808 + 0.0144 * T_{ave}) \tag{5.14}$$

where T_{trans} is the is the rain/snow transition temperature (global parameter `RAINSNOW_TEMP`), T_{dp} is the dew point temperature, calculated from the relative humidity. If $T^* > T_{ave}$, α_s is one, zero otherwise.

SNTHERM89 Approach(`RAINSNOW_SNTHERM89`)

Uses the parameter-free empirical formula from the SNTHERM89 model (Jordan, 1991), also used in Noah-MP3.6.

$$\alpha_s = \begin{cases} 0.0 & \text{if } T_{ave}\text{>2.5} \\ 0.6 & \text{if } T_{ave}\text{>2.0} \\ 1.0 - 0.2666 \cdot (T_{ave} - 0.5) & \text{if } T_{ave}\text{>0.5} \\ 1.0 & \text{otherwise} \end{cases} \tag{5.15}$$

where T_{ave} is the is the air temperature for the time step.

Wang Approach(`RAINSNOW_WANG`)

Uses the parameter-free empirical sigmoid function formula from (Wang et al., 2019):

$$\alpha_s = \frac{1}{1 + (6.99 \cdot 10^{-5}) \exp(2.0(T_{wet} + 3.97)} \tag{5.16}$$

where T_{wet} is the wet bulb temperature, calculated from air temperature, relative humidity, and air pressure.

5.3.2 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.

Simple method (OROCORR_SIMPLELAPSE)

The simple precipitation lapse rate method employs a simple linear adiabatic method as outlined in Equation 5.17 below:

$$P = P_k + \alpha(z - \hat{z}_k) \tag{5.17}$$

where P is the total precipitation rate [mm/d], P_k is the interpolated precipitation at the HRU [mm/d], z is the elevation of the HRU, $\hat{z}_k$ is the reference elevation calculated from equation 5.1 at the HRU, and α [mm/d/km]is the precipitation correction lapse rate specified using the global parameter PRECIP_LAPSE. Checks are included to ensure positivity of the precipitation rate. Note that this simplifies to the traditional interpretation of gauge orographic corrections for a single gauge or nearest neighbor interpolation algorithm, i.e.,

$$P = P_g + \alpha(z - z_g) \tag{5.18}$$

where z_g is the elevation of the nearest gauge and P_g is the total precipitation rate at the nearest gauge [mm/d]. This relation is only applied when precipitation at the gauge is non-zero.

HBV method (OROCORR_HBV)

From the HBV model Bergstrom (1995); Lindström et al. (1997):

$$P = P_k \cdot (1.0 + \alpha(z - \hat{z}_k)) \tag{5.19}$$

where P is the total precipitation rate [mm/d], P_k is the interpolated precipitation at the HRU [mm/d], z is the elevation of the HRU, $\hat{z}_k$ is the reference elevation calculated from equation 5.1 at the HRU, and α, the precipitation correction lapse rate, is 0.00008 m^{-1} below 5000 masl, 0 above this elevation.

UBCWM 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 5.20:

$$P = P_g \cdot (1 + \alpha F_t)^{\frac{z - z_g}{100}} \tag{5.20}$$

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 5.21:

$$F_t = \begin{cases} 1, & \text{if } t_{band} \leq 0 \text{ C} \\ 1 - \text{A0STAB}\,(t_{band})^2, & \text{if } t_{band} > 0 \text{ C} \end{cases} \tag{5.21}$$

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.

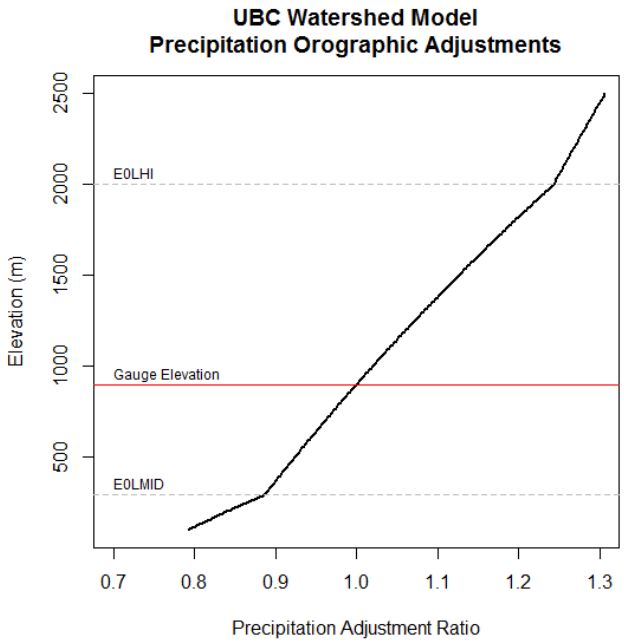

Figure 5.1: UBC Watershed Model Orographic Correction

5.4 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,

$$e_s(T) = 0.6108 \cdot \exp\left(\frac{17.23T}{T + 237.3}\right) \tag{5.22}$$

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

$$\Delta = \frac{4098}{(T + 237.3)} \cdot e_s(T) \tag{5.23}$$

where T is in $^\circ$C. The latent heat of vaporization of water, λ_v [MJ/kg], is estimable by:

$$\lambda_v = 2.495 - 0.002361 * T \tag{5.24}$$

where T is the temperature [$^\circ$C], and the psychrometric constant, γ is here treated as varying with atmospheric pressure, p [kPa],

$$\gamma = \frac{c_a}{0.622 \cdot \lambda_v} p \tag{5.25}$$

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

Note that most of the algorithms below estimate daily PET. Methods are required to downscale these daily estimates to sub-daily time steps, as discussed in 5.10. If the `:DirectEvaporation` command is used, rainfall is automatically reduced by PET, with a corresponding decrease in the available potential evapotranspiration.

5.4.1 PET Estimation

Disabled PET (`PET_NONE`)

The daily PET value used is constant and uniform rate of 0 mm/d. Used (for instance) in routing-only mode to disable over-water evaporation.

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); Lindström et al. (1997). 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:

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

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 spatially and temporally interpolated from the gauge values for `:MonthlyAveEvaporation` and `:MonthlyAveTemperature`. 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,

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

where λ_v [MJ/kg] is the latent heat of vaporization of water, ρ_w [kg/m^3] is the density of water, $\Delta = de_s/dT$ is the slope of the saturated vapor pressure curve, $R_n = S_n + L_n$ [MJ/m^2/d] 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,

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

where c_{can} [m/d] is the canopy conductance, and γ [kPa/ °C] is calculated using 5.25. The final expression is converted from m/d to mm/d. The atmospheric conductance is calculated using the following relationships Dingman (2002):

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

where κ 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):

$$c_{can} = 0.5 \cdot c_{leaf} \cdot \text{LAI}$$

where c_{leaf} is the leaf conductance [m/d], calculated using the expressions detailed in Dingman (2002) and LAI is calculated from equation 3.1.

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:

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

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

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

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:

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

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 deficit-driven ET; this coefficient may be overridden for individual land uses using the `PRIESTLEYTAYLOR_COEFF` land use parameter. Priestley Taylor is predominantly used to estimate evaporation from open water.

Hargreaves (`PET_HARGREAVES`)

From Hargreaves and Samani (1982).

$$\text{PET} = \frac{1}{\rho_w \lambda_v} \cdot S_{ET} \cdot 0.000938 \cdot \left(\sqrt{T^{mon}_{max,F} - T^{mon}_{min,F}}\right) T_{ave,F}$$

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 Fahrenheit, and $T_{ave,F}$ is the daily temperature in Fahrenheit (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 uses metric units.

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

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.

Monthly factor method (`PET_MONTHLY_FACTOR`)

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

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

where E_{mon} [mm/d/K] is a monthly PET factor (specified using the `:MonthlyEvapFactor` command in the .rvt file, on the order of 0.2), T_{ave} is the daily average temperature in Celsius, and δ_{forest} is the land use parameter `FOREST_PET_CORR`), applied only to forested regions. The factor ϵ is an orographic correction factor, given as

$$\epsilon = 0.009 \cdot (z - \hat{z}_k)$$

where z is the HRU elevation, and $\hat{z}_k$ is the reference elevation for the HRU calculated using equation 6.1, respectively. Note that, unlike with most other ET approaches, the orographic correction is necessarily fused with the PET calculation and therefore the orographic PET correction should be set to `OROCORR_NONE`. In previous versions (prior to 3.0), $\hat{z}_k$ was the elevation of the first gauge in the .rvt file.

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.

Oudin PET (`PET_OUDIN`)

A simple method from Oudin et al. (2005). PET is calculated using the following relationship:

$$\text{PET} = \frac{S_{ET}}{\lambda_v \rho_w} \cdot \min\left(\frac{T_{ave} + 5.0}{100}, 0.0\right)$$

where S_{ET} is the shortwave extraterrestrial radiation [MJ/m^2/d], λ_v is the latent heat of vaporization, ρ_w is the density of water, and T_{ave} is the daily average temperature.

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.

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

where the PET is in mm/d, T_{ave} is the average daily temperature [$^\circ$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/$^\circ$C], γ is the psychrometric constant, and S_n is the net incoming solar radiation [MJ/m^2/d].

MOHYSE (`PET_MOHYSE`)

From the MOHYSE model (Fortin and Turcotte, 2006):

$$\text{PET} = \frac{c}{\pi} \cdot \cos^{-1}\left(-\tan(\Lambda) * \tan(\delta)\right) * \exp\left(\frac{17.3 \cdot T}{238 + T}\right)$$

where c is the global parameter `MOHYSE_PET_COEFF`, Λ is the latitude in radians, δ is the solar declination, and T is the average daily temperature in $^\circ$C.

Granger-Gray (`PET_GRANGERGRAY`)

From Granger and Gray (1989), as implemented in the Cold Regions Hydrologic Model (CRHM) Pomeroy et al. (2007):

$$\text{PET} = \frac{1}{\lambda_v \rho_w} \frac{\Delta \cdot 0.9 \cdot R_n + \gamma \cdot D^* \cdot (e_s - e_a)}{\Delta + \gamma / G}$$

where $\Delta = de_s/dT$ is the slope of the saturated vapor pressure curve, γ is the psychometric constant, $R_n = S_n + L_n$ [MJ/m^2/d] is the net radiation to the system, $e_s(T)$ is the saturated vapour pressure and e is the actual vapour pressure. The 0.9 correction for net radiation assumes that 10 percent of energy goes to ground heat flux. Here, the formulation uses a drying power, D^* [MJ/m^2/d/kPa], given as

$$D^* = \lambda_v \rho_w \left((8.19 + 22 z_0) + (1.16 + 8 z_0) \cdot v\right)$$

where z_0 is the vegetation roughness height [m] and v is the wind velocity [m/s]. The correction term G is given by

$$G = \frac{1.0}{0.793 + 0.2 \exp(4.902 D)} + 0.006 D$$

where

$$D = \frac{1}{1 + \frac{R_n}{D^* \cdot (e_s - e_a)}}$$

where e_s [kPa] is the saturated vapour pressure and e_a is the actual vapour pressure. Note that in Granger and Gray (1989), this 'potential' evapotranspiration rate is an actual evaporation rate constrained by moisture availability. The formula for G is similar to that of Granger and Gray (1989) in trend, but not precisely the same.

Linacre (`PET_LINACRE`)

From Linacre (1977).

$$PET = \frac{a\frac{T_{ave}}{100-\phi} + 15 \cdot (F_{ave} - T_{dew})}{80.0 - T_{ave}}$$

where T_{ave} is the average daily temperature, T_{dew} is the daily dew point temperature, ϕ is the latitude, in degrees. The parameter a is equal to 700 for open water evaporation, 500 otherwise.

Vapor Deficit (`PET_VAPDEFICIT`)

Linear variation of PET with vapour deficit:

$$PET = C \cdot (e_s - e_a)$$

where e_s [kPa] is the saturated vapour pressure and e_a is the actual vapour pressure. The parameter C is the land use parameter `PET_VAP_COEFF`.

Blended (`PET_BLENDED`)

Allows the PET to be estimated using a weighted average of two or more PET algorithms described above, in a method consistent with the `:ProcessGroup` command, though with a different syntax. This method must be used in combination with the `:BlendedPETWeights` command in the .rvi file to specify the PET algorithms included in the blended group, as well as the associated weights or weight-generating parameters.

5.4.2 PET Orographic Effects

Orographic effects are calculated using the following algorithms, specified using the `:OroCorrPET` command in the .rvi file. Note that these should typically only be applied if PET data is provided at the gauge; otherwise, temperature orographic corrections will already impact PET estimates.

HBV method (`OROCORR_HBV`)

From the HBV model (Bergstrom, 1995; Lindström et al., 1997):

$$\text{PET} = \text{PET}_k \cdot \alpha\,(1-\beta)\,(z - \hat{z}_k)$$

where PET_k is the interpolated gauge-provided PET rate [mm/d], α is the global PET correction factor (`GLOBAL_PET_CORR`), β is the HBV precip correction factor (`HBV_PRECIP_CORR`), z is the HRU elevation, and $\hat{z}_k$ is the reference elevation for the HRU calculated using equation 5.1, respectively.

PRMS method (`OROCORR_PRMS`)

This orographic correction factor is described in the users's manual of the PRMS model (Leavesley et al. (1983)). 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).

$$\mathrm{PET} = \mathrm{PET}_k \cdot \left(\frac{1}{68 - 0.0118z + \frac{650}{e_{sat}^{max} - e_{sat}^{min}}} \right)$$

where z is the HRU elevation [masl] and PET_k is the interpolated PET at the HRU (implicitly 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.

5.5 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]:

$$\Gamma = \frac{2\pi J}{365} \tag{5.26}$$

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

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

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

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

$$S_n = (1 - \alpha) \cdot f_{can} \cdot f_{cloud} \cdot S_{cs} \tag{5.27}$$

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

$$S_{cs} = f_{atm} \cdot f_{asp} \cdot S_{ET} \tag{5.28}$$

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 5.5.1 details methods for calculating S_{ET}, section 5.5.2 details methods for handling f_{atm}, section 5.5.3 details methods for handling f_{cloud} and section 5.5.4 details methods for handling f_{can}. These individual terms may be visualized in figure 5.2.

5.5.1 Extraterrestrial Shortwave Generation

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

Raven default (`SW_RAD_DEFAULT`)

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

$$S_{ET} = I_{sc} \cdot E_0 \cdot [\cos(\delta) \cdot \cos(\Lambda) \cdot \cos(2\pi t) + \sin(\delta)\sin(\Lambda)] \tag{5.29}$$

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.

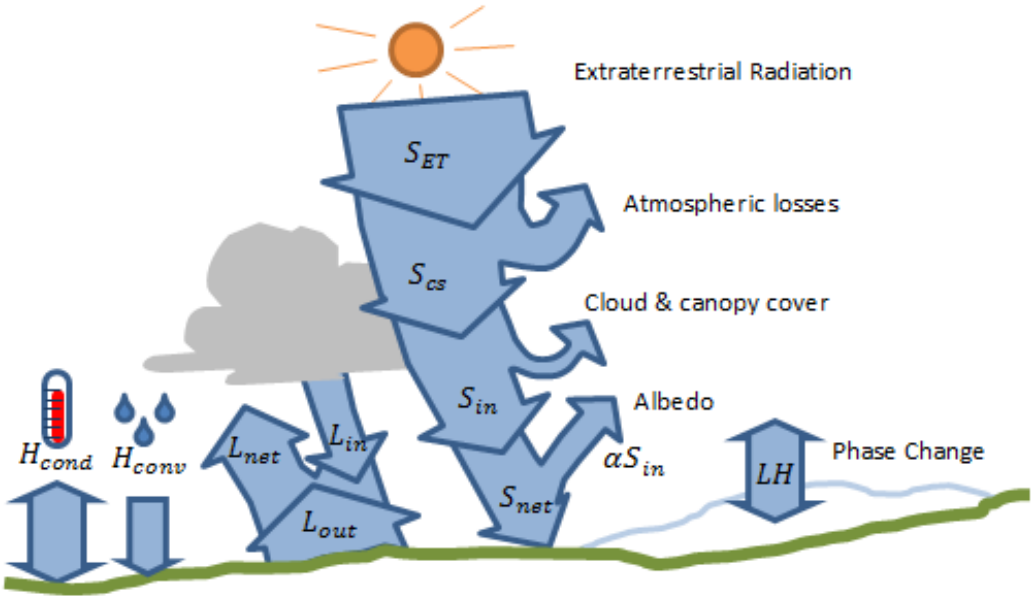

Figure 5.2: The surface energy balance in Raven. Important components include shortwave radiation (S), longwave radiation (L), conductive/convective heat transfer (H), and latent heat/phase change (LH). This figure indicates magnitudes only; the sign convention used in Raven is such that incoming radiation is positive, while outgoing is negative, thus L_{out} will always be negative, and L_{net} will typically be negative.

UBCWM approach (`SW_RAD_UBCWM`)

Shortwave radiation is calculated using the same equations as the `SW_RAD_DEFAULT` approach (equation 5.29), 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 5.29, 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 or on a grid (as generated by a climate model). 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 number. What is actually being input is

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

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

5.5.2 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

$$\begin{aligned} \tau_{dir} &= \exp\left(-0.124 - 0.0207W_p - (0.0682 + 0.0248W_p)M_{opt}\right) \\ \tau_{diff} &= \exp\left(-0.0.363 - 0.0084W_p - (0.0572 + 0.0173W_p)M_{opt}\right) \end{aligned} \tag{5.30}$$

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).

UBCWM 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

$$m_a = \frac{1 - 0.001 \cdot z}{[\cos(\delta) \cdot \cos(\Lambda) \cdot \cos(2\pi t) + \sin(\delta)\sin(\Lambda)]} \tag{5.31}$$

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.

5.5.3 Cloud Cover Corrections

Additional corrections are required to handle cloud cover. While the algorithms for estimating actual cloud cover are included in section 5.7 below, the use of the cloud cover factor for estimating incident radiation is treated here.

UBC approach (`SW_CLOUD_CORR_UBC`)

The UBC watershed model (Quick, 1995) corrects shortwave radiation due to cloud cover using the following equation

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

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.

Dingman approach (`SW_CLOUD_CORR_DINGMAN`)

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

$$f_{cloud} = (0.355 + 0.68 \cdot (1 - C_c)) \tag{5.32}$$

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

Annandale approach (`SW_CLOUD_CORR_ANNANDALE`)

The cloud cover correction factor is generating using the approach from Annandale et al. (2002):

$$f_{cloud} = 0.16 * (1.0 + 0.00027z)\sqrt{(T_{max} - T_{min})} \tag{5.33}$$

where z is the elevation in masl, and T_{max}/T_{min} are the maximum and minimum daily temperature.

5.5.4 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`).

UBCWM method (`SW_CANOPY_CORR_UBCWM`)

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 transmittance 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):

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

where k is the extinction coefficient, LAI is the leaf-area index and SAI is the stem-area index, both estimated as indicated in equation 3.1. 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.

5.6 Longwave Radiation

Longwave radiation is the electromagnetic radiation emitted by materials with near-earth-surface temperatures. The net longwave is the difference between the incoming longwave emitted (or back scattered) by the atmosphere, clouds, and canopy and the outgoing radiation from the land surface (see figure 5.2).

5.6.1 Incoming Longwave Radiation

Raven Default method (`LW_INC_DEFAULT`)

By default, incoming longwave radiation is not calculated, and only net longwave is estimated. The exception is if `LW_RAD_DEFAULT` is used, in which case the `LW_INC_DINGMAN` is used by default.

Interpolate from data (`LW_INC_DATA`)

The incoming longwave radiation (in $MJm^{-2}d^{-1}$) is read from a file, specified at one or more gauge locations or on a grid (as generated by a climate model). The radiation could be either measured, generated from an atmospheric model, or estimated using an external preprocessor.

Skyview method (`LW_INC_SKYVIEW`)

A simple skyview method with air emissivity estimated from (Prata, 1996).

$$W_{inc} = (0.95 \cdot v + \epsilon_{air} \cdot (1 - v)) \cdot \sigma T_{air}^4$$

where v is the sky view factor, corrected for forest cover, σ is the Stefan-Boltzmann constant, T_{air} is the air temperature in Kelvins, and ϵ_{air} is the air emissivity, calculated as:

$$\epsilon_{air} = 1 - (1 + \eta) \exp(-\sqrt{1.2 + 3\eta})$$

where

$$\eta = 46.5 \cdot \frac{e_a}{T_{air}}$$

and e_a is the vapour pressure, in hPa.

Sicart et al. method (`LW_INC_SICART`)

(From Sicart et al. (2006))

$$L_{inc} = L_0 \cdot v + L_F \cdot (1 - v)$$

where v is the sky view factor, corrected for forest cover. The canopy contribution to the incoming radiation is:

$$L_F = 0.98 \cdot \sigma T_{air}^4$$

where σ is the Stefan-Boltzmann constant, and T_{air} is the air temperature in Kelvins. The atmospheric contribution to the incoming radiation is:

$$L_0 = \epsilon_{air} * (1.0 + 0.44 * \mathrm{RH} - 0.18 * tau) \cdot 0.864$$

where the factor 0.864 converts from watts to $MJ/m^2/d$, τ is the ratio of shortwave radiation on an inclined plane to that on a horizontal surface, and the air emissivity is given by **?**:

$$\epsilon_{air} = 1.24 \left(e_a * / T_{air}\right)^{1/7};$$

where e_a is the atmospheric vapour pressure, in hPa.

Dingman method (`LW_INC_DINGMAN`)

The approach documented in Dingman (2002), which uses the air emissivity of **?**, adjusted for forest cover (without skyview correction) and cloud cover.

$$L_{inc} = ((1 - F_c)\epsilon_{air} + 0.99(F_c)) \cdot \sigma T_{air}^4$$

where σ is the Stefan-Boltzmann constant, and T_{air} is the air temperature in Kelvins.

$$\epsilon_{air} = 1.72 \cdot \left(\frac{e_a}{T_{air}}\right) (1.0 + 0.22 * f_{cloud}^2)$$

where f_{cloud} is the cloud cover, and e_a is the vapour pressure.

5.6.2 Net Longwave Radiation

Raven 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.

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

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)

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

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.

Interpolate from data (`LW_RAD_DATA`)

The net longwave radiation (in MJm^{-2}d^{-1}) 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.

UBCWM 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 mm/d, T_{avg} °Cis the daily average temperature, T_{min} °Cis the daily minimum temperature, f is the UBC cloud cover correction factor (see Section 5.7), 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 mm/d, t_{avg} is the daily average temperature, and f_{LW} is the temperature multiplier factor in mm/dK^{-1} which is set in the RVP file using the `:UBCLWForestFactor` keyword. If the forest cover for an HRU is greater than zero then the latter equation is employed. Note that this expression is a linearization of the Stefan-Boltzmann law.

HSPF method (`LW_RAD_HSPF`)

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

$$L_n = 0.361 * (T_{avg} - 6.6) \tag{5.34}$$

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

No method (`LW_RAD_NONE`)

Net longwave radiation is not calculated. Usually used for simple models where radiation is not used.

5.7 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.

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.

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).

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

$$\Delta T = T_{max} - T_{min} \tag{5.35}$$

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

$$C_c = \begin{cases} 1, & \text{if } \Delta T \leq T_{cmin} \\ 1 - \frac{\Delta T - T_{cmin}}{T_{cmax} - T_{cmin}}, & \text{if } T_{cmin} > \Delta T > T_{cmax} \\ 0, & \text{if } \Delta t \geq T_{cmax} \end{cases} \tag{5.36}$$

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.

5.8 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

5.8.1 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.

> **Advice**
>
> These methods generally fall in the category of degree day methods (e.g., POTMELT_DEGREE_DAY, POTMELT_HBV, POTMELT_HMETS), or full energy-balance based methods (e.g., POTMELT_UBCWM, POTMELT_EB). For regional and/or data-poor applications, the degree day methods (which use only temperature) are preferred, as the energy balance based methods require reasonably accurate estimates of wind speed, relative humidity, and radiation. For applications where you are confident in estimates of these quantities, the energy balance methods may be preferred, and will be more sensitive to things like rain-on-snow.

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, but can be set with the land use parameter DD_MELT_TEMP), and M_a is the melt factor [mm/day/deg C], specified using the land use/land type parameter MELT_FACTOR.

No method (POTMELT_NONE)

The potential melt is set to zero. Only to be used when snow is not simulated (e.g., when routing output from a land surface scheme).

UBC approach (POTMELT_UBCWM)

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 (Bergstrom, 1995; Lindström et al., 1997) is given by a corrected version of the degree day approach, with the corrected melt coefficient given by

$$M'_a = C_f \cdot C_a \left(M_{a.min} + (M_{a.max} - M_{a.min}) \cdot \frac{1.0 - \cos(\Gamma - \Gamma_s)}{2} \right) \tag{5.37}$$

where M'_a is the potential melt coefficient, C_f is the forest correction factor, C_a 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 5.26 and Γ_s is the winter solstice angle and is a model constant of 23.5°. The forest and aspect correction factors developed for HBV-EC Hamilton et al. (2000) are described below:

$$C_f = (1.0 - F_c) \cdot (1.0 + (F_c) \cdot M_{RF}) \tag{5.38}$$

$$C_a = \max(1 - A_m \cdot C_s \cdot \cos(\theta), 0.0) \tag{5.39}$$

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:

$$C_s = (1.0 - F_c) \cdot (1.0 + (F_c) \cdot \sin(\theta_s)) \tag{5.40}$$

where θ_s is the landscape slope.

HBV method with rain-on-snow (`POTMELT_HBV_ROS`)

Identical to the `POTMELT_HBV` algorithm, but with an additive term for melt due to rain on the snowpack:

$$M_{melt} = M_{melt} + \frac{c_p^w}{\lambda_f} \cdot R \cdot \max(T, 0)$$

where c_p^w is the specific heat of water, λ_f the latent heat of fusion, R is the rainfall rate in mm/d, T is the air temperature, and α is the land use parameter `RAIN_MELT_MULT`, which defaults to 1.0.

Restricted method (`POTMELT_RESTRICTED`)

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

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

where S_n and L_n are the net incoming radiation, and the melt factor, M_a is the land surface parameter `MELT_FACTOR`. λ_f and ρ_w are the latent heat of fusion [MJ/kg] and density of water [kg/m^3], respectively. An additional factor in the latter portion of the equation converts from meters to millimeters.

Energy balance method (`POTMELT_EB`)

Similar to the `POTMELT_UBCWM` 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.

U.S. Army Corps method (`POTMELT_USACE`)

The U.S. Army Corps of Engineers potential melt model (U.S. Army Corps of Engineers, 1998) takes into account various factors including solar radiation, wind, and long-wave radiation exchange. The equation combines several melt equations, depending on the physical characteristics of the hydrologic response unit (HRU) and precipitation. These melt estimates include shortwave radiation melt, long-wave radiation melt, convection (sensible heat) melt, condensation (latent heat) melt, rain melt, and ground melt. Requires the parameter `WIND_EXPOSURE`, which represents the mean exposure of the HRU to wind considering topographic and forest effects; for open areas this would be equal to 1.0, but may be as low as 0.3 for forested areas. Details may be found in U.S. Army Corps of Engineers (1998).

HMETS method (`POTMELT_HMETS`)

A revised degree day model from the HMETS model (Martel et al., 2017), which uses a degree day factor which varies with cumulative snowmelt. The degree day model is given as

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

where T is the daily average temperature. T_f is the melt temperature [°C] (zero by default, but can be set with the land use parameter `DD_MELT_TEMP`), and M_a [mm/d/°C] is the degree day melt factor, calculated as a function of cumulative melt:

$$M_a = \min\left(M_a^{max}, M_a^{min} \cdot (1 + \alpha \cdot M_{cumul})\right)$$

where the following land use parameters are used: the maximum melt rate M_a^{max} [mm/d/ °C] (MAX_MELT_FACTOR), the minimum melt rate M_a^{min} [mm/d/ °C] (MIN_MELT_FACTOR), and α [1/mm] is the parameter DD_AGGRADATION.

CRHM EBSM method (POTMELT_CRHM_EBSM)

A parameter-free energy-based potential melt model from the Cold Regions Hydrologic Model (CRHM) (Pomeroy et al., 2007).

$$M_{melt} = \frac{1}{\lambda_v \rho_w}(S_n + L_n + Qh + Qp)$$

where convective/conductive heat transfer Q_h [MJ/m^2/d] is estimated from wind velocity, v [m/s], and maximum daily temperature, T_{max} [°C],

$$Q_h = -0.92 + 0.076 \cdot v + 0.19 \cdot T_{max}$$

and the energy content of the rainfall is given by the rainfall R [mm/d] and air temperature T [°C]:

$$Q_p = c_p \rho_w \cdot R \cdot \max(T, 0.0)/1000$$

where c_p and ρ_w are the specific heat capacity and density of water; the 1000 factor converts rainfall to m/d.

Blended (POTMELT_BLENDED)

Allows the potential melt to be estimated using a weighted average of two or more potential melt algorithms described above, in a method consistent with the :ProcessGroup command, though with a different syntax. This method must be used in combination with the :BlendedPotMeltWeights command in the .rvi file to specify the potential melt algorithms included in the blended group, as well as the associated weights or weight-generating parameters.

Not calculated (POTMELT_NONE)

Potential melt is not calculated by the model.

5.9 Atmospheric Variables

This section includes various methods for estimating wind speed, relative humidity, and air pressure.

5.9.1 Wind Speed

The following methods can be used to estimate the wind speed at 2 meters, as used for a number of ET and potential melt estimation algorithms. All of the following estimates will be adjusted with the land cover/land use parameter WIND_VEL_CORR (a positive multiplier), with the constraint that relative humidity must be greater than zero.

Constant wind velocity (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.

UBCWM 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`:

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

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.

5.9.2 Relative Humidity

The following algorithms may be used to estimate relative humidity in Raven. All of the following estimates will be adjusted with the land cover/land use parameter `RELHUM_CORR` (a positive multiplier), with the constraint that relative humidity must be greater than zero.

Constant humidity (`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 temperature, 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. This is the preferred algorithm when no relative humidity data is available.

5.9.3 Air Pressure

The following approaches may be used to estimate atmospheric pressure:

Constant air pressure (`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.

UBCWM approach (`AIRPRESS_UBC`)

From Quick (1995). Air pressure is corrected for elevation above mean sea level, z,

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

where P is in kPa and z is the HRU elevation in metres above sea level.

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 P is in kPa, T_{ave}^{K} is the average temperature for the time step in $^{\circ}$K, and z is the HRU elevation in metres above sea level.

5.10 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

No sub-daily correction (`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 sub-daily 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`)

When hourly temperatures are available, melt and ET are scaled by the hourly temperature in excess of zero degrees Celsius, i.e.,

$$\delta = \frac{\max(T, 0)}{\sum_{n=1}^{24} \max(T_i, 0)}$$

5.11 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.

Relative to first day of month (`MONTHINT_LINEAR_FOM`)

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

Relative to middle day of month (`MONTHINT_LINEAR_MID`)

Monthly values are linearly interpolated, assuming that the specified monthly values correspond to the middle of the month.

Relative to 21st day of the month (`MONTHINT_LINEAR_21`)

Monthly values are linearly interpolated, assuming that the specified monthly values correspond to the 21st day of the month (as done in the UBC Watershed model (Quick, 1995)).

Chapter 6

Forecasting and Assimilation

RAVEN has a number of built-in features to support short-term flood and reservoir inflow forecasting.

6.1 Streamflow Assimilation- Direct Insertion

RAVEN can integrate real-time observations of streamflow into its forecasting model via a simple and unique direct insertion algorithm. For more complicated applications of data assimilation (e.g., ensemble Kalman Filter or similar), it is recommended to use external tools.

When supplied with streamflow observations (using the `:AssimilateStreamflow` command in the .rvi file), RAVEN will override the stream discharge simulated at any point in the domain with streamflow observed at a subbasin outlet. Only the observations at subbasins indicated using the `:AssimilateStreamflow [SBID]` command in the .rvt file will be assimilated. In addition to overriding the flows during hindcasting (when observations are available), it can also propagate a scaled flow forward in time and upstream in space, with the key assumption that the ratio of the observed to simulated flow is more likely to be constant than not. The following scaling relationship is used:

$$\omega_i(t, x_i, t_i^{last}) = 1 + \alpha \cdot \left(\frac{Q^i_{obs}(t) - Q^i_{mod}(t)}{Q^i_{mod}(t)} \right) \cdot \exp(-\beta x_i) \cdot \exp(-\gamma t - t_i^{last})$$

where $\omega_i(t)$ is the weighting factor in subbasin i at time t, $Q^i_{obs}(t)$ is the observed flow at the nearest downstream observation location from basin i at time t, $Q^i_{mod}(t)$ is the simulated flow at the same location at time t (which may reflect the impact of upstream data assimilation in previous time steps), x_i is the downstream distance to the nearest downstream observed flow, and t_i^{last} is the time, in days, since the last observation was observed at this location (typically equal to t in hindcasting). The parameter α (global parameter `ASSIMILATION_FACT`) indicates the unit-less degree of assimilation, where $\alpha = 1$ corresponds to full insertion and $\alpha = 0$ corresponds to none. The parameter β (global parameter `ASSIM_UPSTREAM_DECAY`, with units of 1/km) determines how the scaling factor decreases with distance to the observation. For $\beta = 0$, the scaling factor persists to the headwater of the basin, scaling all upstream flows by the ratio of observed to simulated discharge at the downstream observation; for $\beta > 0$, the persistence last (with diminishing influence with distance) for a distance of roughly $4/\beta$. The parameter γ (global parameter `ASSIM_TIME_DECAY`, with units of 1/d) determines how the scaling factor diminishes with time after the observations are unavailable, such that the assimilated flows converge upon the simulated unassimilated flows at some time forward in the forecast. The scaling factor likewise lasts for a duration of roughly $4/\gamma$, after which assimilation influences ceases and the simulated flow dominates. Both global assimilation parameters are specified in the .rvp file.

In all basins upstream of an observation point, the streamflows and rivulet flows in the basin are scaled using the following:

$$Q'_{mod} = \omega_i \cdot Q_{mod}$$

For basins without a downstream observation point, the scaling factor ω_i is one, and no assimilation is performed.

In this algorithm, the closest downstream observation is always used for scaling. If there is assimilation data at more than one basin downstream, the scaling factor from most immediately downstream takes precedence, i.e., the assimilation data influence will not propagate upstream past another assimilation location.

6.2 Reservoir Stage Assimilation - Direct Insertion

Raven can integrate real-time observations of reservoir stage into a forecasting model via direct insertion. If the `:AssimilateReservoirStage` command is present in the .rvi file, all available lake or reservoir stage observations will be used to override the simulated stage at the start of each time step; assimilation will not be performed in cases where there are blank data values in the observation time series. The reservoir stage reported by Raven will be the simulated stage at the end of each time step after applying the reservoir mass balance. The observed reservoir/lake stage is supplied using `:ObservationData` command in the .rvt file with the `RESERVOIR_STAGE` tag (see appendix A.4.2). This algorithm is necessarily simpler than the streamflow assimilation algorithm above because it doesn't propagate upstream.

To disable reservoir assimilation for a specific reservoir, the observation data for that reservoir must not be read in by Raven.

For more complicated applications of reservoir data assimilation (e.g., ensemble Kalman Filter or similar), it is recommended to use external tools.

6.3 State Nudging

Forecasters can modify states directly in the .rvc initialization file, but can also apply additive or multiplicative changes to states such as soil moisture or snow water equivalent using the `:Nudge` command, also specified from the .rvc file. This nudging can be applied globally to the state variable or to a specified HRU group. See appendix A.5 for details.

6.4 Ensemble Kalman Filter

Raven supports data assimilation of observed hydrograph data using the Ensemble Kalman Filter (EnKF) methodology. The user gets to specify how the ensembles are generated via perturbation of forcings, how many members are included in the ensemble, the nature of any observational error, and which model internal states get updated during assimilation. This approach cannot be used in combination with the direct assimilation approach defined above.

With the EnKF method, Raven runs N simulations over the specified model duration, where N is the size of the EnKF ensemble. After running the model N times, two sets of ensemble solution files are potentially produced: the standard (un-assimilated) model states, and the EnKF-adjusted model states, where the observational data is used to adjust model states such as streamflow or SWE. The EnKF ensemble can be run in five modes:

1. Spinup mode (`:EnKFMode ENKF_SPINUP`), where each ensemble member is run using a single default initial conditions file and with perturbed forcings. Produces an EnKF-adjusted solution file for each ensemble member at the end of simulation that can be used to initialize further simulations.
2. Closed-loop mode (`:EnKFMode ENKF_CLOSED_LOOP`), where each ensemble member is run using the observation-adjusted state file (solution_EnKF.rvc) from the previous run of that member as initial conditions and with perturbed forcings. Produces an EnKF-updated solution file at the end of the simulation run.
3. Forecast mode (`:EnKFMode ENKF_FORECAST`), where each ensemble member is run using the observation-adjusted state file (solution_EnKF.rvc) from the previous run of that member as initial conditions, except without perturbing model forcings. Forecast mode is typically run for the future, when observations are unavailable.
4. Open-loop mode (`:EnKFMode ENKF_OPEN_LOOP`), where each ensemble member is run using the un-adjusted state file from the previous run of that member as initial conditions and with perturbed forcings.
5. Open-loop Forecast mode (`:EnKFMode ENKF_OPEN_FORECAST`), where each ensemble member is run using the un-adjusted state file (solution.rvc) from the previous run of that member as initial conditions, except without perturbing model forcings.

The open-loop modes are strictly used for evaluating the effectiveness of the EnKF assimilation strategy; they are not typically used in operations. For all simulations, each ensemble member is generated by perturbing some subset of forcing functions (usually at least precipitation) prior to t_0 (spinup or closed-loop mode) and using unperturbed forcings after t_0, where t_0 is the start of the forecast (forecast mode). Open loop mode is typically only used as a reference, to evaluate the impact of assimilation. This process is shown in figure 6.1.

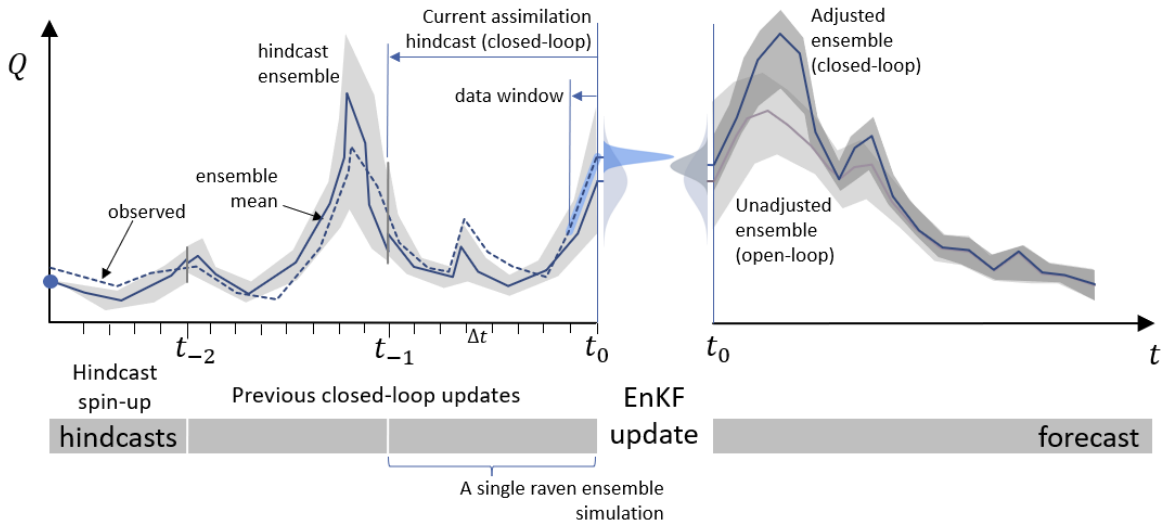

Figure 6.1: EnKF data assimilation process - here EnKF state updates are appiled every eight time steps using the most recent observation data, and t_0 corresponds to the current time such that after t_0 is a future forecast. Raven only applies EnKF data assimilation at a single time for each ensemble run, but these assimilation runs may be strung together in sequence. In this figure, t_{-1} and t_{-2} correspond to the model :StartDate of previous closed-loop forecasts; t_0 corresponds to the start of the forecast run. For standard EnKF, the data window only includes the single model time step prior to t_0.

Specified components of the model state (e.g., streamflow, soil moisture, and/or snow) are updated using perturbed observational data (e.g., hydrographs). This is effectively a computation whereby the *a priori* estimates of system state (represented by an ensemble) is corrected using noisy observational data (also represented by an ensemble) to generate an *a posteriori* estimate of the system state. This updated state is used as initial conditions for an ensemble of deterministic forecasts. The full ensemble of forecasts may be used, or only the ensemble mean (which has to be calculated externally of Raven).

The state updating done at the end of each ensemble run follows the EnKF approach outlined in Mandel (2006). The primary input components are:

- O_{sim}, an N_{obs} x N matrix of simulated output, with one simulation value for every corresponding observation data point used in assimilation.
- O, an N_{obs} x N matrix of perturbed observational data points, where each ensemble member varies around the actual observation based upon the chosen observational error model.
- E, an N_{obs} x N matrix of the error applied to each observational point (the difference between O and the original observations), and
- X, the state matrix. An M x N matrix of the system states at time t_0, where M is the total number of state variables to be updated during assimilation.

Raven assembles these matrices internally. At the end of each spinup or closed loop simulation, the system state is updated, and written to the solution_EnKF.rvc solution file. This update is done via the following matrix operations, detailed in Mandel (2006) using a different notation:

$$\begin{aligned} \mathrm{H} &= \mathrm{O}_{sim} - \overline{\mathrm{O}}_{sim} \\ \mathrm{A} &= \mathrm{X} - \overline{\mathrm{X}} \\ \mathrm{P} &= \tfrac{1}{N-1}(\mathrm{H}\mathrm{H}^T + \mathrm{E}\mathrm{E}^T) \\ \mathrm{Z} &= \mathrm{H}^T(\mathrm{P}^{-1}(\mathrm{O} - \mathrm{O}_{sim})) \\ \mathrm{X}^* &= \mathrm{X} + \tfrac{1}{N-1}\mathrm{A}\mathrm{Z} \end{aligned}$$

where the overbar notation indicates averaging over the N ensemble members and H, A, P, and Z are only used in these intermediate calculations. The inversion of the P matrix is internally calculated via singular value decomposition algorithm of Press et al. (1992), and X^* is the updated system state used as initial conditions for the forecast ensemble members.

Raven should be run in three phases for practical application of EnKF forecasting, by stringing together an initial spinup run, a sequence of closed-loop runs, and a forecast run. Only the EnKF mode, start time, end time, and external forcings such as temperature and precipitation will typically vary between these successive ensemble model runs. All EnKF options (number of members, data horizon, assimilated states, perturbation methods) will typically be fixed for a given watershed.

At steady state, daily simulations will typically include one closed loop run for the previous day followed by one forecast run for some future time horizon (e.g., the 10-day forecast). Testing of the EnKF can be done by comparing results against an open-loop run, replacing the `ENKF_CLOSED_LOOP` and `ENKF_FORECAST` modes above with `ENKF_OPEN_LOOP` and `ENKF_OPEN_FORECAST` options. These options will disable the Kalman Filter application so that states are not adjusted to agree with observations. The model is otherwise identical, including the randomized perturbations of the forcings.

The key choices used in EnKF data assimilation are 1) the number of ensemble members to use, 2) the specification of which observations to include and the observational error model, 3) the states to include in updating, and 4) the strategy for perturbing the forcings. The number of ensemble members is specified in the `:EnsembleMode ENSEMBLE_ENKF` command in the .rvi file. The observations are

included using the :AssimilateStreamflow command and the observational error model is supplied using the :ObservationErrorModel command (both in the .rve file). The forcing perturbation approach (which can be applied differently in each subbasin group) is specified using one or more :ForcingPertubation commands in the .rve file. These are all documented in appendix A.8, with an example EnKF template setup included in appendix F.14.

By default, the file management for EnKF simulation is such that all individual ensemble outputs are generated in a set of N output folders defined by the :OutputDirectoryFormat command in the .rve file. Each simulation generates two output solution files in this folder: solution.rvc and solution_EnKF.rvc. When run in succession, EnKF simulations will by default read one of these solution files as initial conditions (either with or without assimilation, as determined by the open/closed EnKF mode). These files are overwritten at the end of their simulations. If a forecasting system wishes to save these states, it must therefore intervene in between successive EnKF runs. If these solution files are saved, they may be used to re-boot EnKF simulations at any previous time by overriding the default assumed location of the initial conditions using the :EnsembleRVCFormat command documented in appendix A.8.

Currently, Raven only supports the use of streamflow observations or lake/reservoir stage for EnKF assimilation.

6.5 Deltares FEWS support

Raven readily plugs into the Deltares Delft-FEWS forecasting environment https://www.deltares.nl/en/software/flood-forecasting-system-delft-fews-2/, communicating directly to FEWS via custom NetCDF input and output files. Instructions for getting Raven to directly communicate with FEWS are found in appendix C. FEWS provides a number of external tools for ensemble model simulation, state updating, and data assimilation.

Chapter 7

Tracer, Heat, and Contaminant Transport

Raven can be used to track conservative contaminants, enthalpy (i.e., temperature),and/or synthetic tracers (here collectively referred to as constituents) through a watershed via advection. Transport is activated using the `:Transport` command in the .rvi file (see appendix A.1.6, and the user specifies the name of the constituent. For results to be interesting, boundary and/or initial conditions must also be supplied. For special constituents like enthalpy (temperature simulation) and isotopes, transport automatically activates other processes (in-reach energy balance calculation and isotopic enrichment processes, respectively). For all user-specified constituents, the constituents are assumed to be conservative tracers or subject to simple first-order decay.

7.1 Advective Transport

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 (m, 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 density [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 [mg/mm-m^2] 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 advection of heat, the principles are the same, but the units of [mg] are replaced with [MJ], and instead of mass concentrations, the specific (volumetric) enthalpy is tracked. This is converted to (and reported as) an equivalent temperature in the model output files.

The base outputs from the transport simulation are the average concentrations of a given constituent in each of the various storage compartments and pollutographs (time series of in-stream concentration) at subbasin outlets. The `:CustomOutput` command may be used to report local concentrations and mass fluxes.

7.2 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 Robin conditions, where a user-specified (dry) mass flux is applied to a given compartment

These are handled using the `:FixedConcentration` and `:MassFlux` commands (for time-invariant concentrations or fluxes) and using the `:FixedConcentrationTimeSeries` or `:MassFluxTimeSeries` commands for time-variable fluxes. These source terms may be applied selectively to arbitrary groups of HRUs.

7.3 Constituent Routing

7.3.1 Catchment Routing

Constituents are routed through the catchment in a manner consistent with the catchment routing process described in section 4.1. 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} [mg/d] 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 4.1. This expression is identical for synthetic tracers and heat transport, but calculated in units of [L/d] and [MJ/d], respectively.

7.3.2 In-channel Routing

Raven currently supports in-channel routing of transport constituents only with the `ROUTE_DIFFUSIVE_WAVE`, `ROUTE_PLUG_FLOW`, and `ROUTE_NONE` methods of in-channel routing. For these methods, a discrete transfer function approach is used similar to that used in the in-catchment routing,

$$QC(t+\Delta t) = \sum_{n=0}^{N} QC_{in}(t - n\Delta t) \cdot UH'_n \tag{7.2}$$

where QC [mg/d] is the mass loading, QC_{in} [mg/d] is the loading from upstream at time t, and $\vec{UH'}$ is a unitless vector of size N which describes the distribution of arrival times to the channel outlet, and is the same distribution used by the in-channel routing for water, described in section 4.2. Again, This expression is identical for synthetic tracers and heat transport, but calculated in units of [L/d] and [MJ/d], respectively.

7.3.3 In-reservoir Routing

Constituent routing in the reservoir is based upon an explicit solution of the Crank-Nicolson discretized mass balance on the reservoir,

$$\frac{dM}{dt} = \sum_{i=1}^{N} Q_{in}^{i} C_{in}^{i} - Q_{out} C - \lambda M \tag{7.3}$$

where M is the reservoir mass (in mg), Q_{in}^{i} and Q_{out} are the N inflows and single outflow rates from the reservoir (in m^3/s), C_{in}^{i} are the concentrations [mg/m^3] from the multiple inflows, and λ is the decay rate of the constituent [1/d]. Note that evaporation is presumed not to carry the constituent from the reservoir surface. The discrete form of the equation, after summing all of the mass inflow terms together into a single effective mass inflow, $Q_{in}C_{in} = \sum_{i=1}^{N} Q_{in}^{i} C_{in}^{i}$, is:

$$\frac{M^{n+1} - M^{n}}{\Delta t} = \frac{1}{2}\left(Q_{in}^{n} C_{in}^{n} + Q_{in}^{n+1} C_{in}^{n+1}\right) + \frac{1}{2}\left(Q_{out}^{n} C^{n} + Q_{out}^{n+1} C^{n+1}\right) - \frac{\lambda}{2}\left(C^{n} + C^{n+1}\right) \tag{7.4}$$

where n indicates the time step, and the concentration C^n is evaluated as $M^n/V(h^n)$ where $V(h)$ is the volume of the reservoir for a stage of h. This expression may be directly rearranged to determine the mass (or energy, or composition) in the reservoir at the end of the time step, M^{n+1}.

7.4 Synthetic Tracers

Raven can support the simultaneous use of multiple synthetic tracers to calculate the percentage of streamflow sourced from different regions of the watershed (e.g., from headwater basins) or different physical stores on the landscape (e.g., to determine the percentage of flow sourced from snowmelt). This can be done by defining a tracer with a fixed concentration of 1.0 [mm/mm] in the store of interest (e.g., snowpack), and initial conditions of zero concentration everywhere else in the model (the default). This simplest form of tracer can be included with just two lines in the Raven input .rvi file. For example, we can tag all snow with a tracer using the following:

```
:Transport SNOWMELT
:FixedConcentration SNOWMELT 1.0 SNOW
```

In the case of a synthetic 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 [mm/mm] can be interpreted as the fraction of storage which is marked by tracer. Synthetic tracer concentrations should range from 0 to 1 [mm/mm], i.e., mm of water from a given source per mm of total water.

7.5 Thermal Transport

Raven supports simulation of advection of heat on the landscape and treats the full energy balance in each non-headwater river reach. This is enabled by specifying a constituent named `TEMPERATURE` in the `:Transport` command. Instead of using temperature as the state variable of interest, volumetric

enthalpy of the water h_w, in [MJ/m^3] is simulated, defined from the temperature, T, and percent frozen, p_f, as:

$$h_w(T,p_f) = \begin{cases} c_w \rho_w T & T > 0 \\ -\lambda_f \rho_w p_f & T = 0; 0 \leq p_f \leq 1.0 \\ h_t + c_i \rho_w T & T < 0 \end{cases} \tag{7.5}$$

where c_w and c_i [MJ/kg/K] are the specific heat capacities of liquid water and ice, respectively, ρ_w [kg/m^3] is the density of water, λ_f [MJ/kg] is the latent heat of fusion of water, and $h_t = -\lambda_f \rho_w$ [MJ/m^3] is the transition enthalpy at which water fully freezes. Simulating volumetric enthalpy instead of temperature allows Raven to support simulation of freezing with fewer numerical artefacts.

In each non-headwater reach, the following energy balance is solved using a Lagrangian approach for each parcel of water as it moves from inlet to outlet:

$$V\frac{dh_w}{dt} = \underbrace{k^* A_s (T_{air} - T)}_{\text{convection}} + \underbrace{R^{inc} A_s - R^{og}_{LW} A_s}_{\text{radiation}} - \underbrace{\text{ET} \rho_w \lambda_v A_s}_{\text{latent heat}} + \underbrace{Q_f A_s}_{\text{frict.}} + \underbrace{q_{mix} A_b c_w \rho_w (T_{GW} - T)}_{\text{GW mixing}} + \underbrace{2\frac{k_{bed}}{d_{bed}} A_b (T_{bed} - T}_{\text{bed conduction}} \tag{7.6}$$

where V is the volume of the parcel, h_w [MJ/m^3] is the volumetric enthalpy of the parcel, k^* [MJ/m^2/d/K] is the convective exchange coefficient of the reach (subbasin parameter `CONVECT_COEFF`), A_s [m^2] is the reach surface area, T_{air} is the air temperature [$^\circ$C], T is the water temperature [$^\circ$C], R_{inc} is the incoming radiation on the stream surface, R^{og}_{LW} is the outgoing longwave energy from the stream surface, ET is the evaporation from the stream surface, ρ_w is the density of water, λ_v is the latent heat of volatilization, Q_f is the heat gained through friction with the bed, q_{mix} [m/d] is the hyporheic exchange flux with the groundwater (subbasin parameter `HYPORHEIC_FLUX`, k_{bed} [MJ/m/d/K] is the riverbed thermal conductivity, d_{bed} [m] is the riverbed material thickness, A_b [m^2] is the wetted bed area, c_w is the specific heat of water, and T_{GW} [$^\circ$C] is the groundwater temperature. The outgoing longwave term is linearized using a first order Taylor series.

The details of this semi-analytical approach will be found in a forthcoming paper; because the energy balance is solved analytically, there are no time stepping or discretization constraints, nor are there stability issues; the numerical method is unconditionally stable. Because this problem is only solved in each reach, and the temperature from headwater basins is solely estimated from advective mixing alone, care must be taken in interpreting simulation results in headwaters.

A similar Lagrangian energy balance is applied for water during its transit between being released from the landscape and reaching the channel, i.e., for water as it is being processed through in-catchment routing. Because the pathway of this water is not explicitly represented and can represent a combination of overland flow or subsurface pathways, Raven provides two empirical parameters that enable the water to either exchange sensible heat with the atmosphere or mixing withe the groundwater. The energy balance for parcels of water on their way to the channel is:

$$\frac{dT}{dt} = k_1 (T_{air} - T) + k_2 (T_{GW} - T) \tag{7.7}$$

where k_1 [1/d] and k_2 [1/d] are the subbasin parameters `SENS_EXCH_COEFF` and `GW_EXCH_COEFF`, respectively. These parameters would typically be used within calibration. Increased values of k_1 will force the runoff temperature to equilibrate the air temperature, while increased values of k_2 will force the runoff temperature to equilibrate with the groundwater store temperature, as specified using the `:HyporheicLayer` command.

Stream temperature simulation requires the following modification to the input files:

1. The following command sequence should be added to the .rvi file:

```
:Transport TEMPERATURE
:FixedConcentration TEMPERATURE ATMOS_PRECIP -9999
:FixedConcentration TEMPERATURE SNOW         0.0
:FixedConcentration TEMPERATURE CANOPY_SNOW  0.0
```

2. Specific WATER HRUs must be added to the HRU list - one in each non-headwater subbasin. These HRUs are used for determining the air temperature, evaporation rate, and incoming radiation at the stream reach surface. Each HRU is 'tied' to the corresponding subbasin by specifying the `REACH_HRU_ID` parameter for each subbasin within a `:SubBasinProperties` command block.

3. The exchange is controlled by the key subbasin parameters `CONVECT_COEFF` and `HYPORHEIC_FLUX` (also specified in the `:SubBasinProperties` command block), and the parameters which influence the ET rate and incoming radiation on the stream surface (dependent upon the chosen algorithms for shortwave canopy corrections and PET). It may also be useful to fix the groundwater local temperature using the `:FixedTemperature` command. The subbasin parameters `SENS_EXCH_COEFF` and `GW_EXCH_COEFF` may also be set to non-zero values to modify the runoff temperature. Lastly, the subbasin parameters `RIVERBED_CONDUCTIVITY` and `RIVERBED_THICKNES` control conductive heat exchange with the bed materials.

4. The fidelity and quality of the stream temperature simulations is strongly tied to the level of discretization of the stream network. A model which provides adequate flow predictions may generally require a finer resolution of subbasins to adequately simulate stream temperature.

5. Temperature models are very sensitive to diurnal variation, and will perform better when run using model time steps of 4 hours or less.

Lakes and reservoirs are also simulated as part of the stream temperature model. Each lake is treated as fully-mixed (i.e., no thermocline is allowed to form), and exchanges heat via atmospheric radiative, sensible, and latent heat exchange plus conduction with the lake bed. The model supports partial freezing of the lake, but this freezing is assumed not to impact the lake outflow hydraulics. The model solves the following coupled energy balance equations for the lake and bottom sediment, where these equations are solved using the Newton-Raphson method as applied to a Crank-Nicolson approximation of these equations in each time step:

$$\begin{aligned}\frac{dH_w}{dt} &= Q_{in}h_{in} - \frac{Q_{out}}{V}H_w + k^*(T_{air}-T)A + R^{net}A - \text{ET}\rho_w\lambda_v A + \frac{k_{bed}2A_bc_w\rho_w}{D_{bed}}(T_b - T)\\ \frac{dH_s}{dt} &= -\frac{k_{bed}2A_bc_w\rho_w}{D_{bed}}(T_b - T)\end{aligned}$$

where the parameters are defined as in equation 7.7 except k^* [MJ/m^2/K/d] is the lake convection coefficient (reservoir parameter specified using the `:LakeConvectionCoeff` command), k_{bed} [MJ/m/K/d] is the thermal conductivity of the lake bed material (reservoir parameter `:LakebedThermalConductivity`, and D_{bed} [m] is the thickness of the lake bed (reservoir parameter `:LakebedThickness`). The variables H_w and H_s refer to the enthalpy of the lake water and lake bed material (including water), respectively, and A is the (time-varying) lake surface area. The bed interface area A_b is treated as constant and equal to the corresponding HRU area. h_{in} is the volumetric enthalpy of the incoming water.

Raven automatically generates the file StreamReachEnergyBalances.csv if `TEMPERATURE` transport is enabled. If there are also "gauged" subbasins with reservoirs/lakes, then Raven will generate the output file LakeEnergyBalances.csv which reports upon the detailed energy balance components of these lakes over time. Neither NetCDF nor Ensim output format is supported for these outputs.

7.6 Geochemistry

Geochemical processes operate similarly to hydrologic processes and determine the decay, consumption, and transformation of transported constituents. The specification of geochemical processes must be after the `:Transport` command is used to declare which constituents are to be tracked in the model. These appear in the `:GeochemicalProcesses` block as follows:

```
:Transport STRONTIUM
:Transport STRONTIUM_S SORBED
:GeochemicalProcesses
  :Decay        DECAY_LINEAR   RAD_DECAY STRONTIUM
  :Equilibrium  EQ_FIXED_RATIO SORPTION  STRONTIUM STRONTIUM_S
:EndGeochemicalProcesses
```

Much like the `:HydrologicalProcesses` commands, each geochemical process consists of a process (e.g., `:Decay`), an algorithm to represent that process (e.g., `DECAY_LINEAR`), and one or more constituents (`STRONTIUM`). Unlike the hydrological processes, here a user-specified 'process name' is also linked to each geochemical process (e.g., `RADIOACTIVE_DECAY`). This is because the same constituents may experience multiple similar processes in the same storage compartment (e.g., consumption /loss via two different linear mechanisms); the process name helps to distinguish between these and also helps to track the impact of these processes in the output.

Note that all of the calculations below assume units are in terms of constituent mass, not molar concentrations. Therefore handling of stoichiometry and units conversion is (in part) left to the user.

7.6.1 Decay

Any constituent may be subjected to first order linear decay/consumption/loss in any water compartment, with the decay rate being 1) uniform everywhere in the domain 2) conditional upon compartment or 3) conditional on soil class. This simple decay relation (in terms of aqueous concentration of the constituent, C) is given as:

$$\frac{dC}{dt} = -kC \tag{7.8}$$

In all cases, the mean decay over the time step is numerically calculated using the following discrete expression over the time step Δt:

$$\frac{dm}{dt} = m \cdot \frac{(1 - \exp(-k\Delta t))}{\Delta t}; \tag{7.9}$$

where k is the decay rate [1/d] and m [mg/m^2] is the constituent mass density. This is derived from the analytical solution to Equation 7.8 as evaluated over a single time step. All mass lost to decay is tracked as moving to the `CONSTITUENT_SINK` storage compartment for mass balance accounting. These processes are handled using a `:Decay` command in the `:GeochemicalProcesses` command block.

Simple linear decay (`DECAY_LINEAR`)

The linear decay rate, k [1/d] in 7.9 is a constant parameter defined by the geochemical parameter `DECAY_COEFF` for the target compartment or soil class and the specified constituent.

Denitrification(DECAY_DENITRIF)

The linear decay rate, k [1/d] in 7.9 is a denitrification rate which varies with temperature and soil saturation:

$$k = k' \cdot C(T) \cdot \left(\frac{\phi_m}{\phi_m^{max}} \right)$$

where k' [1/d] is the geochemical parameter DECAY_COEFF for the target compartment or soil class, ϕ_m [mm] and ϕ_m^{max} are the soil water storage and maximum soil capacity of soil layer m, and $C(T)$ is a function of temperature, equal to 1 for temperatures between 10 °Cand 30 °C, and linearly varying to zero as the temperatures approach 5 °Cand 50 °C, respectively. This is only intended to be used if the constituent is (some variation of) NITRATE.

7.6.2 Equilibrium

A process whereby two constituents approach (or are assumed to always preserve) an equilibrium balance. Handled using a :Equilibrium command in the :GeochemicalProcesses command block.

Fixed mass ratio equilibrium (EQUIL_FIXED_RATIO)

Assumes the concentrations of two aqueous constituents are always at a fixed ratio:

$$C_1 = aC_2$$

If this ratio is not preserved, a precise amount of C_1 is converted to C_2 or vice versa in order to maintain this equilibrium. The coefficient a is the geochem parameter PAR_EQFIXED_RATIO, which may be global or specified as specific for a given water compartment. Note that because Raven does not simultaneously solve chemical equations, this equilibrium is only approximately preserved if competing with other chemical reactions or additions of mass. Note - this approach can be used to represent chemical equilibrium. For example, if you have the following reaction:

$$C_1 \leftrightharpoons C_2$$

with unitless equilibrium coefficient K_{eq}, we get

$$\frac{[C_1]}{[C_2]} = K_{eq}$$

where square brackets denote molar concentrations. Given the molecular weights γ_1 and γ_2 (g/mol) of the two species, we can calculate the fixed ratio a as:

$$a = \frac{\gamma_1}{\gamma_2} K_{eq}$$

Fixed mass ratio equilibrium (EQUIL_FIXED_RATIO)

Assumes a rate-limited reaction whereby concentrations of two aqueous constituents approach a fixed ratio.

$$C_1 \leftrightharpoons aC_2$$

$$\begin{aligned} \frac{dC_1}{dt} &= -k\left(C_1 - aC_2\right) \\ \frac{dC_2}{dt} &= +ks\left(C_1 - aC_2\right) \end{aligned}$$

where the coefficient a [-] is the geochemical parameter `PAR_EQFIXED_RATIO` and k [1/d] is the geochemical paramter `PAR_EQUIL_COEFF`. Both may be global or specified as specific for a given water compartment. The parameter s is the stoichiometric ratio between the two components relating the mass lost of constituent 1 to the mass gained by constituent 2, and is specified using the `:StoichiometricRatio` command in the .rvp file.

Linear Equilibrium Sorption (`EQUIL_LINEAR_SORPTION`)

Assumes an instantaneous reaction whereby the aqueous and sorbed concentrations of a constituent species are linearly related:

$$C_s = K_d C_w$$

where C_w [mg/l] is the aqueous concentration, C_s [mg/kg] is the sorbed concentration, and K_d is the sorption coefficient (geochemical parameter `PAR_SORPT_COEFF`). In terms of mass density, this relationship translates to:

$$m_s = \left(\frac{K_d \rho_b H n}{V_w} \right) \cdot m_w$$

where m_s and m_w are the sorbed and aqueous mass densities, ρ_b [kg/m^3] is the bulk density of the soil (soil parameter `BULK_DENSITY`), H is the soil layer thickness, n is the soil porosity (soil parameter `POROSITY`), and V_w [m] is the water content of the soil. This process is only enabled within soil storage compartments. Note that because Raven does not simultaneously solve chemical equations, this equilibrium is only approximately preserved if competing with other chemical reactions or additions of mass.

7.6.3 Chemical Transformation

A process whereby one constituents transforms into another at a rate proportional to the concentration of one or more species, and potentially influenced by other controls (such as temperature). Handled using a `:Transformation` command in the `:GeochemicalProcesses` command block.

Linear transformation (`TRANS_LINEAR`)

A single species transforming to another at a rate proportional to the concentration of the first species:

$$C_1 \rightarrow aC_2$$

This is treated internally as a set of rate reactions:

$$\begin{aligned} \frac{dC_1}{dt} &= -kC_1 \\ \frac{dC_2}{dt} &= +ksC_1 \end{aligned}$$

where k [1/d] is the rate of reaction, the geochemical parameter `PAR_TRANSFORM_COEFF`, which may be global or specified as specific for a given water compartment. The parameter s is the stoichiometric ratio between the two components relating the mass lost of constituent 1 to the mass gained by constituent 2, and is specified using the `:StoichiometricRatio` command in the .rvp file.

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 simulation output. The model diagnostic output can readily be used by model-independent optimization and parameter estimation tools, such as Ostrich or PEST (as briefly discussed in section 2.6). 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 simulated 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`, `:IrregularObservations`, `:IrregularWeights` and `:EvaluationMetrics`) can be found in appendix A. The `:ObservationData` command is used for continuous observations at regular intervals (e.g., hydrographs) while the `:IrregularObservations` is used to store infrequent observations with irregular spacing (e.g., those from snow surveys).

8.2 Diagnostic Algorithms

In all of the algorithms below, ϕ_i is an observation of interest, $\hat{\phi}_i$ is the corresponding simulated 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). By default, Raven calculates the diagnostics for the entire simulation period, however, the diagnostics for any sub-period of the simulation may be evaluated using the `:EvaluationPeriod` command documented in appendix A.1.3. Alternately, the evaluation period command can also limit diagnostic calculation to low or high flows only (e.g., the top 20% of observed flows). Specification of the time series of observation weights is done via the `:ObservationWeights` or `:IrregularWeights`

commands, as documented in appendix A.4.2. Blank observation data (denoted by the flag value -1.2345 in any time series) is ignored, and treated as if it has a weight of zero.

Nash-Sutcliffe Efficiency (`NASH_SUTCLIFFE`)

Returns the Nash-Sutcliffe efficiency coefficient:

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

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

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

Running Average Nash-Sutcliffe Efficiency (`NASH_SUTCLIFFE_RUN`)

Returns Nash-Sutcliffe efficiency coefficient between the M-day running average of the observed and the M-day running average of the simulated variables:

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

where $\hat{\phi}_i^*$ is the M-day running average of the simulated value (averaging window centred on day i), and ϕ_i^* is the M-day running average of the observed value (centred on day i), and $\bar{\phi}$ is the weighted mean of observations,

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

The window size, M, in days, is specified in the `NASH_SUTCLIFFE_RUN` command.

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

Returns the Nash-Sutcliffe efficiency metric applied to the logarithmic transform of the variable of interest, usually discharge.

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

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

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

Fourth-power Nash-Sutcliffe Efficiency (NSE4)

$$\text{NSE4} = 1 - \frac{\sum_{i=1}^{N} w_i \left(\hat{\phi}_i - \phi_i\right)^4}{\sum_{i=1}^{N} w_i \left(\bar{\phi} - \phi_i\right)^4}$$

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

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

Daily Nash-Sutcliffe Efficiency (`DAILY_NSE`)

Nash-Sutcliffe efficiency metric calculated using daily average values of simulated and observed flows. Calculations begin with the first full day of values.

Fuzzy Nash-Sutcliffe Efficiency (`FUZZY_NASH`)

Nash-Sutcliffe efficiency metric calculated in such a way that only deviations beyond a fixed fraction, ω, of the observed value are penalized.

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

where $\epsilon(\hat{\phi}_i, \phi_i)$ is defined as:

$$\epsilon(\hat{\phi}_i, \phi_i) = \begin{cases} \hat{\phi}_i - (1+\omega)\phi_i & \text{if } \hat{\phi}_i > (1+\omega)\phi_i \\ \hat{\phi}_i - (1-\omega)\phi_i & \text{if } \hat{\phi}_i < (1-\omega)\phi_i \\ 0 & \text{otherwise} \end{cases}$$

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

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

Root-mean-squared Error (`RMSE`)

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

Fourth-root-mean-quadrupled Error (`R4MS4E`)

$$\text{R4MS4E} = \sqrt[4]{\sum_{i=1}^{N} w_i \left(\hat{\phi}_i - \phi_i\right)^4}$$

Square-root-transformed mean-squared Error (`RTRMSE`)

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

Kling Gupta Efficiency (`KLING_GUPTA`)

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

$$\text{KGE} = 1 - \sqrt{(r-1)^2 + \left(\frac{\hat{\sigma}}{\sigma} - 1\right)^2 + \left(\frac{\hat{\mu}}{\mu} - 1\right)^2}$$

where r is the Pearson correlation coefficient, σ and $\hat{\sigma}$ are the standard deviation of the observed and simulated data, respectively, and μ and $\hat{\mu}$ are the means of the observed and simulated data, respectively.

Kling Gupta Efficiency Variant (`KGE_PRIME`)

Returns the Kling-Gupta efficiency metric as defined above except using the ratios of coefficient of variation (σ/μ) rather than the ratios of standard deviations.

Daily Kling Gupta Efficiency (`DAILY_KGE`)

Returns the Kling-Gupta efficiency metric for daily observations as defined in Gupta et al. (2009), regardless of simulation time step.

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.

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

Absolute Percentage Bias (`ABS_PCT_BIAS`)

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

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

Average Absolute Error (`ABSERR`)

Returns the weighted average absolute error.

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

Average Absolute Error - Running average (`ABSERR_RUN`)

Returns the weighted average absolute error applied to a moving window average of the simulated and observed time series.

Maximum Absolute Error (`ABSMAX`)

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

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

Relative Absolute Error (`RABSERR`)

Returns the weighted relative absolute error.

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

Spearman Rank Correlation Coefficient (`SPEARMAN`)

Returns the unweighted Spearman rank correlation coefficient

$$\text{SPEARMAN} = \frac{\text{cov}(R(\hat{\phi}), R(\phi))}{\sigma_{R(\hat{\phi})}\sigma_{R(\phi)}}$$

where $R(X)$ is a vector comprised of the rank of the ordered arguments X, $\sigma_{R(X)}$ is the standard deviation of these ranks, and cov() denotes the covariance of two vector inputs. With the Spearman coefficient calculation, observation weights are ignored unless they are zero, in which case the zero-weight data point is excluded from the calculation.

Peak difference (`PDIFF`)

Returns the difference between the peak simulated 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.

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

Percent peak difference (`PCT_PDIFF`)

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

$$\text{PCT\_PDIFF} = \left\{\max\left\{\hat{\phi}_i\right\} - \max\left\{\phi_i\right\}\right\} / \left\{\max\left\{\phi_i\right\}\right\}$$

Absolute percent peak difference (`ABS_PCT_PDIFF`)

The difference between the peak simulated data and peak observed data, normalized by the peak observed data and taken as an absolute value. Non-zero weights have no effect on this calculation, but zero weights will force the corresponding data points to be ignored.

$$\text{ABS\_PCT\_PDIFF} = \left|\left\{\max\left\{\hat{\phi}_i\right\} - \max\left\{\phi_i\right\}\right\} / \left\{\max\left\{\phi_i\right\}\right\}\right|$$

Monthly Mean Squared Error (`TMVOL`)

Describes the total monthly mean error between simulated 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.

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 simulated data. N^* is the number of adjacent non-blank data entries. Non-zero observation weights are ignored.

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 simulated values are constantly above or below the observed values.

Persistence Index (`PERSINDEX`)

The Persistence Index, defined in Bennett et al. (2013), compares the sum of squared error to the error that would occur if the previous observation value (ϕ_{i-1}) was used to predict the next value (ϕ_i). Similar to NSE, this metric is equal 1 if the model is perfect, and is less than zero if the model predicts more poorly than a hydrograph shifted by one time step.

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

Chapter 9

Raven Code Organization

*This section is intended primarily for Raven software developers

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 and PDEs from the evaluation of flux-storage relationships, enabling the testing of various numerical schemes without having to dig into each subroutine for each hydrologic process.

9.1 Classes

The Class diagram for the Raven code is depicted in figure 9.1. 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.

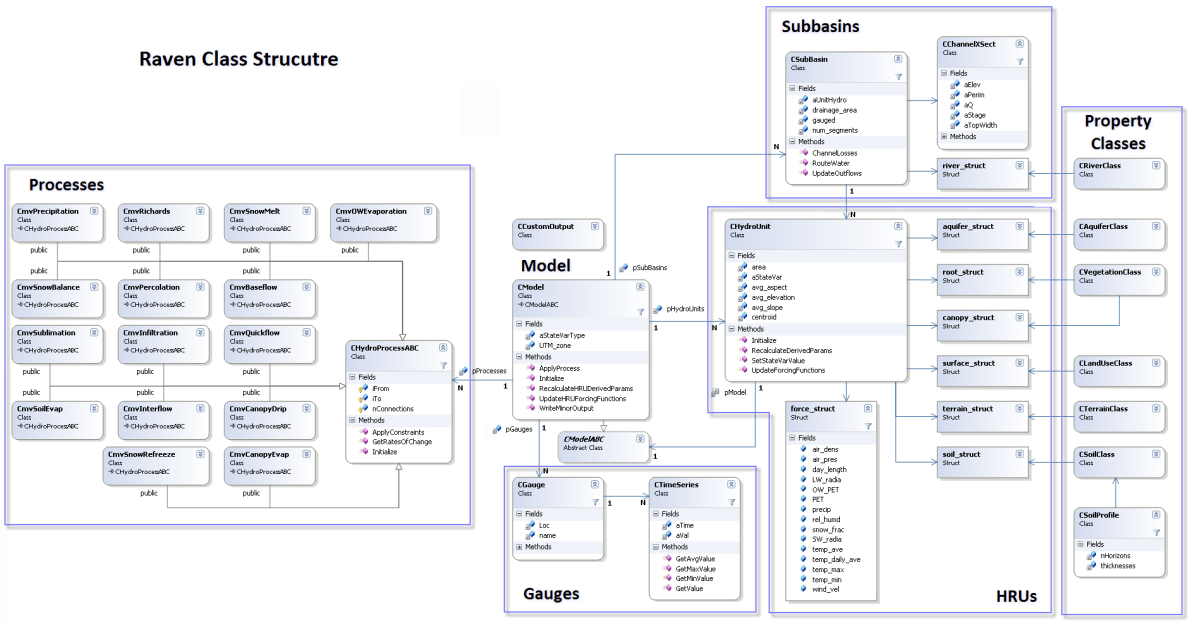

Figure 9.1: Raven class diagram

9.1.1 CModel class

The `CModel` class is a container class for all of the hydrologic 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.
- `IncrementBalance()`
- `IncrementLatBalance()`
- `IncrementCumulInput()`
- `IncrementCumOutflow()` 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()` Called every time step - 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 time step.
- `RecalculateHRUDerivedParams()`, `UpdateTransientParams()` called every time step - updates derived and specified model parameters which change over time.
- `ApplyProcess()`, `ApplyLateralProcess()` 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 $\{\phi\}$
- `UpdateDiagnostics` Compares current simulated and observed output for the time step and updates diagnostic measures.

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.)

9.1.2 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.

9.1.3 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 simulated watershed(s). Conceptualized as a subbasin.

9.1.4 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 land use 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

9.1.5 CHydroProcessABC class

An abstraction of any hydrologic 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 five 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), mg/m^2/d (for constituent mass) 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 9.1.6

9.1.6 Hydrologic Processes

All hydrologic 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.

9.2 Contributing to the Raven Framework*

Source code for Raven is available online, with file support for Microsoft Visual Studio, both 2013 and 2017 versions. 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.

9.2.1 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 9.2.2.
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 9.2.3
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 simulated 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 ideally not be used except later in the ApplyConstraints routine. A good rule of thumb is that the time step should not appear anywhere in this code. This may not be strictly possible with some more complicated algorithms.
- each process 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 mm/d 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`.

9.2.2 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

9.2.3 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 class, soil class, vegetation class, or terrain class.
2. Add the new global parameter to the `global_struct` structure, non-global parameters to the corresponding `soil_`, `veg_`, `terrain_`, or `surface_struct` (corresponding to land use). 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::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_name
```

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_name
```

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. Note that comments may be included on individual lines using the # character as the first word on the line. Inline comments are also allowed - Raven will ignore all text to the right of the # symbol. An .rvi file is structured as follows:

```
# ------------------------------------------
# Raven Input (.rvi) file
# ------------------------------------------
:StartDate       2000-10-01 00:00:00
:EndDate         2001-09-30 00:00:00
:TimeStep        01:00:00
# -Options----------------------------------
:Routing         ROUTE_HYDROLOGIC
:CatchmentRoute  ROUTE_GAMMA_CONVOLUTION # inline comment
:Evaporation     PET_PENMAN_MONTEITH
:SoilModel       SOIL_TWO_LAYER
# -Processes--------------------------------
:HydrologicProcesses
  :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
:EndHydrologicProcesses
# -Custom Output----------------------------
:CustomOutput DAILY   AVERAGE SOIL[0] BY_HRU
:CustomOutput MONTHLY MAXIMUM SOIL[1] BY_BASIN
```

Note that, for the most part, input commands in Raven are unstructured - spacing, tabs, etc., should not impact the ingestion of input. Most commands can be input in arbitrary order. The key exceptions to this are

1. The `:SoilModel` command must precede the `:HydrologicProcesses` block, `:DefineHRUGroups`

command, and :LakeStorage command.

2. The :HydrologicProcesses block must precede any :Transport command.
3. If HRU groups are to be used for conditional application of processes in the :HydrologicProcesses block, for :CustomOutput, for disabling HRUs, or for transport boundary conditions, they must be declared first using the :DefineHRUGroups command.

A.1.1 Required RVI Commands

The .rvi file consists of the following required commands:

```
:StartDate [yyyy-mm-dd hh:mm:ss]
```

(Required) Starting date and hour of the simulation.

```
:EndDate [yyyy-mm-dd hh:mm:ss]
# OR
:Duration [days]
```

(Required) Ending date and hour of the simulation (:EndDate) or duration of the simulation (:Duration), in decimal days, beginning from the start date specified. Only one of these commands should be used. If both appear, the last command to appear is used.

```
:TimeStep [time step in days]
# OR
:TimeStep [hh:mm:ss]
```

(Required) Time step for the simulation, expressed in days as a real-valued number (e.g., 0.04166667 for one hour) or using hh:mm:ss format (e.g., 01:00:00 for one hour). As Raven is intended for sub-daily calculations, the time step must be less than or equal to 1.0. It also must evenly divide into a day - 2 hours or 5 minutes is allowed, but 1 hour 40 minutes is not.

```
:SoilModel [soilmodel string] {(conditional) 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

This command must be placed before the :HydrologicProcesses block.

```
:DefineHRUGroups [HRUgrp1] {HRUgrp2} ... {HRUgrpN}
```

(Somewhat required) Declaration of HRU groups that may be used for (1) conditional application of hydrologic processes, (2) grouping of custom output, (3) disabling of groups of HRUs, or (4) . They must be defined prior to use in the .rvi file. They are populated in the .rvh file using the :HRUGroup-:EndHRUGroup command. Here, the HRUgrp is a unique string identifier for the group (e.g., OpenHRUs or ForestBurnSite)

```
:HydrologicProcesses
...
:EndHydrologicProcesses
```

(Required) These commands bracket the list of hydrologic processes to be simulated (see section A.1.5)

A.1.2 Model Operational Options

The following section discusses about the several hydrologic 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 table D.4 for more details about the required parameters for each option.

```
:CatchmentRoute [method]
```

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 4.1:

- `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 [method]
```

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 4.2:

- `ROUTE_DIFFUSIVE_WAVE` (**default/recommended**) - analytical solution to the diffusive wave equation along the reach using a constant reference celerity
- `ROUTE_HYDROLOGIC` - iterative level-pool routing using channel characteristics and Manning's equation
- `ROUTE_NONE` - water is not routed from subbasin to subbasin. Intended for single-subbasin/single catchment models or numerical testing only.
- `ROUTE_STORAGE_COEFF` - 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. May require adjustment of time step to ensure convergence.

Importantly, only `ROUTE_DIFFUSIVE_WAVE`, `ROUTE_PLUG_FLOW`, and `ROUTE_NONE` routing algorithms support constituent and/or thermal transport.

```
:Method [method]
```

(Optional) Numerical method used for simulation. The method string be one of the following:

- `ORDERED_SERIES` (**default**) - Process ordering is defined as being the same as the order of hydrologic process in the input file. Many Raven methods (in particular, constituent transport) ONLY work with the ordered series approach, and it is strongly recommended that users use this option.
- `EULER` - uses the classical Euler method, with operator-splitting. Process order as specified in the input file does not matter. This option is NOT recommended, and should be used only for research.

```
: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 5.1 are supported:

- `INTERP_NEAREST_NEIGHBOR` (**default**) - the nearest neighbor (Voronoi) method
- `INTERP_INVERSE_DISTANCE` - inverse distance weighting
- `INTERP_INVERSE_DISTANCE_ELEVATION` - inverse distance weighting with consideration of elevation changes
- `INTERP_AVERAGE_ALL` - averages all specified gauge readings
- `INTERP_FROM_FILE [filename]` - weights for each gauge at each HRU are specified in a file named `filename` with the following contents:

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

  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 5.3.1, are supported:

- `RAINSNOW_DINGMAN` (**default**)
- `RAINSNOW_DATA` - gauge or gridded time series of snowfall used
- `RAINSNOW_UBC`
- `RAINSNOW_HBV`
- `RAINSNOW_HSPF`
- `RAINSNOW_HARDER`
- `RAINSNOW_WANG`
- `RAINSNOW_SNTHERM89`

```
:Evaporation [method]
```

PET calculation method for land surface. Can be one of the following methods, described in detail in section 5.4:

- `PET_HARGREAVES_1985` (**default**)
- `PET_DATA` - gauge or gridded user-specified time series used
- `PET_OUDIN` - works quite well in Canadian watersheds
- `PET_PENMAN_MONTEITH` - physically-based, but data requirements are intensive

- PET_PENMAN_COMBINATION
- PET_PRIESTLEY_TAYLOR
- PET_HARGREAVES
- PET_HAMON_1961
- PET_TURC_1961
- PET_MAKKINK_1957
- PET_PENMAN_SIMPLE33
- PET_PENMAN_SIMPLE39
- PET_FROMMONTHLY - use only for HBV emulation
- PET_MONTHLY_FACTOR - use only for UBCWM emulation
- PET_MOHYSE
- PET_VAPDEFICIT
- PET_NONE - for routing-only mode; disables evaporation
- PET_CONSTANT - constant value; not recommended in practice

Note that the evaporation algorithm will be influence by whether the :DirectEvaporation command is used.

```
:OW_Evaporation [method]
```

(Optional) PET calculation method for open water. Has the same options as :Evaporation command.

```
:DirectEvaporation
```

(Optional) If this command is added, rainfall (but not snowfall) is automatically reduced through evapotranspiration up to the limit of the calculated PET, before it has the opportunity to interact with the land surface. PET is likewise reduced by the quantity of rainfall evaporated.

```
:SnowSuppressesPET
```

(Optional) If this command is added, presence of snow suppresses PET to zero.

```
:SuppressCompetitiveET
```

(Optional) If this command is added, competitive ET is disabled and ET routines independently remove water based upon PET instead of PET reduced by simultaneous evaporative processes. By default (without this command), ET routines appropriately 'compete' to satisfy evaporative demand. This command is only recommended for emulation of existing models.

```
:AllowSoilOverfill
```

(Optional) If this command is added, soils may be filled beyond their maximum storage capacity; this excess water may then be moved using an :Overflow process. This command is generally recommended only for emulation of existing models (and is used, for instance, in the HMETS model).

```
:OroPrecipCorrect [method]
```

(Optional) Method for correcting total precipitation for orographic (elevation) effects. The following methods, discussed in detail in section 5.3.2, are supported:

- `OROCORR_NONE` (**default**)
- `OROCORR_HBV`
- `OROCORR_UBC`
- `OROCORR_UBC_2`
- `OROCORR_SIMPLELAPSE`

```
: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_SIMPLELAPSE`

```
:OroPETCorrect [method]
```

(Optional) Method for correcting estimated PET for orographic (elevation) effects. The following methods are supported, as discussed in section 5.4.2:

- `OROCORR_NONE` (**default**)
- `OROCORR_HBV`
- `OROCORR_PRMS`

Note: No specific parameters are 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 5.5, 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 [method]
```

(Optional) Means of correcting shortwave radiation to the surface due to canopy cover. The following methods, described in detail in section 5.5, are supported:

- SW_CANOPY_CORR_NONE (**default**)
- SW_CANOPY_CORR_STATIC
- SW_CANOPY_CORR_DYNAMIC
- SW_CANOPY_CORR_UBC - From Quick (2003)

```
:SWCloudCorrect [method]
```

(Optional) Means of correcting shortwave radiation to the surface due to cloud cover. The following methods, described in detail in section 5.5, are supported:

- SW_CLOUD_CORR_NONE (**default**)
- SW_CLOUD_CORR_DINGMAN
- SW_CLOUD_CORR_UBC - From Quick (2003)

```
:LWRadiationMethod [method]
```

(Optional) Means of estimating net longwave radiation. The following methods are supported, as discussed in section 5.6:

- LW_RAD_DEFAULT (**default**) - From Dingman (2002) - as of Raven v3.8, this method by default uses the LW_INC_DINGMAN incoming shortwave radiation calculation, which will be overridden by other choices of :LWIncomingMethod
- LW_RAD_DATA - gauge or gridded time series used
- LW_RAD_UBC - From Quick (2003) - ignores incoming longwave estimates
- LW_RAD_HSPF - From Bicknell et al. (1997) - ignores incoming longwave estimates
- LW_RAD_VALIANTZAS - From Valiantzas (2006) - ignores incoming longwave estimates
- LW_RAD_NONE

```
:LWIncomingMethod [method]
```

(Optional) Means of estimating incoming longwave radiation. The following methods are supported, as discussed in section 5.6:

- LW_INC_DEFAULT (**default**) - Uses LW_INC_DINGMAN (Dingman, 2002) if LW_RAD_DEFAULT is also used, otherwise incoming longwave is presumed zero, and only net longwave is calculated.
- LW_INC_DATA - gauge or gridded time series used
- LW_INC_SKYVIEW - (From Prata (1996))
- LW_INC_SICART - (From Sicart et al. (2006))
- LW_INC_DINGMAN - (From Dingman (2002))

```
:CloudCoverMethod [method]
```

(Optional) Means of calculating cloud cover percentages, if used. The following methods, as described in section 5.7, 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 5.9.1:

- `WINDVEL_CONSTANT` (**default**) - an unrealistic constant wind velocity of 3 m/s. Should be used only when processes are not strongly dependent upon wind speed.
- `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 5.9.2:

- `RELHUM_CONSTANT` (**default**) - constant relative humidity of 0.5. Should be used only when processes are not strongly dependent upon relative humidity.
- `RELHUM_DATA` - gauge or gridded time series used
- `RELHUM_MINDEWPT` - uses minimum dew point to estimate relative humidity. Recommended if observational data is unavailable.

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 5.9.3:

- `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 precipitation is intercepted by the canopy). The following methods are supported, as described in section 3.1.1:

- `PRECIP_ICEPT_USER` (**default**)
- `PRECIP_ICEPT_LAI`
- `PRECIP_ICEPT_EXPLAI`
- `PRECIP_ICEPT_NONE`

- PRECIP_ICEPT_HEDSTROM

```
:PotentialMelt [method]
```

(Optional) If used, estimates the potential melt. The following methods are supported , as discussed in section 5.8.1:

- POTMELT_DEGREE_DAY (**default**)
- POTMELT_NONE
- POTMELT_EB
- POTMELT_RESTRICTED
- POTMELT_UBCWM
- POTMELT_HBV
- POTMELT_CRHM
- POTMELT_HMETS

```
:RechargeMethod [method]
```

(Optional) Typically used for coupling with other hydrologic models that independently calculate runoff and recharge, which is handled by Raven for routing and water management.

- RECHARGE_NONE (**default**)
- RECHARGE_DATA - gridded time series used

```
:MonthlyInterpolationMethod [method]
```

(Optional) If used, performs monthly interpolations. The following methods, as discussed in section 5.11, are supported:

- MONTHINT_UNIFORM - monthly variables are treated as constant during each month
- MONTHINT_LINEAR_MID (**default**) - the monthly variables are linearly interpolated from the midpoint of each month
- MONTHINT_LINEAR_FOM - the monthly variables are linearly interpolated from the 1st day of each month
- MONTHINT_LINEAR_21 - the monthly variables are linearly interpolated from the 21st day of each month (yes, this seems very specific)

Note: No specific parameter is 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 5.10:

- SUBDAILY_NONE (**default**)
- SUBDAILY_UBC

- SUBDAILY_SIMPLE

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

```
:LakeStorage [lake storage variable]
```

Specifies the state variable to be used for rainfall on lake HRUs, typically SURFACE_WATER (default) or LAKE_STORAGE. If the lake storage state variable is used, it is critical that the user provide a hydrologic process mechanism which removes water from the lake in addition to accumulating it through precipitation. This command is typically only used for HBV emulation, which uses a non-conventional lake evaporation routine.

```
:Calendar [calendar string]
```

(Optional) Specifies the calendar to be used. By default, Raven uses the standard proleptic Gregorian calendar which includes leap years. However, for compatibility with some climate model outputs for long-term forecasting, sometimes it is useful to be able to run with (e.g.,) 365 day calendars. Raven supports the following calendars (consistent with the NetCDF calendar convention):

- PROLEPTIC_GREGORIAN (**default**)
- JULIAN - ignores special leap years (every 100/400 years)
- GREGORIAN - ignores special leap years before 1583
- STANDARD - same as GREGORIAN
- NOLEAP - leap years ignored
- 365_DAY - same as NOLEAP
- ALL_LEAP - every year gets a February 29
- 366_DAY - same as ALL_LEAP

The :Calendar command MUST be near the very top of the .rvi file before the :StartDate or :EndDate commands are indicated, because all date calculations are dependent upon the proper calendar description.

Warning: strange things can happen when mixing different calendars for forcings and simulation, and it is strongly advised that all calendars are consistently applied.

A.1.3 Optional Input/Output Control Commands

```
:RunName [name]
```

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]
```

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 default behavior. 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 [rvc_name]
:rvp_Filename [rvp_name]
:rvt_Filename [rvt_name]
:rve_Filename [rve_name]
:rvl_Filename [rvl_name]
```

Same as `:rvh_Filename [name]` above, but for .rvc,.rvp,.rve,.rvl, and .rvt files, respectively.

```
:OutputDirectory [directory name]
```

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/`.

This is equivalent to the (preferable) command line argument `-o [directory name]`. If used, this should be called as early as possible in the .rvi file. This command supports both absolute and relative path names. Recommended practice is to specify the output directory from the command line, rather than using this command.

```
:CreateRVPTemplate
```

Produces a template .rvp file in the same directory as the .rvi file based upon the hydrologic process list and model options in the .rvi file, so the user knows which parameters need to be specified for the given model configuration. NOTE: this turns off model operation, only the template file will be created. Only the .rvi file is required if this command is used. The same effect is accomplished using the command argument `-template`.

```
:OutputInterval [frequency]
```

The frequency of printing output to the output files. Default of 1 (printing every time step). Typically used for simulations with small time steps (e.g., if frequency=60 for a model with a time step of 1 minute, standard output is printed hourly).

```
:WriteMassBalanceFile
```

The file runname_WatershedMassEnergyBalance.csv (or .tb0) is generated (see appendix B)

```
:WriteForcingFunctions
```

The file runname_ForcingFunctions.csv (or .tb0) is generated (see appendix B)

```
:WriteDemandFile
```

The file runname_Demands.csv is generated (see appendix B)

```
:WriteWaterLevels
```

The file runname_WaterLevels.csv is generated (see appendix B)

```
:WriteLocalFlows
```

The file runname_Hydrographs.csv (see appendix B) is augmented with the local subbasin contribution to its outlet (i.e., the component of the outlet hydrograph sourced from the subbasin itself, in m^3/s). Only enabled subbasins listed as 'gauged' are reported.

```
:WriteSubbasinFile
```

The file runname_SubbasinProperties.csv is generated (see appendix B)

```
:WriteEnsimFormat
```

Specify that the output files generated by RAVEN should be in an EnSim (*.tb0) format instead of .csv. Used primarily for visualization with the Green Kenue software.

```
:WriteExhaustiveMB
```

The file runname_ExhaustiveMB.csv is generated (see appendix B)

```
:WriteMassLoadings
```

The file runname_MassLoadings.csv is generated (see appendix B)

```
:EndPause
```

This command forces the program output to stay on the screen (e.g., as a DOS window) until the user exits manually. Default behaviour is that the command prompt will close once execution finished.

```
:DebugMode
```

The equivalent of including `:WriteMassBalanceFile` and `:WriteForcingFunctions`. Also generates the output file debug.csv.

```
:SilentMode
```

If this command is included, output to the command prompt is minimized. Useful during automated calibration or uncertainty analysis to speed program operation.

```
:NoisyMode
```

If this command is included, output to the command prompt is maximized. Useful during debugging of program or problematic model.

```
:SuppressOutput
```

Suppresses all standard output, including generation of hydrograph, transport output, solution.rvc, custom output, and watershed storage files. Does not turn off optional outputs which were requested elsewhere in the input file. Does not turn off creation of diagnostics.csv. Useful during automated calibration to speed program operation.

```
:WaterYearStartMonth [integer month]
```

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

```
:OutputDump [YYYY-MM-DD hh:mm:ss]
```

Outputs snapshot of all state variables to file state_(timestamp).rvc, where timestamp is the indicated time in the command. The format of this file 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. This is useful for long model operations where interruption could cause work to be lost. Alternately, it can be used to generate intermediate warm start states.

```
: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 most available observation data, typically reported using time-averaged values. This is mostly used for direct comparison to emulated models, which typically do not report time step-averaged flows.

```
: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`
- `NASH_SUTCLIFFE_RUN[#1]`
- `LOG_NASH`
- `FUZZY_NASH`
- `NSE4`
- `RMSE`
- `KLING_GUPTA`
- `KGE_PRIME`

- DAILY_KGE
- PCT_BIAS
- ABSERR
- ABSERR_RUN
- ABSMAX
- PDIFF
- TMVOL
- RCOEF
- NSC
- RSR
- R2
- SPEARMAN

These metrics are defined in section 8.2, and the parameter #1 is the size of the running average window, in time steps (e.g., 7 for a weekly running average if the time step is in days). By default, these metrics are calculated for the full model duration (or whenever non-zero observation weights are present). However, the metrics may also be calculated only for specific time periods (e.g., calibration or validation periods) using the :EvaluationPeriod.

```
:EvaluationPeriod [per_name] [yyyy-mm-dd] [yyyy-mm-dd] {cond} {thresh}
# e.g.,
:EvaluationPeriod CALIBRATION 2002-10-01 2008-09-30
```

Specifies that the diagnostics specified using the :EvaluationMetrics command should (in addition to being calculated for the entire simulation) also be calculated also for sub-periods of the model simulation, usually calibration or validation periods. Each period is bounded by a start date and end date (the first and second date entries in the command). The period name is appended to the name of the diagnostic in the Diagnostics.csv output file. Users may add as many evaluation periods as needed (e.g., to generate annual fit statistics).

The optional arguments cond and thresh can be used to exclude observation data based upon a threshold percentile. cond can be one of IS_GREATER_THAN or IS_LESS_THAN and thresh is a number between 0 and 1. This conditional clause determines the frequency of observed flows in the evaluation period then retains data above or below the specified threshold percentile (expressed from 0 to 1). For instance,

```
:EvaluationPeriod CALIB_HI 2002-10-01 2008-09-30 IS_GREATER_THAN 0.2
```

Evaluates the diagnostics for the 80% highest magnitude observations during the simulation duration, and disregards the 20% smallest observations. For observed hydrographs, this may be considered retaining all flows larger than the Q20 flow.

```
:AggregateDiagnostic [agg_stat] [datatype]
e.g.,
:AggregateDiagnostic AVERAGE HYDROGRAPH
```

Reports the aggregate statistics for every diagnostic metric and every evaluation period and reports these in the Diagnostics.csv file. For instance, if the :EvaluationMetrics command contains the KLING_GUPTA diagnostic, you can use this to report the mean Kling Gupta efficiency of the simulated hydrographs across all basins. The agg_stat may be one of (MAXIMUM, MINIMUM, AVERAGE, or MEDIAN) and the datatype uses the same convention as listed in the documentation for the :ObservationData, e.g., HYDROGRAPH for discharge measurements or RESERVOIR_STAGE for lake level measurements. The aggregation does not adjust for length of data record, so these aggregate diagnostics can break down during simulations with no data coverage at one or more gauges.

```
:NetCDFAttribute [attribute name] [value]
```

Adds a user-specified attribute with name 'attribute name' and arbitrary string value 'value'. Commas are not allowed in the value string. This will typically only be used for automation of Raven within another platform.

```
:UseStopFile
```

If enabled, Raven looks for a file named 'stop' (no extension) in the model directory. If this file appears at any time during model operation, the model immediately exits after writing the solution file. This is used to interrupt long model runs, and is a safe way to prematurely terminate a run prior to completion. This should be used sparingly on supercomputing systems, as it makes repeated file open calls.

A.1.4 Custom Output

```
:CustomOutput [time_per] [stat] [variable] [space_agg] {filename}
```

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 D.1), the forcing name (see table D.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. stat is the statistic reported over each time interval, one of:

- AVERAGE
- MAXIMUM
- MINIMUM
- RANGE
- MEDIAN
- CUMULSUM

- 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. The CUMULSUM generates the integrated value over the time aggregation period, e.g., MONTHLY CUMULSUM RAINFALL will generate the monthly total rainfall (in mm) from the integrated rainfall rate (in mm/d).

space_agg refers to the spatial evaluation domain for reporting, and is one of:

- BY_BASIN
- BY_HRU
- BY_HRU_GROUP
- BY_SB_GROUP
- ENTIRE_WATERSHED

In all cases, the variable statistics will be determined using the area-weighted average, i.e., if

```
:CustomOutput MONTHLY MAXIMUM SOIL[0] BY_BASIN
```

is specified, it will report the maximum basin average soil moisture in the top soil layer in any given month, NOT the maximum HRU soil moisture found in the basin within that month.

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.

There are also three special forms of custom output for tracking fluxes between simulated storage compartments. The first reports the cumulative flux from a single storage compartment, using the following syntax:

```
:CustomOutput DAILY AVERAGE From:SNOW BY_HRU
```

where the term after the From: command is a state variable from table D.1. This example returns the cumulative loss from snowpack in the form of snowmelt or sublimation (in mm).

```
:CustomOutput DAILY AVERAGE To:SNOW BY_HRU
```

where the term after the To: command is a state variable from table D.1. This example returns the cumulative gain of snow (in mm).

```
:CustomOutput DAILY AVERAGE Between:SOIL[0].And.ATMOSPHERE BY_HRU
```

where the terms after the `Between:` and `.And.` commands are both state variables from the table D.1. This example returns the cumulative loss of water from the top soil (`SOIL[0]`) to the atmosphere, i.e., the actual evapotranspiration rate from the top soil. To get the total actual evaporation rate, you can use the special state variable `AET`, i.e.,

```
:CustomOutput DAILY AVERAGE AET BY_HRU
```

Note that a special syntax is used for reporting transported constituents (i.e., if the user wishes to generate a custom report of tracer concentrations or temperatures). In this case the state variables are labeled as `!CONSTIT|WATERSTOR|`, where `CONSTIT` is the constituent name (e.g., `SNOW_TRACER`) and `WATERSTOR` is the water storage state variable from table D.1 (e.g., `SOIL[0]`). Thus, to report the percentage of water in the topsoil sourced from snowmelt in each HRU, one could use the following command:

```
:CustomOutput DAILY AVERAGE !SNOW_TRACER|SOIL[0] BY_HRU
```

(this assumes that the `:Transport SNOW_TRACER` command is applied above this entry in the .rvi file.)

If the user requires this custom output file to be written as an Ensim (.tb0) or NetCDF (.nc) file, all `:CustomOutput` commands must be preceded by the `:WriteEnsimFormat` or `:WriteNetCDFFormat` command, respectively. Raven does not support mixing of custom output formats.

```
# Common/Popular Custom Outputs:
# ---------------------------------------

# Daily AET:
:CustomOutput DAILY AVERAGE AET BY_HRU

# Net runoff (includes baseflow) every time step:
:CustomOutput CONTINUOUS AVERAGE RUNOFF BY_HRU

#monthly total gains/losses
:CustomOutput MONTHLY CUMULSUM PRECIP BY_BASIN
:CustomOutput MONTHLY CUMULSUM AET BY_BASIN
```

A.1.5 Hydrologic Processes

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

```
:ProcessName ALGORITHM {ProcessFrom} {ProcessTo}
```

where `:ProcessName` is the name of the process (e.g., :CanopyDrip), `ALGORITHM` refers to the particular algorithm used for simulation (e.g., `CANDRIP_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 D.1.

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 hydrologic 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 usually determined from the chosen algorithm, though there are certain routines (e.g., many baseflow or percolation algorithms) which require the user to specify the 'to' or 'from' compartment.

> 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. In general, processes should be ordered from fast to slow and precipitation and snowmelt should be applied prior to infiltration. This becomes less of an issue with decreasing time step size.

As shown in the template files in appendix F, the :Alias command may be used to give 'nicknames' to state variables which can be used instead of the Raven standard syntax. This is most often done to distinguish between actual storage compartments (e.g., SOIL[1]) and conceptual storage compartments (e.g., the alias ROUTING_STORE). For example,

```
:Alias FAST_RESERVOIR SOIL[1]
:Alias SLOW_RESERVOIR SOIL[2]
# SLOW_RESERVOIR now refers to SOIL[2] when used
```

Table A.1 includes a detailed description of the process commands available in Raven.

The Lateral Flush process

Lateral flow processes may require the specification of the source and destination HRU groups as well as the state variables. The :LateralFlush process, for instance, uses the following syntax:

```
:LateralFlush RAVEN_DEFAULT [SourceGrp] [SourceSV] To [DestGrp] [DestSV]
```

Where the source and destination HRU group (SourceGrp and DestGrp, within a given basin) and source and destination state variable (water compartments SourceSV and DestSV, from table D.1), are specified. For instance,

```
:LateralFlush RAVEN_DEFAULT Uplands SURFACE_WATER To Wetlands DEPRESSION
```

The preceding command will drain all surface and subsurface runoff from the Uplands HRU group to depression storage in an HRU belonging to the Wetlands HRU group. Note that only one recipient HRU in the destination group is allowed in each subbasin (i.e., you couldn't have two HRUs belonging to the Wetlands group in a single subbasin).

The Lateral Equilibrate process

Lateral equilibration is used to represent the basin-wide equilibration of storage over time. It would typically be used to represent groundwater exchange between deep groundwater storage or wetlands. The :LateralEquilibrate process uses the following syntax:

```
:LateralEquilibrate RAVEN_DEFAULT [HRUGroup] [SV] [mix_rate] {INTERBASIN}
```

Where HRUGroup denotes which HRU group this applies to in a given basin (often all HRUs), the SV refers to a source variable from table D.1). mix_rate is the percentage of water equilibrated per day (for a time step of 1.0 and mixing rate >1/d, the storage will be instantaneously equilibrated every day). If the additional INTERBASIN flag is included, the equilibration will occur across the entire simulated watershed, and not just in each subbasin. Note that if there are fewer than 2 HRUs in each group per subbasin, then this command will have no effect.

This replaces the now deprecated :AggregatedVariable command, and also supports transport simulation.

Conditional Application of Processes

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

```
:Flush              RAVEN_DEFAULT PONDED_WATER   SURFACE_WATER
  :-->Conditional   HRU_TYPE IS_NOT GLACIER
:Flush              RAVEN_DEFAULT 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, WETLAND, or STANDARD)
- conditionals based upon land use type, e.g.,

  ```
  :-->Conditional LAND_CLASS IS PEATLAND
  ```

  where LAND_CLASS 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_GROUPs are defined using the :HRUGroup command in the .rvh file.

The only available comparison operators are IS and IS_NOT.

Blended Forcing Functions

The calculation for potential evapotranspiration (PET) and potential melt can be done using a weighed average of multiple algorithms respectively, which is referred to as a blended forcing. For PET, this is done by specifying PET_BLENDED as the :Evaporation method, and specifying the selected PET algorithms and weights in the following line with the :BlendedPETWeights command. For example,

```
:Evaporation           PET_BLENDED
:BlendedPETWeights PET_GRANGERGRAY 0.333 PET_HAMON 0.333 ...
  PET_PENMAN_MONTEITH 0.334
```

Equivalently, one fewer value than the number of required weights N can be specified, in which case Raven treats the supplied values as weight-generating parameters, and uses the pie-sharing method of Mai et al. (2022) to determine N weights from $N-1$ values. For example,

```
:Evaporation           PET_BLENDED
:BlendedPETWeights PET_GRANGERGRAY 0.5556 PET_HAMON 0.50 PET_OUDIN
```

Similarly, the potential melt blend can be specified by providing the potential melt method as POTMELT_BLENDED, and using the :BlendedPotMeltWeights command in the following line. The associated weights or weight-generating parameters follow the same syntax as for :BlendedPETWeights. For example,

```
:PotentialMeltMethod           POTMELT_BLENDED
:BlendedPotMeltWeights POTMELT_HMETS 0.75 POTMELT_RESTRICTED
```

A.1.6 Transport Commands

```
:Transport [constit_name]
```

(Optional) This command declares a new transport constituent named constit_name which can be advected through the system. The constituent name is user-specified. If constit_name is one of the reserved names TEMPERATURE, 18O, or 2H, the constituents are treated as enthalpy, Oxygen-18, and

Deuterium, respectively, and additional processes are activated in the simulation. This command must appear before any of the other transport-related commands in the .rvi file.

```
:FixedConcentration [constit_name] [compartment] [conc] {HRUgrp}
```

(Optional) This command applies a type one boundary condition in all water storage compartment state variables of type `compartment` (taken from the state variable list of table D.1) in HRU group `HRUgrp`. All water passing through this storage compartment will be assigned the specified concentration (`conc`, in mg/l)for the constituent named `constit_name`. 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 synthetic tracers, it is useful to specify a concentration of 1.0 (unitless). For energy transport, use the `:FixedTemperature` command. For a non-constant specified concentration condition, the `:FixedConcentrationTimeSeries` command (in the .rvt file) should be used instead.

```
:FixedTemperature [compartment] [temp] {HRUgrp}
```

(Optional) This command applies a type one fixed concentration boundary condition in all water storage compartment state variables of type `compartment` (taken from the state variable list of table D.1) in (optional) HRU group `HRUgrp`. All water passing through this storage compartment will be assigned the specified temperature (`temp`). Note that the `TEMPERATURE` constituent 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. The units of `temp` are °C. A special flag is used to indicate that precipitation temperature should be equal to air temperature:

```
:FixedTemperature ATMOS_PRECIP -9999
```

```
:MassFlux [compartment] [flux] {HRUgrp}
```

(Optional) This command applies a type three mass influx boundary condition in all water storage compartment state variables of type `compartment` (taken from the state variable list of table D.1) in (optional) HRU group `HRUgrp`. The constant flux of constituent (specified in units of mg/m^2/d) is unaccompanied by addition of water. This must exist after the corresponding `:Transport` command. If the optional `HRU_group` is omitted, then the condition applies to all storage compartments of this type throughout the watershed. For a non-constant flux, the `:MassInfluxTimeSeries` command (in the .rvt file) should be used instead.

A.1.7 Geochemical Processes

The geochemical processes should be within a `:GeochemcialProcesses`-`:EndGeochemicalProcesses` command block and must appear after the `:Transport` commands which define the constituents that may participate in (bio)geochemical processes.

```
:GeochemicalProcesses
  :ProcessName [ALGORITHM] [proc_name] [constit_1] {compart}
  # or
  :ProcessName [ALGORITHM] [proc_name] [constit_1] [constit_2] {comp}
:EndGeochemicalProcesses
```

Each geochemical process must have a corresponding process command `:ProcessCommand` (e.g., `:Decay`), algorithm name `ALGORITHM` (e.g., `DECAY_LINEAR`), user-specified process name `proc_name` which usually defines the physical process this represents (e.g., `DENITRIFICATION`) and the constituent name `constit_1` from a existing `:Transport` command. For processes that involve two constituents (two different chemical species or sorbed/aqueous components of the same species), a second constituent name `constit_2` must be supplied. The optional parameter `comp` (e.g., `SOIL[0]`) refers to a water storage compartment in which the process is enabled. If no compartment is specified, this process is applied in all water storage compartments in all HRUs. Conditional application of processes in space should be handled via local parameterization - for example, setting the decay coefficient equal to zero in soil class types or storage compartments where decay should not occur.

All of the following commands are geochemical processes that would be specified within this command block.

```
:Decay [ALGORITHM] [proc_name] [constit] {compart}
```

Supports the following `ALGORITHM` options, discussed in section 7.6.1:

- `DECAY_LINEAR`
- `DECAY_DENITRIF`

```
:Equilibrium [ALGORITHM] [proc_name] [constit1] [constit2] {compart}
```

Supports the following `ALGORITHM` options, discussed in section 7.6.1:

- `EQUIL_FIXED_RATIO`
- `EQUIL_LINEAR`
- `EQUIL_LINEAR_SORPTION` - note: constituent 1 should be aqueous, 2 should be sorbed

```
:Transformation [ALGORITHM] [proc_name] [constit1] [constit2] {compart}
```

Supports the following `ALGORITHM` options, discussed in section 7.6.3:

- `TRANS_LINEAR`
- `TRANS_NONLINEAR`

A.1.8 Other Control Commands

```
:RedirectToFile [filename]
```

(Optional) This treats the contents of file "filename" as if they were simply inserted into the .rvi file at the location of the `:RedirectToFile` command. Both relative and absolute path names for the file are accepted, where relative path names are relative to the location of the .rvi file (i.e., input/Weights.txt would be a file one directory below the .rvi directory).

```
:ApplyManagementOptimization
```

(Optional) Enables management optimization. With this option management optimization is turned on and the .rvm file is read. See section 4.5 for more details.

```
:DisableHRUGroup [HRUgrp]
```

(Optional) This command disables all of the HRUs in the group, meaning that the model will not simulate the mass/energy balance for any of the HRUs. For instance, if you had a large model and only wanted to simulate a single headwater basin, you would create an HRU group that included HRUs not within that basin, then apply the `:DisableHRUGroup` command to that single group of HRUs. In most cases, it is desirable to disable entire subbasins - the model will not provide comprehensible results if random assortments of individual HRUs are disabled.

```
:AssimilateStreamflow
```

(Optional) If this command is used, specified streamflow observations (indicated using the `:AssimilateStreamflow` command in the .rvt file) are assimilated into model predictions as indicated in section 6.1. Model output will then be a combination of data and model simulation; this is typically used only in a forecasting environment. Note that streamflow assimilation manifests as mass balance errors as reported in the WatershedStorage.csv output, because it necessarily adds or removes water to respect observation data.

```
:AssimilateReservoirStage
```

(Optional) If this command is used, all available reservoir stage observations are assimilated into model predictions as indicated in section 6.1. Model output will then be a combination of data and model simulation; this is typically used only in a forecasting environment. Note that stage assimilation manifests as mass balance errors as reported in the WatershedStorage.csv output, because it necessarily adds or removes water to respect observation data.

```
:ReservoirDemandAllocation [method]
```

(Optional) This command indicates how irrigation demand addressed using the `:ReservoirDownstreamDemand` command is distributed to upstream reservoirs. Here, two methods are available:

- `DEMANDBY_CONTRIB_AREA` if demand is allocated proportionately to contributing area of each reservoir
- `DEMANDBY_MAX_CAPACITY` if reservoir maximum storage capacity is used.

This command is only used if the `_AUTO` flag is used to specify the reservoirs supplying a single `:ReservoirDownstreamDemand` item.

```
:RandomSeed [seed]
```

Indicates the random seed to be used in any internal probabilistic calculations. Note that the internal random seed is connected to the model start date, such that any probabilistic simulation with the same random seed and model start date and time will give identical results.

```
:CallExternalScript [system command]
```

(Optional) This calls an external script or system command at the start of every time step. If the system command includes the tags `<model_time>`, `<date>`, `<version>`, or `<output_dir>`, then these tags in the command will be replaced with the model time (time from the start of the model, in days), date string in ISO standard format, version number (e.g., 3.0.2), or output directory, respectively. This can

be used, for instance, to assist in coupling multiple models, and would typically be used with the Raven live (.rvl) file. Because Raven writes output files continuously throughout the simulation, current states of the system may be read from the last line of any .csv file (or using the start of time step time stamp in a NetCDF file), processed accordingly from the external script, with script output written to the live file for Raven to ingest in time for the next time step of calculations. External scripting may support (e.g.),

- representing complex region-specific management plans by reading in flows and modifying diversion quantities
- representing reservoir operations plans that incorporate economic optimization
- coupling to other earth systems models (e.g., glacier mass balance or hydraulic models)
- coupling to agricultural or urbanization models of changing land cover

```
:ReadLiveFile [frequency]
```

(Optional) Reads externally-generated live file (see appendix A.7) every `frequency` time steps.

```
:ChunkSize [size]
```

(Optional) Specifies memory buffer size for NetCDF files, in megabytes. Defaults to 10MB.

```
:Mode [mode character]
:IfModeEquals [mode character]
#...
:EndIfModeEquals
```

(Optional) A conditional statement which can be used to control which parts of any base input file (.rvi,.rvp,.rvh,.rvt, or .rvc) are used in the simulation. If the mode character, as specified with the `:Mode` command or the `-m` command line argument is not equal to the mode character in the `:IfModeEquals` statement, the contents of the block are ignored. These work identically to if statements in a programming language, but they cannot be nested and there is no if-else equivalent. The mode must be a single alphanumeric character. Example usage:

```
:IfModeEquals A
  :EndDate 2020-10-31 00:00:00 # full simulation, output suppressed
  :SuppressOutput
:EndIfModeEquals
:IfModeEquals B
  :EndDate 2016-10-31 00:00:00 # truncated simulation, full output
  :WriteForcingFunctions
  :WriteWatershedStorage
:EndIfModeEquals
```

Process	Algorithms	"From" Storage Compartments	"To" Storage Compartments
Precipitation			
:Precipitation	PRECIP_RAVEN	ATMOS_PRECIP*	*All**
Evapotranspiration/Evaporation Processes			
:CanopyEvap	CANEVP_RUTTER	CANOPY	ATMOSPHERE
	CANEVP_MAXIMUM	CANOPY	ATMOSPHERE
:SoilEvaporation	SOILEVAP_VIC	SOIL[0]	ATMOSPHERE
	SOILEVAP_HBV	SOIL[0]	ATMOSPHERE
	SOILEVAP_CHU	SOIL[0]	ATMOSPHERE
	SOILEVAP_TOPMODEL	SOIL[0]	ATMOSPHERE
	SOILEVAP_SEQUEN	SOIL[0]	ATMOSPHERE
	SOILEVAP_ROOTFRAC	SOIL[0]	ATMOSPHERE
	SOILEVAP_GAWSER	SOIL[0]	ATMOSPHERE
:LakeEvaporation	LAKE_EVAP_BASIC	LAKE, SURFACE_WATER*	ATMOSPHERE
:OpenWaterEvaporation	OPEN_WATER_EVAP	DEPRESSION	ATMOSPHERE
Soil Processes			
:Infiltration	INF_RATIONAL	PONDED_WATER	SOIL[0], SURFACE_WATER
	INF_SCS	PONDED_WATER	SOIL[0], SURFACE_WATER
	INF_ALL_INFILTRATES	PONDED_WATER	SOIL[0], SURFACE_WATER
	INF_GREEN_AMPT	PONDED_WATER	SOIL[0], SURFACE_WATER
	INF_GA_SIMPLE	PONDED_WATER	SOIL[0], SURFACE_WATER
	INF_UPSCALED_GREEN_AMPT	PONDED_WATER	SOIL[0], SURFACE_WATER
	INF_HBV	PONDED_WATER	SOIL[0], SURFACE_WATER
	INF_UBC	PONDED_WATER	SOIL[0], SOIL[1], SOIL[2], SOIL[3],SURFACE_WATER
	INF_VIC	PONDED_WATER	SOIL[0], SURFACE_WATER
	INF_VIC_ARNO	PONDED_WATER	SOIL[0], SURFACE_WATER
	INF_PRMS	PONDED_WATER	SOIL[0], SURFACE_WATER
:Percolation	PERC_GAWSER	SOIL[m]/AQUIFER*	SOIL[m]/AQUIFER*
	PERC_LINEAR	SOIL[m]/AQUIFER*	SOIL[m]/AQUIFER*
	PERC_POWER_LAW	SOIL[m]/AQUIFER*	SOIL[m]/AQUIFER*
	PERC_PRMS	SOIL[m]/AQUIFER*	SOIL[m]/AQUIFER*
	PERC_SACRAMENTO	SOIL[m]/AQUIFER*	SOIL[m]/AQUIFER*
	PERC_CONSTANT	SOIL[m]/AQUIFER*	SOIL[m]/AQUIFER*
	PERC_GR4J	SOIL[m]/AQUIFER*	SOIL[m]/AQUIFER*
:CapillaryRise	CRISE_HBV	SOIL[m]/AQUIFER*	SOIL[m]/AQUIFER*
:Baseflow	BASE_LINEAR	SOIL[m]/AQUIFER*	SURFACE_WATER
	BASE_POWER_LAW	SOIL[m]/AQUIFER*	SURFACE_WATER
	BASE_CONSTANT	SOIL[m]/AQUIFER*	SURFACE_WATER
	BASE_VIC	SOIL[m]/AQUIFER*	SURFACE_WATER
	BASE_THRESH_POWER	SOIL[m]/AQUIFER*	SURFACE_WATER
	BASE_GR4J	SOIL[m]/AQUIFER*	SURFACE_WATER
	BASE_TOPMODEL	SOIL[m]/AQUIFER*	SURFACE_WATER
:Interflow	PRMS	SOIL[m]*	SURFACE_WATER
Wetland/Depression/Lake Processes			
:Seepage	SEEP_LINEAR	DEPRESSION	SOIL[m]*
:DepressionOverflow	DFLOW_THRESHPOW	DEPRESSION	SURFACE_WATER
	DFLOW_LINEAR	DEPRESSION	SURFACE_WATER
:LakeRelease	LAKEREL_LINEAR	LAKE	SURFACE_WATER
:Abstraction	ABST_PERCENTAGE	PONDED_WATER	DEPRESSION
	ABST_FILL	PONDED_WATER	DEPRESSION
	ABST_SCS	PONDED_WATER	DEPRESSION

Table A.1: Hydrologic process commands for the .rvi file. Compartments with an asterisk must be specified within the command.

Snow Processes			
:SnowMelt	MELT_POTMELT	SNOW	SNOW_LIQ,PONDED_WATER,SURFACE_WATER*
:Snow Refreeze	FREEZE_DEGREE_DAY	SNOW_LIQ	SNOW
:Snow Balance	SNOBAL_SIMPLE_MELT	SNOW	PONDED_WATER,SNOW_LIQ*
	SNOBAL_COLD_CONTENT	SNOW,SNOW_LIQ	SNOW, SNOW_LIQ,PONDED_WATER
	SNOBAL_HBV	SNOW,SNOW_LIQ	SOIL[0]
	SNOBAL_TWO_LAYER	SNOW[0,1],SNOW_LIQ[0,1]	SNOW[0,1],SNOW_LIQ[0,1],SURFACE_WATER
	SNOBAL_CEMA_NEIGE	SNOW	PONDED_WATER
	SNOBAL_GAWSER	SNOW_SNOW_LIQ	SNOW_LIQ,ATMOSPHERE,PONDED_WATER
	SNOBAL_UBC	SNOW,SNOW_LIQ	SNOW,SNOW_LIQ,SURFACE_WATER
:Sublimation	SUBLIM_SVERDRUP	SNOW	ATMOSPHERE
	SUBLIM_KUZMIN	SNOW	ATMOSPHERE
	SUBLIM_CENTRAL_SIERRA	SNOW	ATMOSPHERE
	SUBLIM_PSBM	SNOW	ATMOSPHERE
	SUBLIM_WILLIAMS	SNOW	ATMOSPHERE
:SnowAlbedoEvolve	SNOALB_UBC		
Vegetation			
:CanopyDrip	CANDRIP_RUTTER	CANOPY	PONDED_WATER
	CANDRIP_SLOWDRAIN		
:CropHeatUnitEvolve	CHU_ONTARIO		
Glacier Processes			
:GlacierMelt	GMELT_SIMPLE_MELT	GLACIER_ICE	GLACIER
	GMELT_HBV	GLACIER_ICE	GLACIER
	GMELT_UBC	GLACIER_ICE	GLACIER
:GlacierRelease	GRELEASE_LINEAR	GLACIER	SURFACE_WATER
	GRELEASE_HBV_EC	GLACIER	SURFACE_WATER
Special Processes			
:Flush	FLUSH_RAVEN	any*	any*
:Overflow	OVERFLOW_RAVEN	any*	any*
:Split	RAVEN_DEFAULT	any*	any*
:Convolution	CONVOL_GR4J1	any*	CONVOLUTION[m]*
	CONVOL_GR4J2	any*	CONVOLUTION[m]*
:LateralFlush	RAVEN_DEFAULT	any*	any*

Table A.2: Hydrologic process commands for the .rvi file. (cont'd)

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.

```
# ----------------------------------------
# Raven Example Classed Parameter File
# ----------------------------------------
# Class definition ---------------------------
: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
: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
# Soil Profile definition -------------------
:SoilProfiles
#     name, #horizons, hor1, th1, hor2, th2
      LAKE,         0
   GLACIER,         0
  LOAM_SEQ,         2, LOAM, 0.5, SAND, 1.5
  ALL_SAND,         2, SAND, 0.5, SAND, 1.5
:EndSoilProfiles
# Parameter specification -------------------
:GlobalParameter WET_ADIABATIC_LAPSE 0.5
:LandUseParameterList
  :Parameters,  MELT_FACTOR, MIN_MELT_FACTOR
  :Units     ,       mm/d/K,          mm/d/K
    [DEFAULT],          3.2,             1.3
    GRASSLAND,          3.5,       _DEFAULT
:EndLandUseParameterList
```

As with the *.rvi file, ⋆ or # denotes a comment.

A.2.1 Required Commands

```
:SoilClasses
  {soil_class_name}x[NSC]
:EndSoilClasses
```

or

```
:SoilClasses
  :Attributes     ,%SAND,%CLAY,%SILT, %ORGANIC
  :Units          , none, none, none,    none
  {soil_class_name,%sand,%clay,%silt,%organic}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 .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 the 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 in appendix A.2.3, may be added to the input file *after* the `:SoilClasses` command has been called. For conceptual models, the soil composition will generally not be specified.

```
:SoilProfiles
  {profile_name,num_horiz,{soil_class_name,thick.}x{#horizons}}x[NP]
:EndSoilProfiles
```

Defines all `NP` stored soil profiles, which is a collection of soil horizons with known depth and thickness, each belonging to a soil class. Each profile includes a number of horizons (`num_horiz`) followed by the soil class names and thicknesses of each horizon. The soil horizons should be specified from the top downward. Because the parameter `soil_class_name` is required, this command must come after the `:SoilClasses` command. The thickness (`thick.`) of each horizon is specified in meters.

In the case of conceptual hydrologic models, these soil horizons may correspond instead to conceptual soil stores. These typically correspond to physically meaningful layers (e.g., topsoil, unsaturated zone, and deep groundwater), but additional "soil" layers may represent routing stores or other temporary storage reservoirs.

The special cases of lakes, exposed rock, wetlands, 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`, `WATER`, `ROCK`, `WETLAND`, and `GLACIER`, all with zero horizons. ANY soil profile that starts with these terms is not subject to soil-based process algorithms.

Glaciers can have more than zero horizons to represent groundwater processes, but infiltration and evapotranspiration from the surface soil is disabled. HRUs with LAKE profiles have their canopy processes disabled.

```
: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 .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.

The maximum LAI and maximum canopy height are used to deterimine the LAI and canopy height over the course of the year, using the :SeasonalCanopyLAI and :SeasonalCanopyHeight commands.

```
:LandUseClasses
  :Attributes   ,IMPERMEABLE_FRAC, FOREST_COVERAGE
  :Units        ,           fract,           fract
  {LU_class_name,IMPERMEABLE_FRAC, FOREST_COVERAGE}x[NLU]
:EndLandUseClasses
```

Defines all NLU 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 .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. It is recommended (but not required) to use either 0 (open) or 1 (fully forested), with partial coverage handled via HRU definition.

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 NTC 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 .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 or equal to the leftmost (smallest) survey coordinate x. The channel configuration definitions are depicted in figure A.1. A representative bedslope (expressed as the slope ratio) is also needed: this is used to calculate flow rates using Manning's equation.

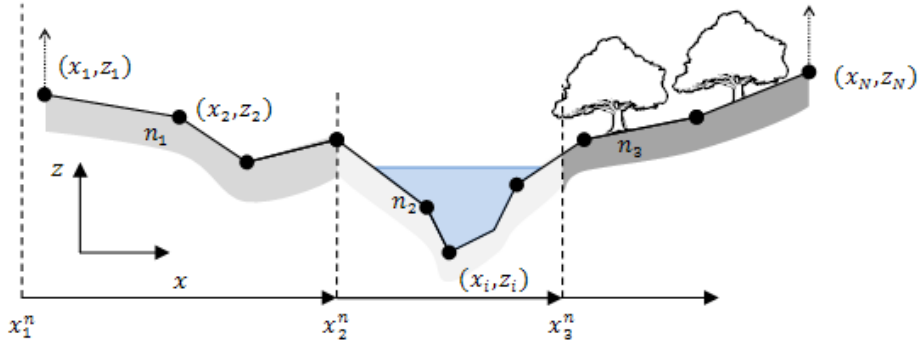

Figure A.1: Channel Profile definition. Each channel is defined by a cross sectional profile and a number of zones with different Manning's n values.

As an example, the following profile command generates the channel shown in figure A.2.

```
:ChannelProfile Reach3
  :Bedslope 0.08
  :SurveyPoints
    0.000 0.25
```

```
    1.000 0.00
    1.750 0.00
    2.000 0.25
  :EndSurveyPoints
  :RoughnessZones
    0.000 0.07
    0.500 0.02
    1.875 0.08
  :EndRoughnessZones
:EndChannelProfile
```

Note that it is undesirable to overly constrain the lateral extent of the channel, i.e., if there is any chance that the water levels reach the leftmost or rightmost channel point. Also note that Manning's n and slope may both be overwritten for a specific subbasin via the `:SubBasinProperties` command in the .rvh file.

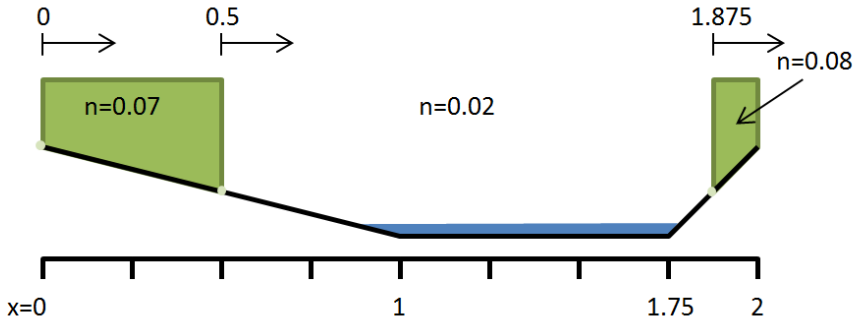

Figure A.2: Example channel profile generated using example command.

```
:ChannelRatingCurves [channel_name]
  :Bedslope [slope]
  :StageRelations
    {[stage] [area] [width] [flow]} x num curve points
  :EndStageRelations
: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].

```
:TrapezoidalChannel [name] [bot_width] [bot_elev] [incline] [n] [slope]
```

An abbreviated alternative to the `:ChannelProfile` command. Defines a trapezoidal channel profile with the unique name `name`, and is used as an alternative to `:ChannelProfile`. The parameters are the bottom width `bot_width` (in [m]), bottom elevation `bot_elev`, incline `incline` (unitless), Manning's roughness coefficient `n` (unitless), and bed slope `slope`, expressed as a slope ratio. The bottom elevation is only used as a datum for reporting water level. The incline should be interpreted as N in an 1:N slope, e.g., it is 0 for vertical channel sides, 1 for a 45° angle. Note that this option can be used to represent a rectangular channel (`incline`=0) or a triangular channel (`bot_width`=0).

```
:CircularConduit [name] [diameter] [bot_elev] [n] [slope]
```

An abbreviated alternative to the `:ChannelProfile` command. Defines a circular conduit (e.g., culvert) profile with the unique name `name`, and is used as an alternative to `:ChannelProfile`. The parameters are the diameter `diameter` (in [m]), bottom elevation `bot_elev`, Manning's roughness coefficient `n` (unitless), and bed slope `slope`, expressed as a slope ratio. The bottom elevation is only used as a datum for reporting water level.

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  , { par_name1, par_name1,..., par_nameNP}
  :Units       , {unit_type1,unit_type2,...,unit_typeNP}
  [DEFAULT]    , {  def_val1,  def_val2,...,  def_valNP} [opt.]
  {SOIL_CLASS} , {  par_val1,  par_val2,...,  par_valNP}}x[<=NSC]
:EndSoilParameterList
```

where available soil parameter names (`par_name`) are described in the table A.3 and the soil class names (with the exception of the special `[DEFAULT]` tag) must already have been declared in the `:Soil-Classes` 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.

Advice

Note that the units must be consistent with the native units of each parameter indicated in table A.3 - 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 (e.g., hydraulic conductivity) have a physical basis, certain algorithms, particularly for lumped models, abstract a physical process so that coefficients in the relationships between storage and fluxes are completely conceptual. These conceptual 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.3.

Vegetation Parameter Specification

```
:VegetationParameterList
  :Parameters     , { par_name1, par_name1,..., par_nameNP}
  :Units          , {unit_type1,unit_type2,...,unit_typeNP}
  [DEFAULT]       , {  def_val1,  def_val2,...,  def_valNP} [opt.]
  {VEG_CLASS_NAME , {  par_val1,  par_val2,...,  par_valNP}}x[<=NVC]
:EndVegetationParameterList
```

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.5. Note that the [DEFAULT] vegetation type is optional.

```
: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 or as harvesting occurs, i.e., $LAI = LAI_{max} \cdot f$, where $f(t)$ is the monthly correction factor specified here 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 or as harvesting occurs, i.e., $h_{veg} = h_{max} \cdot f$, where $f(t)$ is the monthly correction factor specified here 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
  :Parameters  , { par_name1, par_name1,..., par_nameNP}
  :Units       , {unit_type1,unit_type2,...,unit_typeNP}
  [DEFAULT]    , {  def_val1,  def_val2,...,  def_valNP} [opt.]
  {LULT_CLASS} , {  par_val1,  par_val2,...,  par_valNP}}x[<=NLC]
:EndLandUseParameterList
```

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

Global Parameter Specification

The following global parameters can also be specified, anywhere in the .rvp file. Note that the preferred format for single-value parameters (i.e., not vectors of parameters) is to use the `:GlobalParameter` command. Many of the below commands are equivalent to this command, retained only for backwards compatibility with earlier versions of Raven.

```
:GlobalParameter [PARAM_NAME] [value]
```

Can be used to specify the value of any scalar global parameter, where the list of global parameter names is in table A.6.

Please note that the `:GlobalParameter` command is the only one truly needed to specify single-valued global parameters in table A.6. The remainder of the commands shown below have been deprecated, and are only provided as a reference for those using older models which may include these commands. The only exception to this are the global parameters which include monthly sequences (e.g., `:UBCNorthSWCorr`)

```
:AdiabaticLapseRate [rate]
# is equivalent to (the preferred option)
:GlobalParameter ADIABATIC_LAPSE [rate]
```

The base adiabatic lapse rate [°C/km].

```
:PrecipitationLapseRate [rate]
# is equivalent to (the preferred option)
:GlobalParameter PRECIP_LAPSE [rate]
```

The simple linear precipitation lapse rate [mm/d/km], as used in the `OROCORR_SIMPLELAPSE` orographic correction algorithm.

```
:RainSnowTransition [rainsnow_temp] [rainsnow_delta]| \\ %
# equivalent to (the preferred option)
:GlobalParameter RAINSNOW_TEMP [rainsnow_temp]
:GlobalParameter RAINSNOW_DELTA [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]
# equivalent to (the preferred option)
:GlobalParameter SNOW_SWI [saturation]
```

Maximum liquid water content of snow, as percentage of SWE [0..1]. Usually ~0.05.

```
:AvgAnnualRunoff [runoff]
# equivalent to (the preferred option)
:GlobalParameter AVG_ANNUAL_RUNOFF [runoff]
```

This parameter should be the average annual runoff for the entire simulated watershed, [mm/yr]. 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.

```
:WetAdiabaticLapseRate [rate] [A0PPTP]
# equivalent to (the preferred option)
:GlobalParameter WET_ADIABATIC_LAPSE [rate]
:GlobalParameter UBC_A0PPTP [A0PPTP]
```

The wet adiabatic lapse rate [$^{\circ}$C/km] and the UBCWM threshold precipitation, A0PPTP, for temperature lapse rate [mm/d] (usually $\sim$5 mm/d).

```
:ReferenceMaxTemperatureRange [range]
# equivalent to (the preferred option)
:GlobalParameter UBC_MAX_RANGE_TEMP [range]
```

A parameter (A0TERM) used in the UBC watershed model orographic corrections for temperature [$^{\circ}$C].

```
:UBCTempLapseRates [A0TLXM A0TLNM A0TLXH A0TLNH P0TEDL P0TEDU]
```

Parameters used in the UBC watershed model orographic corrections for temperature. A0TLXM and A0TLXH [$^{\circ}$C/km] are the low and high elevation lapse rates of the maximum daily temperature; A0TLNM and A0TLNH [$^{\circ}$C/km] are the low and high elevation lapse rates of the minimum daily temperature; P0TEDL and P0TEDU [$^{\circ}$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 [m]; 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 in the UBCWM PET orographic correction algorithm [$^{\circ}$C/km].

```
:UBCNorthSWCorr [J F M A M J J A S O N D]
```

Monthly correction factors (unitless) 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 (unitless) 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 [value]
```

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 [value]
```

The UBCWM fraction of solar radiation penetrating cloud cover [0..1], as used in the SW_CLOUD_–CORR_UBCWM cloud cover correction algorithm.

```
:UBCLWForestFactor [value]
```

The UBCMW Longwave correction factor for forests [mm/d/K](∼0.75), as used in the LW_RAD_UBCWM longwave radiation estimation routine.

```
:AirSnowCoeff [value]
```

This is the air/snow heat transfer coefficient in units of [1/d], as used in the SNOTEMP_NEWTONS snow temperature evolution routine.

```
:AvgAnnualSnow [value]
```

This parameter is the average annual snow for the entire watershed in mm SWE. It is used in the CEMA_NEIGE snowmelt algorithm.

In addition to the above specifications for global parameters, Raven can also override 'global' parameters in specific subbasins through the use of subbasin groups. These are detailed below.

```
:GlobalParameterOverride [param_name] [group_name] [value]
```

This command allows the user to override the global parameter specified for a specific subbasin group, in effect allowing for a distributed set of global parameters. Here, the group_name refers to a subbasin group created using the :SubBasinGroup command, the param_name is one of the global parameters named in table A.6. Note that this override should be using sparingly, as it overrides the global parameter of the relevant HRUs in each time step, and can increase the computational cost of the model substantially.

```
:SBGroupOverrideWeights [param_name] [group_name] [w1] ... [wN]
:SBGroupOverrideWeights [param_name] [group_name] [u1] ... [u{N-1}]
```

This command is typically used for calibration, allowing the user to specify blended forcing weights (currently for either potential melt or potential evapotranspiration) for a specific subbasin group, overriding the global blended forcing weights specified in the .rvi file. The `group_name` refers to a subbasin group created using the `:SubBasinGroup` command, the `param_name` is one of the parameters `PET_BLEND_WTS` or `POTMELT_BLEND_WTS`, and either the user can either specify N weights or $N-1$ independent uniform numbers (weight-generating parameters) used to calculate weights (see `:ProcessGroup` command for more information on calculating N weights from $N-1$ independent uniform numbers in blended models).

Transport/Geochemical Parameters

```
:GeochemParameter [pname] [c1] {c2} [process] {comp.} {class} [value]
# e.g.,
:GeochemParameter DECAY_COEFF  NITRATE DENITRIFICATION SOIL[1] 0.0099
:GeochemParameter DECAY_COEFF  STRONTIUM RAD_DECAY  0.000066
```

Can be used to specify the value of any scalar geochemical parameter, where the list of parameter names (`pname`) is in table A.7. The `c1` argument is the name of the constituent, the (optional) `c2` is the name of the 2nd constituent for transformation parameters, `process` is the same process name as indicated in the corresponding geochemical process definition in the .rvi file, (optional) `comp.` is a state variable water storage (e.g., `SOIL[0]`), (optional) `class` is the soil class the parameter is linked to. `value` is the value of the parameter, with units identified in table A.7.

```
:StochiometricRatio [c1] [c2]  [value]
```

The ratio of mass produced of constituent 2 to mass lost of constituent 1.

Special Commands

The following special commands can be used for temporally variable landscape change (e.g., to simulate urbanization, glacial retreat, 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. All land use changes prior to the model start date are processed at the start of the simulation, so if land use change occurs prior to t=0, the most recent will be treated as the initial HRU land cover. Any series of `:LandUseChange` commands should be input in chronological order.

```
: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. All vegetation changes prior to the model start date are processed at the start of the simulation, so if vegetation change occurs prior to t=0, the most recent will be treated as the initial HRU vegetative cover. Any series of `:VegetationChange` commands should be input in chronological order.

```
:HRUTypeChange [HRU group] [new type tag] [YYYY-mm-dd]
```

The HRU type vegetation for the specified HRU group is changed to the new HRU type (`GLACIER`, `ROCK`, `WETLAND`, `STANDARD`, or `LAKE`) on the specified date in ANSI YYYY-mm-dd format. This command is mostly used to represent conversion from glacier to non-glacier. The change occurs just after midnight of the night before. There is no limit to the number of HRU type changes in the model. All HRU type changes prior to the model start date are processed at the start of the simulation. Any series of `:HRUTypeChange` commands should be input in chronological order.

```
:TransientParameter [PARAM_NAME] [class] {(optional) ClassName}
  [date yyyy-mm-dd] [time hh:mm:ss.0] [interval] [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, `interval` is the time interval of the supplied time series and `N` is the total number of entries. `PARAM_NAME` corresponds to one of the parameters included in tables A.3, A.4, A.5, or A.6. `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. This command must be provided in the main model .rvp file and not within nested files. Note that this command cannot work within data blocks (e.g., a the entire `:SoilParameterList-:EndSoilParameterList` block would have to be in a single file, not just the tabular data in that block).

Table A.3: 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.

Name	Definition	Units	Range	
SAND_CON	percent sand content of mineral soil (sand+clay+silt=1)	[0..1]	0.0-1.0	
CLAY_CON	percent clay content of mineral soil	[0..1]	0.0-1.0	
SILT_CON	percent silt content of mineral soil	[0..1]	0.0-1.0	
ORG_CON	percent organic content of soil (mineral+org.=100%)	[0..1]	0.0-0.8	
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 (?ae)	[-mm]		
WILTING_PRESSURE	(positive) wilting pressure	[-mm]		
HEAT_CAPACITY	saturated volumetric heat capacity	[MJ/m3/K]		
THERMAL_COND	saturated soil thermal conductivity	[MJ/d/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 pressure	[-]		
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]	0.01 - 1000	
PERC_N	VIC/ARNO percolation exponent	[-]	1.00 - 20	
PERC_COEFF	linear percolation coefficient	[1/d]		
SAC_PERC_ALPHA	Sacramento percolation multiplier	[-]	1.0 - 250.0	
SAC_PERC_EXPON	Sacramento percolation exponent	[-]	1.00 - 5.0	
HBV_BETA	HBV infiltration exponent	[-]	0.0 - 7.0	
MAX_BASEFLOW_RATE	maximum baseflow rate	[mm/d]	0.001 - 1000	
BASEFLOW_N	VIC/ARNO baseflow exponent	[-]	1.0 - 10.0	
BASEFLOW_COEFF	linear baseflow storage/routing coefficient	[1/d]		
BASEFLOW_COEFF2	linear baseflow storage/routing coefficient	[1/d]		
BASEFLOW_THRESH	threshold saturation for onset of baseflow	[0..1]		
BF_LOSS_FRACTION	percentage of baseflow directed to deep GW	[0..1]		
STORAGE_THRESHOLD	threshold water content for onset of baseflow	[mm]	0-100	
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 deficit	[mm]		
GR4J_X2	GR4J Maximum groundwater exchange rate	[mm/d]		
GR4J_X3	GR4J reference storage for baseflow/GW exchange	[mm]		

Table A.4: Land use parameters. The parameters with an asterisk can be autogenerated by Raven or overriden by the model user.

Name	Definition	Units	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]	0-1000	
MAX_DEP_AREA_FRAC	percentage of landscape covered by depressions when full	[0..1]	0-0.8	
DD_MELT_TEMP	melting temperature for degree day methods	[°C]	0.0	
MELT_FACTOR*	maximum snow melt factor used in degree day models	[mm/d/ °C]	3.5	Conceptual Model Parameters
DD_REFREEZE_TEMP*	degree day reference (freezing) temperature	[°C]	0.0	
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	
REFREEZE_EXP	exponent used in HMETS_SNOWBAL refreeze relationship	[-]	0.5	
DD_AGGRADATION	degree day increase rate with cumulative melt (HMETS pot. melt.)	[1/mm]	0.1	
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_SEEP_K	depression linear seepage constant	[1/d]	0.001-0.5	
DEP_K	depression linear overflow constant	[1/d]	0.01-0.5	
DEP_THRESHOLD	threshold storage at which flow commences	[mm]		
PDM_B	soil pareto distribution parameter	[-]		
PDMROF_B	wetland pareto distribution parameter	[-]		
PONDED_EXP	exponent used in SOILEVAP_HYPR model	[-]	1-5	
OW_PET_CORR*	fraction of PET to apply to open water evaporation	[-]	0.2-1.2	
LAKE_PET_CORR*	fraction of PET to apply to lake evaporation	[-]	0.8-1.2	
LAKE_REL_COEFF	linear lake storage coefficient	[1/d]	0.001-0.5	
FOREST_PET_CORR*	fraction of PET to apply to forest evapotranspiration	[-]	0.1-1.2	
GAMMA_SCALE{2}	Gamma unit hydrograph scale parameters	[1/d]	0.1-20	
GAMMA_SHAPE{2}	Gamma unit hydrograph shape parameters	[-]	0.5-5	
HMETS_RUNOFF_COEFF	HMETS runoff coefficient	[0..1]	0.3-1 (<1)	
AET_COEFF	SOILEVAP_LINEAR proportionality constant	[1/d]	0.05	
HYMOD2_G	SOILEVAP_HYMOD2 ET lower resistance parameter	[0..1]		
HYMOD2_KMAX	SOILEVAP_HYMOD2 ET resistance parameter	[0..1]	1	
HYMOD2_EXP	SOILEVAP_HYMOD2 ET exponent	[-]	1	
GR4J_X4	GR4J time routing parameter	[d]	0-100	
UBC_ICEPT_FACTOR*	UBC Interception factor	[-]		

Table A.5: Vegetation Parameters. The parameters with an asterisk can be automatically generated by Raven or overridden by the model user.

Name	Definition	Units	Range	
MAX_HEIGHT	maximum vegetation height	[m]		
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	Physical Parameters
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	[-]		
PET_VEG_CORR	vegetation multiplier for PET	[-]		

Table A.6: Available global parameters in Raven.

Name	Definition	Units	Range	
ADIABATIC_LAPSE	adiabatic temperature lapse rate	°C/km	0-7	Physical Parameters
WET_ADIABATIC_LAPSE	wet adiabatic temperature lapse rate	°C/km	0-7	
PRECIP_LAPSE	precipitation lapse rate for orographic correction	mm/d/km	0-100	
RAINSNOW_TEMP	rain/snow halfway transition temperature	°C	-1.0-1.0	
RAINSNOW_DELTA	range of rain-snow transition zone (about RAINSNOW_TEMP)	°C	0-4	
SNOW_SWI	water saturation fraction of snow	0..1	0.04-0.07	
SNOW_SWI_MIN	minimum water saturation fraction of snow	0..1	0.04-0.05	
SNOW_SWI_MAX	maximum water saturation fraction of snow	0..1	0.05-0.15	
SNOW_TEMPERATURE	default snow temperature if not explicitly simultated	°C	-2.0-0.0	
SNOW_ROUGHNESS	roughness height of snow	mm	0-5.0	
AVG_ANNUAL_SNOW	avg annual snow as SWE	mm	0-100	
AVG_ANNUAL_RUNOFF	avg annual runoff from basin	mm	0-1000	
MAX_SNOW_ALBEDO	albedo of fresh snow	0..1	0.95	
MIN_SNOW_ALBEDO	very old snow/glacier albedo	0..1	0.3	
BARE_GROUND_ALBEDO	bare ground albedo	0..1	0.1-0.4	
MAX_REACH_SEGLENGTH	maximum reach segment length	km		Conceptual/Numerical Model Parameters
MAX_SWE_SURFACE	maximum SWE in surface snow layer (SNOBAL_TWO_LAYER)	mm		
AIRSNOW_COEFF	air/snow heat transfer coefficient	1/d		
UBC_GW_SPLIT	UBC groundwater split parameter	0..1	0.4	
UBC_EXPOSURE_FACT	UBC Sun exposure factor of forested areas	0..1		
UBC_CLOUD_PENET	UBC Fraction of solar radiation penetrating cloud cover	0..1		
UBC_LW_FOREST_FACT	UBC temperature factor to estimate LW radiation in forests	mm/d/K		
UBC_FLASH_PONDING	UBC ponding threshold for flash factor	mm		
UBC_ALBASE	albedo exponential decay threshold value	-	0.65	
UBC_ALBREC	albedo decay constant	1/d	0.9	
UBC_ALBSNW	daily snowfall required to bring albedo to that of new snow	mm	15	
ALB_DECAY_COLD	linear albedo decay rate for cold conditions	1/d	0.008	
ALB_DECAY_MELT	linear albedo decay rate for melting conditions	1/d	0.12	
SNOWFALL_ALBTHRESH	threshhold snowfall rate to refresh albedo to fresh snow	mm/d	10	
UBC_MAX_CUM_MELT	estimate of maximum annual snowmelt	mm	4000	
SWI_REDUCT_COEFF	rate of SWI reduction with increasing cumulative melt	1/mm	0.02	
MOHYSE_PET_coeff	PET coefficient for MOHYSE PET algorithm	-	1.0	
ASSIMILATION_FACT	degree of assimilation (=0 for none, =1 for full insertion)	0..1	1	Alg Params
ASSIM_TIME_DECAY	controls degree of assimilation after observations end	1/d	0.2	
ASSIM_UPSTREAM_DECAY	controls degree of assimilation upstream of observation	1/km	0.1	
RESERVOIR_RELAX	relax. factor for reservoir simulation of target stage/flow	0..1	0.4	

Table A.7: Available geochemical parameters in Raven.

Name	Definition	Units	Reasonable range
DECAY_COEFF	linear decay coefficient	1/d	>0
TRANSFORM_COEFF	transformation coefficient	1/d (mg/L)^n/d for n$\neq$1	>0
TRANSFORM_N	non-linear transformation exponent	-	>0
EQFIXED_RATIO	fixed equilibrium ratio (a in $C_1=aC_2$)	-	>0
EQUIL_COEFF	linear equilibrium coefficient (k in $k(C_1-aC_2)$)	1/d	>0
SORPT_COEFF	linear sorption coefficient, K_d	L/kg	>0

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
# ------------------------------------------
: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, 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,[NONE],0.0,0.0
  102,10,145, 43,-80,1,URBAN   ,BROADLEAF, ALL_SAND,SAND_AQ,[NONE],0.0,0.0
  103,10,143, 43,-80,2,FORESTED,BROADLEAF,     TILL,SAND_AQ,[NONE],0.0,0.0
  104,10,147, 43,-80,2,FORESTED,BROADLEAF,     TILL,SAND_AQ,[NONE],0.0,0.0
:EndHRUs
:HRUGroup ForestedHRUs
 101,103,104
:EndHRUGroup
:RedirectToFile Reservoirs.rvh
:RedirectToFile SubBasinParams.rvh
```

Due to the length of each line above, the `:Attributes` and `:Units` headers of the `:HRUs` command are shown using line wrapping (denoted by '...').

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). Can be non-unique. This value is used for labelling output.
- `downstream_ID` - The ID of the subbasin (or conduit) that receives the outflow from this basin. If the drainage for this subbasin leaves the simulated 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 simulated 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
# NOTE: the '...' above denotes line wrapping
```

A table of data used to specify an array of HRUs within the subbasins defined in the `:SubBasins` command block. The HRU-specific parameters are defined as follows:

- `ID` - A positive integer **unique to this HRU**. Used to refer to the HRU in other parts of the input file.
- `AREA` - the total HRU area (in km^2)
- `ELEVATION` - the mean HRU elevation (in m.a.s.l)
- `LATITUDE` - Latitude of the HRU centroid (in decimal degrees). Used primarily for interpolation and estimation of solar radiation.
- `LONGITUDE` - Longitude of the HRU centroid (in decimal degrees). Used primarily for interpolation and estimation of solar radiation.
- `BASIN_ID` - the ID of the basin in which the HRU is located (as defined in the `:SubBasins` command `ID` column)
- `LAND_USE_CLASS` - the representative land use class of the HRU (defined in the .rvp file)
- `VEG_CLASS` - the representative vegetation class of the HRU (defined in the .rvp file)

- SOIL_PROFILE - the representative soil profile of the HRU (defined in the .rvp file)
- AQUIFER_PROFILE - unused. use [NONE]
- TERRAIN_CLASS - (optional) the representative terrain class of the HRU (defined in the .rvp file). If terrain classes not used use [NONE]
- SLOPE - mean HRU slope, in degrees from horizontal
- ASPECT - mean aspect (in degrees counter-clockwise from north - i.e., a western aspect would be 90 °). Note: this convention is opposite that used by most GIS terrain analysis tools.

If terrain classes or aquifer profiles are not used in the model (as is common), the flag [NONE] goes in the place of the class specifier.

A.3.2 Optional Commands

```
:Conduits
  :Attributes,   ID, NAME, DOWNSTREAM_ID, XSECT, REACH_LENGTH, GAUGED,
  :Units     , none, none,          none,    none,          km,
none,
  {ID,name,downstream_ID,xsect,reach_length,gauged}x[number of conduits]
:EndConduits
```

This command is nearly identical to the :SubBasins-:EndSubBasins command block, and also indicates flow connectivity. However, it is used to represent the subsurface sewer network, aqueducts, or subsurface rivers in karst. Unlike subbasins, conduits do not have corresponding HRUs or areal extent. Rather, the only process simulated in a conduit connection is in-reach routing. The indexing system is the same as that used for subbasins, i.e., a conduit and subbasin cannot share the same ID. Conduits can outflow to other conduits or to other subbasins, but is not allowed to form a cyclical network. The xsect will typically be a :CircularConduit cross section or similar.

```
: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
:EndSubBasinProperties
```

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 D.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.

```
:PopulateHRUGroup [HRUgroup] With [con_base] [condition] [con_data]
```

An alternative to the :HRUGroup command which automatically populates the HRU group based upon certain criteria. The cond_base command indicates the basis for the criterion, one of (HRUS, LANDUSE, VEGETATION, or ELEVATION). The condition indicates the means of evaluating the criterion, one of (NOTWITHIN, WITHIN_SBGROUP, BETWEEN, EQUALS, NOTEQUALS). The con_data is dependent upon the condition. For the NOTWITHIN condition, the condition data is another HRU group name and the criterion must be HRUS. For the BETWEEN condition, the condition data is a range of elevations, and the only currently valid criterion basis is the elevation. For the EQUALS and NOTEQUALS conditions, the vegetation or land use names are specified, to group HRUs based upon class membership (or non-membership). For the WITHIN_SBGROUP condition, the name of a valid subbasin group is used. For example, the following commands are valid:

```
:PopulateHRUGroup CroplandHRUs    With LANDUSE EQUALS CROPLAND
:PopulateHRUGroup NonCroplandHRUs With LANDUSE NOTEQUALS CROPLAND
:PopulateHRUGroup BroadleafHRUs   With VEGETATION EQUALS BROADLEAF
:PopulateHRUGroup NotRock         With HRUS NOTWITHIN RockHRUGroup
:PopulateHRUGroup LowBand         With ELEVATION BETWEEN  0 500
:PopulateHRUGroup GrandRiverHRUs  With HRUS WITHIN_SBGROUP GrandBasins
```

```
:IntersectHRUGroups [HRUGroup] From [HRUGroup1] And [HRUGroup2]
```

This command populates the HRU group HRUGroup with any HRUs shared by HRUGroup1 and HRUGroup2. These two groups need to have already been defined and populated before this command is called.

```
:MergeHRUGroups [HRUGroup] From [HRUGroup1] [HRUGroup2] ... [HRUGroupN]
```

This command populates the HRU group HRUGroup with any HRUs that exist in the list of HRU groups HRUGroup1 to HRUGroupN. It will not duplicate group members if they are present in multiple groups. The list of groups need to have already been defined and populated before this command is called.

```
:DisableHRUGroup [group_name]
```

This command disables the HRU group specified, and has to be included after the HRU group has been defined (in the .rvi file via :DefineHRUGroup) or populated (in the .rvh file via :HRUGroup). All disabled HRUs are not included in the simulation, and all subbasins comprised entirely of disabled HRUs are likewise not simulated.

```
:SubBasinGroup [group_name]
  2118, 3024, 3056, 4567
:EndSubBasinGroup
```

Subbasin groups are used for a number of reasons: to generate custom output only for a select set of basins (or organize/aggregate output for multiple sets of basins) or to control parameterization of a large number of basins (for instance, to set or adjust the Manning's n parameter for a set of geomorphologically similar river reaches). Individual subbasin members of the group are specified with their ID numbers (as defined

in the :SubBasins command), separated by commas. Unlike the :HRUGroup command, ranges are not supported.

```
:PopulateSubBasinGroup [SBgroup] With [con_base] [condition] [con_data]
```

An alternative to the :SubBasinGroup command which automatically populates the subbasin group based upon certain criteria. The cond_base command indicates the basis for the criterion, currently only SUBBASINS. The condition indicates the means of evaluating the criterion, either WITHIN, NOTWITHIN, UPSTREAM_OF or DOWNSTREAM_OF. The con_data is dependent upon the condition. For the NOTWITHIN condition, the condition data is another subbasin group name and the criterion must be SUBBASINS. For the UPSTREAM_OF and DOWNSTREAM_OF, the con_data indicates a subbasin ID. For example, the following commands are valid:

```
:PopulateSubBasinGroup NotErie With SUBBASINS NOTWITHIN Erie
:PopulateSubBasinGroup UpstreamOfBasin2 With SUBBASINS UPSTREAM_OF 2
:PopulateSubBasinGroup DownFromBasin35 With SUBBASINS DOWNSTREAM_OF 35
```

The UPSTREAM_OF query includes the subbasin indicated (i.e., it is all subbasins upstream of the indicated subbasin outlet), while the DOWNSTREAM_OF does not (i.e., it is all subbasins directly downstream of the indicated subbasin outlet). Note that this command can be called iteratively to add subbasins to an existing subbasin group.

```
:IntersectSubBasinGroups [SBgroup] From [SBgroup1] And [condition] [SBgroup2]
```

An additional method logic for building subbasin groups which intersects two subbasin groups. The condition indicates the means of intersection, either WITHIN or NOTWITHIN. In this command, both conditions must be satisfied to add the given subbasin to the new group (i.e. if NOTWITHIN is used, a subbasin must exist in SBGroup1 and explicitly not exist in SBGroup2 to be added to the new subbasin group).

```
:MergeSubBasinGroups [SBgroup] From [SBgroup1] ... [SBgroupN]
```

This command populates the subbasin group SBgroup with any subbasins that exist in the list of subbasin groups SBgroup1 to SBgroupN. It will not duplicate group members if they are present in multiple groups. The list of groups need to have already been defined and populated before this command is called.

```
:DisableSubBasinGroup [group_name]
```

This command disables the subbasin group specified, and has to be included after the subbasin group has been defined and populated (in the .rvh file via :SubBasinGroup). All disabled subbasins are not included in the simulation, and all HRUs within these subbasins are also disabled and not simulated.

Advice

This command may be used to simulate or calibrate only a subset of the model domain. Subdomains in the middle of a watershed may also be simulated if all inflows to the subbasins are proscribed using the :BasinInflowHydrograph command.

```
:GaugedSubBasinGroup [group_name]
```

This command overrides the "gauged" status of subbasins that were indicated in the :SubBasins-:EndSubBasins command block. All subbasins within the subbasin group group_name are set as "gauged" basins where hydrograph and other output is generated. All subbasins not in this group have their basin-specific output disabled.

```
:SBGroupPropertyOverride [group_name] [parameter_name] [value]
```

This command simultaneously sets the parameter values for all subbasins within a group. The group_name refers to a subbasin group created using the :SubBasinGroup command, the parameter_name is one of the parameters named in table D.3, and value is the value of the parameter being specified. Units have to be consistent with those in table D.3.

```
:SBGroupPropertyMultiplier [group_name] [parameter_name] [mult]
```

This command is typically used for calibration, allowing the user to simultaneously adjust the parameter values for a large number of subbasins simultaneously. The group_name refers to a subbasin group created using the :SubBasinGroup command, the parameter_name is one of the parameters named in table D.3, and mult is the multiplier used to adjust the parameters (base values should already have been specified using the :SubBasinProperties command; this multiplier will not work for parameters which do not appear in .rvh files and are instead automatically calculated).

```
:SBGroupOverrideWeights [group_name] [parameter_name] [w1] ... [wN]
:SBGroupOverrideWeights [group_name] [parameter_name] [u1] ... [u{N-1}]
```

This command is typically used for calibration, allowing the user to specify blended forcing weights (currently for either potential melt or potential evapotranspiration) for a specific subbasin group, overriding the global blended forcing weights specified in the .rvi file. The group_name refers to a subbasin group created using the :SubBasinGroup command, the parameter_name is one of the parameters PET-BlendedWeights or POTMELTBlendedWeights, and either the user can either specify N weights or $N-1$ independent uniform numbers (weight-generating parameters) used to calculate weights (see :ProcessGroup command for more information on calculating N weights from $N-1$ independent uniform numbers in blended models).

```
:RedirectToFile [filename]
```

This treats the contents of file "filename" as if they were simply inserted into the .rvh file at the location of the :RedirectToFile command. This is useful for storing individual sets of commands in an organized format (e.g., redirecting to a separate file with a number of :Reservoir blocks). If no path is specified, the filename must be reported relative to the working directory. This command must be provided in the main model .rvh file and not within nested files. Note that this command cannot work within data blocks (e.g., a the entire :SubBasins-:EndSubBasins block would have to be in a single file, not just the tabular data in that block).

A.3.3 Reservoirs and Lakes

```
# Man-made reservoir
:Reservoir [name]
  :SubBasinID [SBID]
  :HRUID [HRUID] # optional
  :StageRelations
```

```
    [number of points (N)]
    h_1, Q_1, V_1, A_1, {U_1}
    h_2, Q_2, V_2, A_2, {U_2}
    ...
    h_N, Q_N, V_N, A_N, {U_N}
  :EndStageRelations
  :MaxCapacity {capacity, in m^3} # optional
  :SeepageParameters [K_seep] [h_ref] # optional
  :OutflowControlStructure #optional
  # ...
  :EndOutflowControlStructure
:EndReservoir
```

This command creates a reservoir at the outlet of the subbasin referenced by `SBID` characterized by `N` points on the indicated stage-discharge, stage-volume, and stage-area curves. Here, stage (`h_i`) is in meters, flow (`Q_i`) and underflow (`U_i`, optional) are in m^3/s, volume stage (`V_i`) is in m^3, and area stage (`A_i`) is in m^2. The stage increments can be unevenly spaced but must be increasing from `h_1` to `h_N`. Area and volume both must be monotonically increasing with increasing stage. For numerical stability, it is expected that changes in volume with stage increments are approximately equal to the area times the change in stage (i.e., a useful test is to compare ΔV to $A\Delta h$).

Evaporation from the reservoir surface is 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). If no HRU ID is provided, evaporation from the reservoir is presumed negligible. The reservoir volume, outflow, and net precipitation to the reservoir surface are obtained by interpolating their value from the specified stage-discharge $Q(h)$, stage-area $A(h)$, and stage-volume $V(h)$ relations, defined here by `N` points along the rating curves. The underflow relation $Q_u(h)$ is optional, and is assumed to be zero if omitted; if included, the total flow from the reservoir will be $Q(h) + Q_u(h)$. Note that the minimum stage supplied in the `:StageRelations` should be the minimum expected stage (usually the bottom of the reservoir), and the maximum should be above the expected maximum stage. See figure 4.1b for additional clarification of terms. Note that multiple operational constraints upon stage and flow for reservoirs may be specified as time series in the .rvt file using commands such as the `:ReservoirMaxStage` command. The optional `:MaxCapacity` item the maximum storage capacity of the reservoir in m^3, but is only used to inform the `:ReservoirDemandAllocation` operation; it does not constrain the stage or outflow from the reservoir - this should be handled via maximum stage constraints or via the stage-discharge curve of the reservoir.

Groundwater seepage parameters `K_seep` ($m^3/s/m$) and `h_ref` (m) can be used to represent groundwater seepage from the reservoir, where the losses are calculated as $Q_{loss} = K_{seep} \cdot (h - h_{ref})$. If the reference groundwater head (h_ref) is larger than the reservoir stage, the reservoir gains water, otherwise it loses water. By default, K_{seep} is zero, and no groundwater losses/gains are considered.

The details of the `:OutflowControlStructure` commands are outlined below in section A.3.4. Each reservoir may have multiple outflow control structures governed by very general rules. If controlling outflow exclusively with control structures, the discharge values in the `:StageRelations` table should all be set to zero. Otherwise, the outflows from the control structure will be in addition to the primary outflow indicated by the `:StageRelations` relationship.

```
# Lake-like reservoir
:Reservoir [name]
  :SubBasinID [SBID]
```

```
  :HRUID [HRUID]
  :WeirCoefficient [C]
  :CrestWidth [width [m]]
  :MaxDepth [depth [m]]
  :LakeArea [area [m2]]
  :AbsoluteCrestHeight [elevation [masl]] {optional}
:EndReservoir
```

This command creates a **lake-like reservoir** at the outlet of the subbasin referenced by `SBID`, and is the preferred option for natural reservoirs. Evaporation from the reservoir surface are obtained from the HRU referenced by `HRUID`, as with the above `:Reservoir` command. Here, the discharge-stage, volume-stage, and area-stage relations are generated using the following overflow weir formulae for a prismatic lake:

$$\begin{aligned} Q(h) &= \frac{2}{3}\sqrt{2g}C \cdot L \cdot s^{3/2} \\ A(h) &= A \\ V(h) &= A \cdot (s + D) \end{aligned}$$

where s is the stage measured with reference to the crest height (which can be negative), D is the specified maximum lake depth (`:MaxDepth` [m]), g is the gravitational constant [m/s^2], C is the weir coefficient (`:WeirCoefficient`), A is the constant lake areas (`:LakeArea`), [m^2], and L is the crest width (`:CrestWidth`, [m]). See figure 4.1a for additional clarification of terms. Typically the weir coefficient is held fixed at a value of about 0.6, and the crest width is calibrated to represent the unknown crest width and overflow resistance. `:AbsoluteCrestHeight` may be supplied to reference stages to real lake stage; by default stage is with reference to the crest height, i.e., a zero stage would be just at the crest. Note that when many reservoirs and lakes are supplied, they would usually be kept in one or more separate files via the `:RedirectToFile` command. You can override the calculated area and volume relationships using the `:AreaStageRelation` and `:VolumeStageRelation` commands, respectively.

```
:MinStageConstraintDominant
```

Used within the `:Reservoir`-`:EndReservoir` block, this command is used to override the default ordering of outflow constraints for reservoirs. With this command, the minimum stage constraint defined by `:ReservoirMinStage` will override minimum flow, overrriden flow, and maximum flow constraints, changing the order of constraints indicated in section 4.3 but only in this reservoir. This will likewise override minimum demand flow calculated from the use of the `:ReservoirDownstreamDemand` command, such that the reservoir will not meet downstream demands if the stage is at minimum.

```
:DemandMultiplier [value]
```

Used within the `:Reservoir`-`:EndReservoir` block, this command is used to modify the percentage of downstream irrigation demand met by this reservoir. The `:ReservoirDownstreamDemand` command is used to allocate irrigation demand from downstream subbasins of a reservoir, thus increasing the minimum flow from that reservoir. The calculated minimum flow is multiplied by this demand multiplier. If set to 1.0, the reservoir completely respects the constraints from the downstream demand calculations. If set to 0.0, the reservoir will not be constrained in any way by downstream irrigation demand. This command may be used to help evaluate a range of water management strategies.

```
:LakebedThickness [value]
```

Used within the :Reservoir-:EndReservoir block, this command specifies the thickness of the lake bed beneath the lake or reservoir, in meters. This quantity is only used for thermal and contaminant transport, where mass or energy may be exchanged with this stationary soil layer. If not specified, the lake bed thickness is set to 2m.

```
:LakebedThermalConductivity [value]
```

Used within the :Reservoir-:EndReservoir block, this command specifies the lake bed thermal conductivity, in units of [MJ/m/K/d]. This quantity is only used for thermal transport, to control conductive heat exchange between the lake and its bed materials. If not specified, the lake bed thickness is set to zero, such that no conductive heat exchange occurs

```
:LakeConvectionCoeff [value]
```

Used within the :Reservoir-:EndReservoir block, this command specifies the constant lake surface convection coefficient, in units of [MJ/m^2/K/d]. This quantity is only used for thermal transport, to control conductive heat exchange between the lake surface and atmosphere. It defaults to a value of zero, such that convective heat exchange is ignored.

```
:DZTRReservoirModel
  :MaximumStorage            [Vmax]
  :MaximumChannelDischarge   [Qmax]
  :MonthlyMaxStorage         J F M A M J J A S O N D
  :MonthlyNormalStorage      J F M A M J J A S O N D
  :MonthlyCriticalStorage    J F M A M J J A S O N D
  :MonthlyMaxDischarge       J F M A M J J A S O N D
  :MonthlyNormalDischarge    J F M A M J J A S O N D
  :MonthlyCriticalDischarge J F M A M J J A S O N D
:EndDZTRReservoirModel
```

Used within the :Reservoir-:EndReservoir block, this command overrides the default stage-discharge relationship used to determine reservoir outflow with the time-variable volume-discharge relationship that is defined in Yassin et al. (2019). Historical volume-storage observations can be used to estimate these parameters to emulate historical (but unknown) reservoir outflow operational rules. All monthly storages (e.g., :MonthlyMaxStorage) are in m^3 and all flows (e.g., :MonthlyMaxDischarge) are in m^3/s; each row gets 12 monthly values. The order of the above commands must be respected and no comments are allowed between commands in this block. The details of operation are to be found in the Yassin et al. (2019) paper. With this command, any stage-discharge curve indicated in the command structure is ignored.

```
:VolumeStageRelation
  [number of points (N)]
  h_1  V_1
  h_2  V_2
  ...
  h_N  V_N
:EndVolumeStageRelation
```

Used within the :Reservoir-:EndReservoir block, this command is used to override Raven's default calculation of the volume-stage relationship for a lake-like reservoir. The relationship is specified

using N (h_i,V_i) pairs, which don't need to be evenly spaced. Stage is in units of meters, and volume is in cubic metres. Volumes must monotonically increase with increasing stage.

```
:AreaStageRelation
  [number of points (N)]
  h_1  A_1
  h_2  A_2
  ...
  h_N  A_N
:EndVolumeStageRelation
```

Used within the :Reservoir-:EndReservoir block, this command is used to override Raven's default calculation of the surface area-stage relationship for a lake-like reservoir. The relationship is specified using N (h_i,V_i) pairs, which don't need to be evenly spaced. Stage is in units of meters, and lake surface area is in square metres. Areas must monotonically increase with increasing stage.

A.3.4 Outflow Control Structures

Multiple outflow control structures may be simulated with user-specified operating regimes for any :Reservoir entity.

```
:OutflowControlStructure
  :TargetSubbasin [SBID] #optional
  :DownstreamReferenceElevation [elev] #optional

  # Outflow relations: -------------------------
  :StageDischargeTable [curve_name1]
    [number of points(N)]
    {h_n  Q_n} x N
  :EndStageDischargeTable
  :StageDischargeTable [curve_name2]
    [number of points(N)]
    {h_n  Q_n} x N
  :EndStageDischargeTable
  :BasicWeir [curve_name3] [elev] [crestwidth] [coeff]

  # Operating Regime Definitions---------------
  :OperatingRegime [regime_name]
    :UseCurve [curve_name]
    :Condition [condition definition]
    :Constraint [constraint definition]
  :EndOperatingRegime
:EndOutflowControlStructure
```

The :OutflowControlStructure command describes an outflow control structure which releases water from a reservoir to a target subbasin (usually the subbasin directly downstream of the reservoir). This command must be located inside of a :Reservoir-:EndReservoir command block. Each outflow control structure is defined by (1) one or more outflow relations (2) one or more operating regimes determining under what conditions each outflow relation is applied. If no conditions are valid, then outflow from the control structure is zero (the default operating regime). The optional element :Target-

Subbasin is used to specify the recipient subbasin of the outflow. If not present, the target subbasin is the downstream basin, as indicated in the :SubBasins block. The optional element :Downstream-ReferenceElevation [m] is the reference elevation of the thalweg of the recipient subbasin reach. This elevation is needed if regime constraints are conditional upon downstream water level, calculated as the sum of the water depth in the channel plus the reference elevation.

Outflow relations

Raven supports five basin outflow relations:

```
:StageDischargeTable [name]
    [number of points(N)]
    {h_n  Q_n} x N
:EndStageDischargeTable
```

A basic stage-discharge relationship described by a lookup table of discharge (Q_n, in m^3/s) with respect to stage (h_n, in m). The flows must monotonically increase with stage. Turning off the flow above a certain stage is accomplished using constraints. The name must be unique, as it links this relationship to specific regimes using the :UseCurve command described below.

```
:BasicWeir [name] [elev] [width] [coeff]
```

A simple weir or overflow gate with crest elevation h_{crest} (elev, in m), width W (width, in m), discharge coefficient C_d (coeff, dimensionless). The outflow from the structure is calculated as:

$$Q_{out} = \frac{2}{3} C_d \sqrt{2g} \cdot W \cdot (\max(h - h_{crest}, 0.0))^{3/2}$$

where g is the gravitational constant. For a sharp-crested weir, the discharge coefficient should be 0.4.

```
:SluiceGate [name] [elev] [width] [r_height] [coeff] [NG]
```

A set of broad-crested sluice gates with bottom elevation h_{crest} (elev, in m), width W (width, in m), raised height h_r (r_height, in m), discharge coefficient C_d (coeff, dimensionless, typically 0.5-0.7), and number of gates N_G (NG). The outflow from the structure is calculated under free-flowing conditions as:

$$Q_{out} = N_G \cdot C_d \cdot \sqrt{2g} \cdot W \cdot h_r \left(\max(h - h_{crest}, 0.0)\right)^{1/2}$$

where g is the gravitational constant. It is assumed that $h_r < h - h_{crest}$.

```
:Orifice [name] [elev] [diameter] [coeff] [NO]
```

A set of circular orifices with bottom elevation h_b (elev, in m), diameter D (diameter, in m), discharge coefficient C_d (coeff, dimensionless, typically 0.5-0.7), and number of orifice openings N_O (NO). The outflow from the structure is calculated under free-flowing conditions as:

$$Q_{out} = N_O \cdot C_d \sqrt{2g} \cdot \frac{\pi D^2}{4} \cdot (h - h_b)^{1/2}$$

where g is the gravitational constant. It is assumed that orifices are fully submerged at all times.

```
:BasicPump [name] [flow] [on_stage] [off_stage]
```

A pump which operates when the reservoir stage exceeds the stage `on_stage` [m] and turns off when the stage falls below the stage `off_stage` [m], where `on_stage>off_stage`. The second threshold is to stop the pump from repeatedly turning on and off. The discharge is equal to the specified discharge rate (`flow`, in m^3/s) within this stage range. The basic pump does not have flow ramping, but this may be achieved with flow operating constraints, described below.

Operating Regimes

```
:OperatingRegime [regime_name]
  :UseCurve [curve_name]
  :Condition [condition definition #1]
  :Condition [condition definition #2]
  :Constraint [constraint definition #1]
  :Constraint [constraint definition #2]
:EndOperatingRegime
```

Each operating regime defines conditions under which a given outflow relation is used for a specific control structure. Typical reservoirs would have different regimes based upon season and stage, where different outflow strategies are used under high stage conditions than low stage conditions. Regime rules can therefore be linked to stage, flow, time of year, or specific dates. The `:UseCurve` specifies the name of the outflow relation used (e.g., a `:StageDischargeTable` or `:Orifice` relation within the same `:OutflowControlStructure` command block). These are accompanied by lists of conditions under which the operating regime exists, and lists of constraints to put on the outflow relation. If multiple operating regimes are present, the first in the list whose conditions are satisfied will be applied (i.e., the first regime will be highest priority).

```
:Condition [variable] [comparison] [value] {val2}
# or
:Condition [variable] [comparison] [value] {val2} {IN_BASIN SBID}
```

A condition indicating the operating regime should be active. ALL conditions must be met in order for the operating regime to be active. The `variable` term must be one of:

- `STAGE` - the stage of the reservoir, in m
- `STAGE_CHANGE` - the rate of change in stage of the reservoir, in m/d
- `FLOW` - the total outflow from a reservoir or stream, in m^3/s
- `RIVER_DEPTH` - the depth of water in a specified subbasin
- `DAY_OF_YEAR` - the Julian date
- `DATE` - a date in yyyy-mm-dd format

if the `IN_BASIN` clause is not used, `STAGE` and `FLOW` refer to the stage and total discharge from this reservoir. If not, this indicates that the stage, flow, or river depth is evaluated in a basin other than the one containing the reservoir. The `comparison` string is one of

- `IS_LESS_THAN`
- `IS_GREATER_THAN`
- `IS_BETWEEN` - this requires that `val2` is present

The values of `value` and `val2` (for the `IS_BETWEEN` conditional) are the basis for comparison. Example conditions:

```
:Condition STAGE IS_LESS_THAN 373.3
:Condition STAGE IS_LESS_THAN 273 IN_BASIN 43
:Condition FLOW IS_GREATER_THAN 32.3
:Condition DAY_OF_YEAR IS_BETWEEN 120 242 #May-Aug
:Condition DAY_OF_YEAR IS_BETWEEN 243 119 #Sep-Apr
# note the IS_BETWEEN works with date wrapping around Jan 1
:Condition RIVER_DEPTH IS_LESS_THAN 484.5 IN_BASIN 5
:Condition DATE IS_BETWEEN 2010-07-01 2012-06-31
```

```
:Constraint [variable] [comparison] [value] {val2}
```

Constraints are used to modify the flow directly calculated from the outflow relation. Here, the `variable` can either be `FLOW` or `FLOW_DELTA`, the comparison is one of `IS_LESS_THAN`, `IS_GREATER_THAN`, or `IS_BETWEEN` and `value` and `val2` are the basis of the comparison; for flow, the units of these values are m^3/s, and for flow ramping, the units are $m^3/s/d$. The calculated flow will be modified to satisfy the constraints. If multiple constraints are supplied, the order determines the priority of constraints, with the first constraint being dominant over the second, and so on. Example constraints:

```
:Constraint FLOW IS_LESS_THAN 100
:Constraint FLOW_DELTA IS_LESS_THAN 12
:Constraint FLOW_DELTA IS_BETWEEN -12 12
```

The last constraint indicates that the flow cannot be increased or decreased by more than 12 m^3/s over the course of a day, which would stabilize flows.

Example Control Structures

```
:OutflowControlStructure
  :SluiceGate 1open 423.2 2.0 0.5 0.6 1
  :SluiceGate 2open 423.2 2.0 0.5 0.6 1
  :SluiceGate 3open 423.2 2.0 0.5 0.6 1
  :BasicWeir  overflow 430 4.0 0.4

  # Operating Regime Definitions----------------
  :OperatingRegime winter
    :UseCurve 1open
    :Condition DAY_OF_YEAR IS_BETWEEN 304 90 #Nov-Mar
  :EndOperatingRegime
  :OperatingRegime summer
    :UseCurve 2open
    :Condition DAY_OF_YEAR IS_BETWEEN 91 303 #Apr-Oct
    :Condition STAGE IS_LESS_THAN 426
  :EndOperatingRegime
  :OperatingRegime summer_high
    :UseCurve 3open
    :Condition DAY_OF_YEAR IS_BETWEEN 91 303 #Apr-Oct
```

```
    :Condition STAGE IS_GREATER_THAN 426
  :EndOperatingRegime
  :OperatingRegime overflowing
    :UseCurve overflow
    :Condition STAGE IS_GREATER_THAN 430
  :EndOperatingRegime
:EndOutflowControlStructure
```

The first three regimes are exclusive - the conditions for each cannot be simultaneously satisfied, so the reservoir will operate with either 1, 2, or 3 sluice gates open based upon time of year and stage. Upon very high stage conditions, the overflowing regime can activate (in addition to the outflow from 1 sluice in the winter or 3 in the summer).

The order of regimes does not matter, but the user must be careful to not allow conditions that would lead to duplication of outflows from the same structure (e.g., two regimes representing the same structure with one conditions of 'greater than 300' and one condition of 'greater than 301'). Raven cannot identify these duplicate outflows.

A.4 Time Series Input file (.rvt)

The time series input file is used to store time series of forcing functions (precipitation, temperature, etc.). It also may include additional commands for handling irrigation and diversions (i.e., any control systems). An .rvt file is structured as follows:

```
#-----------------------------------------
# Raven Time Series Input file
#-----------------------------------------
:Gauge Stratford MOE (ID:6148105)
  :Latitude   43.37250
  :Longitude -80.55360
  :Elevation 53
  :RedirectToFile StratfordMOEData.rvt
:EndGauge
:Gauge WaterlooWeatherStation
  :Latitude   43.37
  :Longitude -80.55
  :Elevation 57
  :RedirectToFile WaterlooWeatherStationData.rvt
:EndGauge
:RedirectToFile UpstreamInflow.rvt
:RedirectToFile LandCoverChange.rvt
:RedirectToFile ObservedHydrograph.rvt
```

Note that standard practice is to have a single master `modelname.rvt` file that 'points to' a number of other .rvt files which contain unique data sets, i.e., an individual .rvt file for meteorological forcing data at a single meteorological gauge, another for observed stream flow at a stream gauge, and another reporting pumping from one reservoir. The 'pointing' is done using the `:RedirectToFile` command as shown in the above example file. All of the redirected files are treated as if their contents have been inserted into the master .rvt file.

> Please note that all of Raven's inputs and internal calculations use the standard proleptic Gregorian calendar with leap years included by default. If input data ignores leap years, Raven can handle some alternate calendar types by using the `:Calendar` command.
> All hourly data referenced to GMT (Greenwich Mean Time) or another time zone outside the watershed will need to be shifted to the local time zone to be consistent with the solar calculations, which assume solar noon is at 12:00PM. Raven does not perform time zone shifts automatically.

A.4.1 Meteorological 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.4.7 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 [forcing type] {units}
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  v_1
  v_2
  v_3
  ...
  v_N
:EndData
```

where here, `v_i` are the i^{th} time series values and the `forcing_type` term is one of the forcings listed in table D.2 (e.g., `PRECIP`, `TEMP_MIN`. etc.). `N` is the total number of data points provided, evenly spaced at the specified time interval. Note that this is the default format for most of the regularly spaced time series commands in Raven.

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. Multiple data points may be included on a single line, though the single-column format makes this easier to use in other program utilities. 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 D.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)] [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 D.2), and the units tags should be consistent with the actual desired units in table D.2.

Again, note that the time interval must be specified as a double, and cannot be specified using a format of 00:00:00. Raven will not perform units conversions for you if alternate units are specified in the :Units header.

Other additional terms may be associated with each gauge, contained between the :Gauge-:EndGauge brackets:

```
:Elevation [elevation]
```

The elevation of the gauge, typically in meters above mean sea level. This is used both in interpolation and in orographic correction of gauge data when mapped to HRUs at different elevations. Must be between the :Gauge–:EndGauge brackets

```
:Latitude [latitude]
```

The latitude of the gauge, in degrees. This is used in interpolation of gauged forcings.

```
:Longitude [longitude]
```

The longitude of the gauge, in degrees. This is used in interpolation of gauged forcings.

```
:MeasurementHeight [height]
```

The height of the gauge relative to the ground surface, in meters. This is particularly important for wind velocity measurements to calculate (e.g.) atmospheric conductance and other parameters dependent upon vertical wind speed distribution, but may typically be ignored in temperature-only gauges.

```
: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 at a gauge in separate files. If no path is specified, the filename must be reported relative to the working directory. This command must be provided in the main model .rvt file and not within nested files. 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).

```
: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 at gauges prone to undercatch or otherwise not expected to be representative of local conditions. Must be between the :Gauge–:EndGauge brackets (or between :GriddedForcings–:EndGriddedForcings brackets for gridded rainfall data).

```
: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 at gauges prone to undercatch or otherwise not expected to be representative of local conditions. Must be between the :Gauge–:EndGauge brackets (or between :GriddedForcings–:EndGriddedForcings brackets for gridded snowfall data).

```
:TemperatureCorrection [value]
```

An additive correction (hopefully near 0.0) in °C, applied to reported temperatures at this gauge (specifically, TEMP_AVE, TEMP_DAILY_MIN, TEMP_DAILY_MAX, and TEMP_DAILY_MIN); may be used in bias correction of measured temperatures. Must be between the :Gauge–:EndGauge brackets (or between :GriddedForcings–:EndGriddedForcings brackets for gridded temperature data).

```
: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. Must be between the :Gauge–:EndGauge brackets. Predominantly used for the PET_FROMMONTHLY PET estimation method, not otherwise needed.

```
:MonthlyMinTemperature [J F M A M J J A S O N D]
:MonthlyMaxTemperature [J F M A M J J A S O N D]
```

A list of 12 representative monthly minimum and maximum temperatures at the gauge, from Jan to Dec, in °C. Must be between the :Gauge–:EndGauge brackets. Predominantly used for the PET_HARGREAVES PET estimation method, not otherwise needed.

```
: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. Must be between the :Gauge–:EndGauge brackets. Predominantly used for the PET_FROMMONTHLY PET estimation method, not otherwise needed.

```
: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 PET_MONTHLY_FACTOR estimation routine, not otherwise needed. Must be between the :Gauge–:EndGauge brackets.

```
:CloudTempRanges [cloud_temp_min] [cloud_temp_max]
```

Temperature ranges (in °C) used for estimation of cloud cover using the CLOUDCOV_UBCWM UBCWM model approach, not otherwise needed. Must be between the :Gauge–:EndGauge brackets.

```
:EnsimTimeSeries [filename]
```

A table of time series (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 D.2. This must be between the :Gauge–:EndGauge brackets when providing gauge meteorological data. 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
4.4 -0.6  0
5.0 -2.5  0.6
...
5.6 -3.0 0.3
4.4 -4.6 0.0
1.1 -4.4 0.0
```

Any individual time series can also be read from NetCDF files using a `:ReadFromNetCDF-:EndReadFromNetCDF` block. This command works with ALL of the time series in format similar to the `:Data-:EndData` block, and replaces the date/time/interval/data vector contents,e.g.,

```
:Data [forcing type] [unit]
  :ReadFromNetCDF
    :FileNameNC      [path/filename of .nc file]
    :VarNameNC       [name of variable in .nc file]
    :DimNamesNC      [stations_name] [time_name] # (2-D) or
    :DimNamesNC      [time_name] (1-D)
    :StationIdx      [ID of station of interest (starts with 1)]
    #OR
    :StationIdx      FROM_STATION_VAR
    :TimeShift       [time stamp shift in days] #optional
    :LinearTransform [slope] [intercept] #optional
  :EndReadFromNetCDF
:EndData
```

This optional internal contents of the `:Data-:EndData` block can be used to generate forcing time series from NetCDF data. As indicated in documentation of the `:Data` command, the `forcing_type` is chosen from the options in table D.2. The NetCDF variables need to be either one-dimensional (time) or two-dimensional (time x stations or stations x time). Raven readily supports NetCDF input with an UNLIMITED time dimension. It is suggested that all NetCDF files follow the CF conventions (https://cfconventions.org/).

If the data are two-dimensional, the user needs to specify which station time series should be read in using the `:StationIdx` command. Alternately, as done with the Deltares-FEWS adaptor, users may specify the `FROM_STATION_VAR` flag, which requires that there is a `station_id` array attribute array in the NetCDF file containing gauge names consistent with the names in the `:Gauge` command and in the same order as the station data. Raven will identify the appropriate index from this mapping. For other time series which are not associated with meteorological gauges (e.g., observation data or reservoir operational rules), the `FROM_STATION_VAR` command requires that the `station_id` attribute includes either HRU or subbasin IDs instead of gauge names, as is appropriate for the time series. Therefore, gauge, subbasin, and HRU data must be stored in separate NetCDF input files.

For standard applications, neither the `:FileNameNC` or `:VarNameNC` should include an asterisk character. This wildcard character is replaced with the ensemble ID when Raven is running in ensemble mode,

including ensemble Kalman filter. In such a case, a different file name or variable name (respectively) will be used by each ensemble member. This asterisk/wild card usage is also only allowed in the auxiliary .rvt file specified using the `:ExtraRVTFile` command in the .rve file.

The `:ReadFromNetCDF` block also allows for the specification of a time shift `:TimeShift` in fractional days. For example, a time shift

```
:TimeShift -0.25
```

will shift all data earlier by 6 hours. Hence, a data point that was read for 8:00 am will be handled as 2:00 am in the model. The time shift only applies when the input data are sub-daily. Otherwise data can only be shifted by whole days.

The data read can also be linearly transformed using `:LinearTransform`. The slope and intercept specified will be applied to the data right after reading. The `:LinearTransform` can be used to apply unit conversions of the data. For example, the linear transformation

```
:LinearTransform 1.0 -273.15
```

would convert temperature data that were read in Kelvin into Celsius.

Advice

The `:LinearTransform` command is very useful for converting precipitation data that is natively in units of mm/data interval rather than mm/d, as Raven requires. If the NetCDF precipitation data is accumulated data, the `:Deaccumulate` command may be added to the `:ReadFromNetCDF` block, which will deaccumulate the data.

A.4.2 Observation Time Series

Time series of known flows or other model states may also be specified to support the model, either by calculating diagnostic metrics which report the fit of observed to simulated states, or to override stream discharge or reservoir levels with observed values. These are not linked to a specific Gauge, and would therefore not be included in an :Gauge...:EndGauge bracket. These time series should be stored in a separate .rvt file and referred to in the main .rvt file using the :RedirectToFile command. Note that all of the below time series may be read from NetCDF by using the :ReadFromNetCDF-:EndReadFromNetCDF command from the previous section.

```
:ObservationData [data_type] [basin_ID or HRU_ID] {units}
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  v_1
  v_2
  v_3
  ...
  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 D.1, unless the data is (1) a hydrograph, in which case the HYDROGRAPH tag is used, (2) a reservoir stage, in which case the RESERVOIR_STAGE tag is used, (3) a reservoir inflow (the RESERVOIR_INFLOW tag) (4) a reservoir net inflow (runoff+P-E, the RESERVOIR_NETINFLOW tag), (5) a river outlet water level (the WATER_LEVEL tag) (6) stream temperature (the STREAM_TEMPERATURE tag), (7) stream concentration (the STREAM_CONCENTRATION tag) or (8) lake/reservoir area (the LAKE_AREA tag, with observation units of m^2). For these seven variables, the basin ID is specified. For all other variables (e.g., soil moisture or snow), the HRU ID is specified. With the exception of the hydrograph and inflow hydrographs, 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.

> The units tag for observation data is informational only. Raven will not convert the values if non-standard units are provided. Here, discharge observations must be in m^3/s, stage and water levels in m, temperatures in $^\circ C$, concentrations in mg/l, and lake area in m^2.

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.

> It is critical that snow observations be converted to mm snow water equivalent, rather than snow depth, as often reported. This is a common oversight.

```
:ObservationWeights [data type] [ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  wt_1
  wt_2
  wt_3
  ...
  wt_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.

```
:IrregularObservations [data type] [ID] [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 simulated 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.

```
:IrregularWeights [data type] [ID] [N]
  [date yyyy-mm-dd] [time hh:mm:ss.0] wt_1
  [date yyyy-mm-dd] [time hh:mm:ss.0] wt_2
  ...
  [date yyyy-mm-dd] [time hh:mm:ss.0] wt_N
:EndIrregularWeights
```

This command is used apply weights to irregular observations, where `wt_i` is the weight for the i^{th} irregular data point in a corresponding `:IrregularObservations` time series. Weights must be between zero and one. If values in the time series are null or blank, the weights are automatically treated as zero. The data type, ID, and number of entries all need to match an existing `:IrregularObservations` time series.

A.4.3 Reservoir Control Time Series

The following time series commands deal with reservoir operational constraints.

```
:ReservoirExtraction [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  Q_1
  Q_2
  ...
```

```
  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. This command is usually used to represent diversion flow for irrigation or similar.

```
:VariableWeirHeight [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  h_1
  h_2
  ...
  h_N
:EndVariableWeirHeight
```

where `h_i` is the i^{th} height of the reservoir outflow weir in m. All weir heights should be positive and are relative to the minimum crest height of the stage-discharge curve (i.e., weir heights are not with reference to mean sea level). This minimum crest height is zero by default for a 'lake-like' reservoir (those specified using `:WeirCoefficient` and `:CrestWidth` parameters) and equivalent to the highest stage with zero discharge in reservoirs defined using the `:StageRelations` command. This time series of weir heights is only applied to the reservoir located at the outlet of the subbasin indicated by the basin ID.

```
:ReservoirMaxStage [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  h_1
  h_2
  ...
  h_N
:EndReservoirMaxStage
```

where `h_i` is the i^{th} maximum stage of the reservoir in m (usually with sea level as the datum), and Basin ID corresponds to the subbasin with the corresponding reservoir at its outlet. If the computed stage exceeds this stage during operation, the outflow from the reservoir will be adjusted so as to keep the stage at the specified maximum. This time series is often a constant value corresponding to the maximum flood pool level of a reservoir.

```
:ReservoirMinStage [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  h_1
  h_2
  ...
  h_N
:EndReservoirMinStage
```

where `h_i` is the i^{th} minimum stage of the reservoir in m (usually with sea level as the datum), and Basin ID corresponds to the subbasin with the corresponding reservoir at its outlet. If the simulated stage is below this stage during model operation, the outflow from the reservoir will be set to the minimum reservoir flow (as specified using the `:ReservoirMinStageFlow` command. This time series is typically used to represent reservoir rule curves.

```
:ReservoirMinStageFlow [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  Q_1
  Q_2
  ...
  Q_N
:EndReservoirMinStageFlow
```

where `Q_i` is the ith specified minimum stage discharge from the reservoir in m^3/s, and Basin ID corresponds to the subbasin with the corresponding reservoir at its outlet. If the simulated stage is below the stage specified by the `:ReservoirMinStage` command during model operation, the outflow from the reservoir will be set to this flow, overriding the flow determined through stage-discharge relations. This time series is typically used to represent reservoir rule curves.

```
:OverrideReservoirFlow [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  Q_1
  Q_2
  ...
  Q_N
:EndOverrideReservoirFlow
```

where `Q_i` is the ith overridden outflow rate from the reservoir in m^3/s, separated by the given time interval, and Basin ID corresponds to the subbasin with the corresponding reservoir at its outlet. Regardless of the stage-discharge relation for the reservoir, the flow will be overridden with this specified flow time series unless the value for `Q_i` is Raven's blank value of -1.2345, in which case the discharge will be calculated as normally done using the stage-discharge curve. This command is useful for replacing the calculated flow from a reservoir with observed flow during model calibration. It can also be used in short-term reservoir operations for evaluating discharge scenarios. The only time during which this specified flow is disregarded is if the maximum stage constraint for the reservoir (e.g., as specified using the `:ReservoirMaxStage` command) is exceeded.

```
:ReservoirTargetStage [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  h_1
  h_2
  ...
  h_N
:EndReservoirTargetStage
```

where `h_i` is the ith target stage of the reservoir in m (usually with sea level as the datum), and Basin ID corresponds to the subbasin with the corresponding reservoir at its outlet. If the simulated stage is above or below this stage during model operation, the outflow from the reservoir will be adjusted to move towards this target stage subject to the constraint that the maximum increase rate of the discharge (specified using the `:ReservoirMaxQDelta` command) is respected. This time series is typically used to represent reservoir rule curves. This target stage must be between the minimum and maximum stages specified using the `:ReservoirMaxStage` and `:ReservoirMinStage` commands. If the target stage is given a blank value (-1.2345) for any time increment, the model will use the discharge as calculated from the stage-discharge relation.

```
:ReservoirMinFlow [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  Q_1
  Q_2
  ...
  Q_N
:EndReservoirMinFlow
```

where `Q_i` is the ith minimum flow of the reservoir in m^3/s, and Basin ID corresponds to the subbasin with the corresponding reservoir at its outlet. If the simulated/calculated desired flow from the reservoir is below this flow rate, the flow rate is corrected to this minimum flow value. If downstream reservoir demands are included, they will increase the specified value of this minimum flow rate to also meet downstream demand.

```
:ReservoirMaxFlow [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  Q_1
  Q_2
  ...
  Q_N
:EndReservoirMaxFlow
```

where `Q_i` is the ith maximum flow of the reservoir in m^3/s, and Basin ID corresponds to the subbasin with the corresponding reservoir at its outlet. If the simulated/calculated desired flow from the reservoir is above this flow rate, the flow rate is corrected to this maximum flow value. Only the maximum stage constraint associated with the `:MaxReservoirStage` command will override this maximum flow constraint (i.e., the outflow from the reservoir can exceed this max flow if the reservoir is full).

```
:ReservoirMaxQDelta [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  QD_1
  QD_2
  ...
  QD_N
:EndReservoirMaxQDelta
```

where `QD_i` is the ith maximum flow rate increase in m^3/s/d, and Basin ID corresponds to the subbasin with the corresponding reservoir at its outlet. If the simulated stage is above the target stage indicated by the `:ReservoirTargetStage` command during model operation, the outflow from the reservoir will be adjusted to move towards this target stage subject to the constraint that the maximum increase rate of the discharge (specified using this command) is respected. This time series is typically used to represent reservoir rule curves.

```
:ReservoirMaxQDecrease [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  QD_1
  QD_2
  ...
  QD_N
```

```
:EndReservoirMaxQDelta
```

where QD_i is the i^{th} maximum flow rate decrease rate in $m^3/s/d$, and Basin ID corresponds to the subbasin with the corresponding reservoir at its outlet. If the simulated stage is above the target stage indicated by the :ReservoirTargetStage command during model operation, the outflow from the reservoir will be adjusted to move towards this target stage subject to the constraint that the maximum decrease rate of the discharge (specified using this command) is respected. This time series is typically used to represent reservoir rule curves.

A.4.4 Irrigation, demand, and diversions

```
:BasinInflowHydrograph [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  Q_1
  Q_2
  ...
  Q_N
:EndBasinInflowHydrograph
```

where Q_i is the i^{th} inflow in m^3/s. This command is typically used to (1) specify inflows coming from an unmodeled portion of the domain; (2) override simulated 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. The discharge is introduced at the *upstream* end of a basin reach, therefore this should typically not be used in headwater basins (see :BasinInflowHydrograph2.

```
:BasinInflowHydrograph2 [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  Q_1
  Q_2
  ...
  Q_N
:EndBasinInflowHydrograph2
```

where Q_i is the i^{th} inflow in m^3/s. This command is typically used to add (or subtract, if negative) inflows to or outflows from a stream reach from human activities, e.g., a wastewater treatment plant inflow or irrigation demand. The difference between this and :BasinInflowHydrograph is that it extracts/injects water from the downstream end of the basin stream reach rather than the upstream end. It may therefore be used in headwater basins.

```
:IrrigationDemand [SBID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  Qirr_1
  Qirr_2
  ...
  Qirr_N
:EndIrrigationDemand
```

This time series indicates the time history of irrigation demand, in m^3/s, from a subbasin, drawn from the outlet of the subbasin (downstream of the reservoir, if one is present). If there is sufficient water

available (i.e., enough to satisfy positive flow and/or the environmental minimum flow), the water will be removed. This approach is preferred over using negative flows in a :BasinInflowHydrograph time series, which will not respect these constraints.

```
:ReservoirDownstreamDemand [down_ID] [res_ID] [percent_met] {j1} {j2}
```

This command modifies the minimum outflow from a reservoir or set of reservoirs upstream of a subbasin with irrigation demand (supplied using the :IrrigationDemand time series command) in order to satisfy some percentage of this downstream irrigation demand. Here, down_ID is the subbasin index referring to a subbasin with an irrigation demand time series, the res_ID is the upstream subbasin index of a reservoir which releases sufficient water to satisfy this demand (i.e., its minimum flow rate is modified to satisfy demand), and percent_met is the total percentage of the irrigation demand met by this reservoir, from 0 to 1, with 1 indicating 100% of demand is met by the specified reservoir. If the res_ID is _AUTO and percent_met is _AUTO, the method indicated by the :ReservoirDemandAllocation command in the .rvi file will be used to identify all of the upstream reservoirs and split the demand between them according to contributing area, maximum capacity, or other metrics. Note that the percent met will be globally corrected by the global parameter RESERVOIR_DEMAND_MULT, so if percent met=0.5 and RESERVOIR_DEMAND_MULT is 1.2, then 60% of the downstream demand will be satisfied by the specified reservoir.

Optional terms j1 and j2 are the start and end Julian dates on which this constraint will be applied. Both terms are integers ranging from 1 (Jan 1) to 365 (Dec 31 in a non-leap year). The date range is inclusive, i.e., using the range 233-235 would enable this demand constraint from August 21st to August 23rd inclusively in a non-leap year or August 20th to August 22nd in a leap year.

```
:FlowDiversion [fromSBID] [toSBID] [pct] [Qmin] {start_day} {end_day}

:FlowDiversionLookupTable [fromSBID] [toSBID] {start_day} {end_day}
  nPoints
  {Qsource_i Qdivert_i} x nPoints
:EndFlowDiversionLookupTable
```

These commands provide rules for flow diversions to move water from the outlet of one subbasin to the inlet of another subbasin. The amount of diversion can either be in the form of a percentage of source water discharge (the :FlowDiversion command) or using a lookup table (the :FlowDiversionRatingCurve command. In both cases, the fromSBID and toSBID are the source and target subbasin IDs; if the toSBID is -1, then the water is diverted outside of the model. The optional start_day and end_day command elements are the Julian start date of diversion and Julian end date of diversion; if these are omitted, it is assumed the diversion is year-round. In the case of the :FlowDiversion command, the diverted flow is a percentage of the discharge from the source subbasin at the start of each time step which is diverted, where pct is the decimal percentage (0..1) of flow in exceedance of the minimum flow Qmin (in m^3/s) which is diverted, i.e.,

$$Q_{divert} = \alpha \cdot \max((Q_{source} - Q_{min}), 0)$$

In the case of the :FlowDiversionLookupTable command, the diverted flow is linearly interpolated from a lookup table specified by nPoints pairs of flows, where Qsource is the source basin discharge (in m^3/s) at the start of the time step and Qdivert is the corresponding diversion flow (in m^3/s). The Qsource flows must be in ascending order where the first point is always a flow of zero. No negative diversions are allowed, and the diversion flow should always be less than the source flow (hopefully for obvious reasons!).

```
:EnvironmentalMinFlow [Basin ID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  Q_1
  Q_2
  ...
  Q_N
:EndEnvironmentalMinFlow
```

where `Q_i` is the ith minimum flow constraint in m^{3}s^{-1}. This command is used in conjunction with the `:IrrigationDemand` time series at the same basin, and constrains the irrigation-based extraction such that the minimum flow is always respected. Note that simulated stream discharge can be less than this environmental minimum flow, but that irrigation cannot extract water such that this constraint is violated.

```
:UnusableFlowPercentage [Basin ID] [value]
```

This command is used to constrain the water available for irrigation demand, and would be used in conjunction with the `:IrrigationDemand` time series at the same basin. The value of this percentage (expressed as a fraction from 0.0 to 1.0) indicates the percentage of flow above the environmental minimum flow which is available for irrigation. For instance, if the discharge in a subbasin was 7 m^3/s and the environmental minimum flow was 3m^3/s and the unusable flow percentage was 0.3 =30%, the maximum amount of water that could be used to satisfy irrigation demand would be $(1 - 0.3) \cdot (7 - 3)$ =2.8 m^3/s. Most commonly, this would be used when there are regulatory constraints regarding allowable fractions of streamflow that may be used to satisfy water demand. The unusable flow percentage for all subbasins is zero by default, i.e., irrigation can draw down flows to the environmental minimum (or zero discharge, if the environmental minimum is not specified).

A.4.5 Transport Time Series

The following time series are used to support simulation of constituent transport. In all cases the constituent named `constit_name` must have been included in the `:Transport` command in the .rvi file.

```
:ObservationData STREAM_CONCENTRATION [SBID] {constit_name}
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  C_1
  C_2
  ...
  C_N
:EndObservationData
```

This is a special version of the `:ObservationData` command which can be used to supply in-stream concentrations (in mg/l) at the outlet of the main reach in subbasin `SBID`.

```
:ObservationData STREAM_TEMPERATURE [SBID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  T_1
  T_2
  ...
  T_N
```

```
:EndObservationData
```

This is a special version of the :ObservationData command which can be used to supply in-stream temperatures at the outlet of the main reach in subbasin SBID.

```
:ConcentrationTimeSeries [constit_name] [storage] {HRUGroup}
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  C_1
  C_2
  ...
  C_N
:EndConcentrationTimeSeries
```

This command is used to represent a Dirichlet (specified concentration) condition in a storage unit storage (e.g., SOIL[0]) for the constituent named constit_name. By default this is applied to all storage compartments of this type across the domain, unless the HRUGroup identifier is provided, in which case it is applied only to the specified HRU group. The concentrations C_i should be in units of mg/l. This is the transient equivalent to the :FixedConcentration command.

```
:MassFluxTimeSeries [constit_name] [storage] {HRUGroup}
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  M_1
  M_2
  ...
  M_N
:EndMassFluxTimeSeries
```

This command is used to represent a Neumann (specified flux) condition in a storage unit storage (e.g., SOIL[0]) for the constituent named constit_name. By default this is applied to all storage compartments of this type across the domain, unless the HRUGroup identifier is provided, in which case it is applied only to the specified HRU group. The mass fluxes M_i are provided in units of mg/m^2/d. This is the transient equivalent to the :MassFlux command.

```
:SpecifiedInflowConcentration [constit_name] [SBID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  C_1
  C_2
  ...
  C_N
:EndSpecifiedInflowConcentration
```

This command overrides the concentration of the inflow into a subbasin reach, where the subbasin is specified by its SBID and the constituent of interest is indicated by constit_name. The concentrations C_i must be in units of mg/L.

```
:SpecifiedInflowTemperature [SBID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  T_1
  T_2
  ...
```

```
  T_N
:EndSpecifiedInflowTemperature
```

This command overrides the temperature of the inflow into a subbasin reach, where the subbasin is specified by its `SBID`. The temperature `T_i` must be in units of °C.

```
:MassLoading [constit_name] [SBID]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  M_1
  M_2
  ...
  M_N
:EndMassLoading
```

This command adds a continuous loading of constituent mass into a subbasin reach, where the subbasin is specified by its `SBID` and the constituent of interest is indicated by `constit_name`. The mass loadings must be supplied in units of kg/d.

A.4.6 Special Commands

```
:AnnualCycle [J F M A M J J A S O N D]
# for instance:
:ReservoirTargetStage [Basin ID]
  :AnnualCycle 379 379 379 379 379 382 383 382 380 380 379 379
:EndReservoirTargetStage
```

This command may be used internal to any of the time series commands in this section (or any continuous single-data time series with a similar format), by replacing the date/time/interval/N and data vector contents between the `:Data` and `:EndData` (e.g.,) commands with the single `:AnnualCycle` command. The monthly values here will be interpolated using the method specified in the `:MonthlyInterpolationMethod` (section A.1.2) and used to populate a continuous cyclic time series.

```
:AnnualEvents
  {[jul_day_1] [value]} x (num. of events)
:EndAnnualEvents
# for instance:
:MassLoading NITRATE [Basin ID]
  :AnnualEvents
     114 100.0 #100 kg/d on Apr 24
     128 150.0 #150 kg/d on May 8
  :EndAnnualEvents
:EndMassLoading
```

This command may be used internal to any of the time series commands in this section (or any continuous single-data time series with a similar format), by replacing the date/time/interval/N and data vector contents between the `:Data` and `:EndData` (e.g.,) commands with the `:AnnualEvents-:EndAnnualEvents` command block. The event values here (assumed to last for one entire day) will be used to populate a continuous time series, with all other values assumed to be zero.

```
:OverrideStreamflow  [Basin ID]
```

This command overrides the discharge at the outlet of the basin defined with this ID. For this to work, there must be a corresponding observation HYDROGRAPH data set provided using the :ObservationData command, and there cannot be blank values in the data record during the course of the simulation. This command is often used to replace simulated inflows with observed inflows during model calibration. Note that the overriden flows will NOT be reported in the Hydrographs.csv output file. Rather, the simulated flows prior to overriding are reported.

```
:AssimilateStreamflow  [Basin ID]
```

This command indicates that the observed flows at the indicated basin should be assimilated, if the :AssimilateStreamflow command is included in the .rvi file. For this to work, there must be a corresponding observation HYDROGRAPH data set provided using the :ObservationData command. Blank values in the data record are allowed during the course of the simulation.

A.4.7 NetCDF Input Data

Raven supports gridded or station-based 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 (unless this mapping is 1:1 with subbasins/HRUs).

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
#
:RedirectToFile UpstreamInflow.rvt
:RedirectToFile LandCoverChange.rvt
:RedirectToFile ObservedHydrograph.rvt
```

The forcing inputs like precipitation and temperature are traditionally given as time series per gauging station (see appendix 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]
  :ElevationVarNameNC [elevation variable] #optional
  :TimeShift          [time stamp shift in days] #optional
  :LinearTransform    [slope] [intercept] #optional
  :Deaccumulate       #optional
```

```
  :GridWeights
    :NumberHRUs       [total number of HRUs]
    :NumberGridCells  [total number of grid cells]
      [HRU ID] [Cell ID] [weight]
      ...
  :EndGridWeights
  # OR #
  :MapStationsTo      [HRUS or SUBBASINS]
:EndGriddedForcing
```

One has to specify the type of the forcing input in the `:ForcingType` command, e.g. `PRECIP` or `TEMP_AVE` (see Table D.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. The data must be a 3D dataset; Raven does not support subsetting of higher dimensional (e.g., 4-D) arrays. One can also specify a time shift `:TimeShift` to shift the data. For example, when data given in UTC need to be shifted to local time; positive time shifts make the time stamp later, negative make it earlier. A linear transformation can further be applied to the data. For example, when data are given in Kelvin but need to be provided in Celsius for Raven. For instance, the following shift could be used to convert Fahrenheit temperature gridded data to Celsius for Raven:

```
  :LinearTransform 0.555555 -17.77777
```

which is equivalent to $T_C = (T_F - 32)/1.8$. Note that the output is the desired raven units and the input to the linear transform is in the NetCDF units. The `:Deaccumulate` command, if included will take cumulative precipitation values stored in the NetCDF file (such as are output by some weather models) and convert these to precipitation rates, in mm/d.

The (optional) `:ElevationVarNameNC` refers to an attribute in the NetCDF file (if present) which contains a surface elevation value for each grid cell, in meters above sea level. This is used for orographic corrections of precipitation or temperature data when scaled to the HRUs in the model. If not included, no orographic corrections are applied within the model.

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, i.e., the grid cell ID is $\mathrm{CELLID} = i_{row} * N_{col} + i_{col}$, where i_{row} and i_{col} are the row and column indices of the grid cell, and N_{col} is the number of grid columns. The valid range of grid cell IDs is therefore zero to $(N_{col}N_{row} - 1)$. The weights per HRU ID have to sum up to 1.0 otherwise Raven raises an error message. 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.

This grid weights command block may be generated using the python scripts available, along with an HRU shapefile and NetCDF input file, at:

https://github.com/julemai/GridWeightsGenerator

For NetCDF inputs that are not linked to a 2D grid, but rather store a vector of time series (e.g., from meteorological stations), the following alternate to :GriddedForcing is available:

```
:StationForcing [forcing name]
  :ForcingType       [type]
  :FileNameNC        [path/filename of .nc file]
  :VarNameNC         [name of variable in .nc file]
  :DimNamesNC        [station_name] [time_name]
  :TimeShift         [time stamp shift in days] #optional
  :LinearTransform   [slope] [intercept] #optional
  :Deaccumulate      #optional
  :GridWeights
    :NumberHRUs      [total number of HRUs]
    :NumberStations  [total number of stations]
      [HRU ID] [station ID] [weight]
      ...
  :EndGridWeights
  # OR
  :MapStationsTo [HRUS or SUBBASINS]
:EndStationForcing
```

This differs from :GriddedForcings only in the number of items in the :DimNamesNC command and the use of the :NumberStations command in the grid weights portion; the remaining components of the command are identical to that of :GriddedForcings, as defined above. This command is typically used when 2D (stations x time) array attributes are stored in the NetCDF file instead of 3D gridded data (typically, row x col x time).

```
    :MapStationsTo [HRUS or SUBBASINS]
```

The :MapStationsTo command replaces the :GridWeights-:EndGridWeights command block, and is specifically designed to support the FEWS forecasting. It presumes that the forcing data is stored as an 2D array of stations x time, where there is either a 1:1 relationship between stations and HRUs in the model (using the HRUS keyword) or a 1:1 relationship between NetCDF stations and subbasins (SUBBASINS keyword). It also presumes that there is a vector attribute called 'stations' in the NetCDF file which stores the ordered list of HRU or subbasin IDs, as specified in the .rvh file, where the station order is identical to the ordering of time series in the specified NetCDF variable indicated by :VarNameNC.

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). Alternately, the solution state from a previous run may be used to continue its progress seamlessly.

> Advice
>
> For a basic model run which allows some spin-up period, the .rvc file may generally be left empty with the exception of storage variables with long memories. For instance, groundwater storage and reservoir stage often have to be initialized, otherwise it may take years for these to fill up and equilibrate. Most other stores, including shallow soil storage, snow water equivalent, and depression storage will often equilibrate within a one- or two-year spin-up.
> For forecasting applications, it is recommended to use the solution.rvc from a previous equilibrated model run as the initial conditions for the future run (this is called a 'warm start').

Example File: modelname.rvc

```
# ------------------------------------------
# Raven Initial Conditions Input file
# ------------------------------------------
:HRUStateVariableTable
  :Attributes, SOIL[0], SNOW,
  :Units     ,      mm,   mm,
            1,      145,   33,
            2,      150,   13,
  ...
:EndHRUStateVariableTable
:UniformInitialConditions SOIL[3] 300

:BasinInitialConditions
  :Attributes,     Q
  :Units     ,  m3/s
           31,   3.2
           32,   3.6
  ...
:EndBasinInitialConditions

:InitialReservoirStage 32 432.1
```

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} x nHRUs
  # AND/OR
  {HRUGroup  , SV_value_1,SV_value_2,...,SV_value_NSV} x nHGs
```

```
:EndHRUStateVariableTable
```

Provides initial conditions for state variables in individual HRUs and/or across HRU groups within the model. Here, NSV is the number of state variables for which initial state variable conditions are provided, nHRUs is the number of HRUs for which initial conditions are specified (not all HRUs need to be contained in this table), nHGs is the number of HRU groups included in the table. SV_TAG refers to the state variable tag, with the complete list of valid state variable tags in table D.1. Each row can correspond to an HRU (in which case the first item is the unique integer HRU ID) or an HRU Group (in which case the first item in the row is the unique HRU group name). Each row must have NSV+1 entries. In the latter case, the initial conditions are applied to all initial conditions in the group. If HRU initial conditions are specified for both an HRU and for one or more HRU groups to which the HRU belongs, the last entry of this table will determine the actual initial conditions applied. This command may be repeated more than one time in the .rvc file; again, in the case of duplicate entries, the final state initial condition/HRU combination to appear is the one applied.

Initial conditions for chemical constituents or temperatures use the following special notation for the SV_TAG:
!NITRATE|SOIL[0]
or
!TEMPERATURE|SNOW
i.e., an exclamation point, followed by the name of the constituent followed by a vertical bar followed by the water compartment state variable name.

This table is automatically generated as output in the solution.rvc file for all simulated HRUs, and can be used to continue a previous simulation.

```
:UniformInitialConditions [SV_TAG] [value]
```

Applies a uniform initial condition (value) to the state variable corresponding to SV_TAG, with the complete list of water storage state variable tags (and units) in table D.1. While it depends upon the state variable, the value is usually in units of mm. If called after :HRUStateVariableTable, it will overwrite the initial conditions previously specified.

```
:UniformInitialConcentration [constit_name] [SV_TAG] [value]
```

Applies a uniform initial concentration (in mg/l) of the constituent named constit_name to the water storage state variable corresponding to SV_TAG, with the complete list of water storage state variable tags in table D.1. If called after :HRUStateVariableTable, it will overwrite the initial conditions previously specified.

```
:UniformInitialTemperature [SV_TAG] [value]
```

Applies a uniform initial concentration (in $^{\circ}$C) to the water storage state variable corresponding to SV_TAG, with the complete list of state variable tags in table D.1. If called after :HRUStateVariableTable, it will overwrite the initial conditions previously specified.

```
: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. If neither this command nor the :BasinStateVariables command is supplied, the initial flows in all reaches are estimated from the :AnnualAvgRunoff parameter.

```
:InitialReservoirStage [SBID] [stage]
```

Specifies initial reservoir stage for the reservoir located in the subbasin indicated by subbasin ID SBID, in meters. For lake-type reservoirs, this defaults to meters above crest width if no absolute crest height is provided. Typically, however, this height would be in metres above sea level, and the datum must be consistent with the stage data provided in the reservoir specification of stage-discharge-area-volume relationships.

```
:BasinStateVariables
  :BasinIndex SBID, 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 SBID, 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 each 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.

```
  :BasinTransportVariables [constit_name]
    :BasinIndex ID
      :ChannelMass [val]
      :RivuletMass [val]
      :Mout [nsegs]     [aMout     x nsegs    ] [aMoutLast]
      :Mlat [nQlatHist] [aMlatHist x nQlatHist] [MlatLast]
      :Min  [nQinHist]  [aMinHist  x nQinHist ]
      {:ResMassOut [Mout_res] [last_Mout_res]}
      {:ResMass [mass] [last mass]}
    :BasinIndex ID
    ...
  :EndBasinTransportVariables
```

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 mass transport variables linked to each subbasin. Here, :ChannelMass [mg] is the mass of constituent in the channel, :RivuletStorage [mg] is the mass within the water waiting in catchment storage, :Mout [mg/s] the array of mass outflows at each reach

segment, :Mlat [mg/s] is an array storing the time history of outflows to the channel, :Min [mg/s] is the time history of mass inflows to the uppermost segment of the reach. If a reservoir is linked to the subbasin, :ResMassOut is the mass outflow rate at the start and timestep, reported in mg/s, and :ResMass reports the total mass (in mg) within the reservoir at the end and start of the time step, respectively.

This content is also written to a state file if the constituent is TEMPERATURE, in which case the 'mass' terms actually refer to the enthalpy within the channel/rivulet/reservoir stores (in MJ), and the mass flow terms are energy flow rates, in MJ/s.

```
:TimeStamp [YYYY-mm-dd] [00:00:00.0]
```

This command is usually generated only as part of the Raven solution file and would not typically be modified by the user. 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.

```
:Nudge NUDGE_MULTIPLY [state_var] [mutiplier] {HRU Group}
# or
:Nudge NUDGE_ADD [state_var] [additive factor] {HRU Group}
```

This command can be used to collectively modify state variables either across the watershed or (optionally) just in one HRU group if a group is specified. This is typically used in a forecasting context or to test sensitivity to model initial conditions. The state_var variable must be one of the state variables in D.1 which exists in the model. If NUDGE_MULTIPLIER is used, all state variables of this type in the specified HRU group are multiplied by the factor provided (here, multiplier). If NUDGE_ADD is used, all state variables of this type in the specified HRU group are increased by the additive factor, which implicitly has the same units as the state variable. The :Nudge commands may be used in a separate .rvc file pointed to using the :RedirectToFile command.

A.6 Water Management Input file (.rvm)

The water management input file provides information used for water resources optimization within Raven, including optimization of water demand delivery and reservoir operations. The key content of this file is the specification of water management constraints (which must be satisfied in each time step) and water management goals (which the optimizer attempts to satisfy).

```
# ----------------------------------------
# Example Raven Water Management (.rvm) File
# ----------------------------------------
:LookbackDuration 3

# Demand specification section ---------------
:RedirectToFile demands.rvm
:WaterDemand 323 3230 RAND_Corp_12
  :AnnualCycle 0.01 0.02 0.02 0.8 0.8 0.8 0.8 0.8 0.8 0.2 0.2 0.2
  :Penalty 35
  :ReturnFraction 0.9
:EndWaterDemand

:DemandMultiplier   Plot7_Irrigation     0.95
:DemandPenalty      FarmerGlen           5

# goal specification section -----------------
:DefineDecisionVariable vFarmerBobsDelivered = !D121 + !D122 + !D134

:ManagementConstraint MothLakeBelowMaxStage
  :Expression  !h130 < 1233
  :Condition   DAY_OF_YEAR BETWEEN 121 239
:EndManagementConstraint

:ManagementGoal MothLakeFlowRamping_zoneB
  :Expression  !q130 < 20
  :Condition   !h130 < 1224 + @ts(headroom,0)
  :Penalty     10.0
:EndManagementGoal
```

As with the *.rvi file # denotes a comment, but $\star$ does not - it is a mathematical operator. There are no required commands in the .rvm file - all commands are optional.

A.6.1 Optional Control Commands

The below commands control the overall functioning of the water management optimization tool.

```
:LookbackDuration [duration]
```

The duration over which state variables are stored for use in management constraints and goals, where duration is given in days. For instance, if goals are to be constrained by the reservoir stage three days previous, then a lookback duration of at least 3 is needed. The lookback duration defaults to 1 day if not specified. Note this can get memory-expensive in an hourly model.

A.6.2 Optional Demand Specification Commands

Water demands can be defined as a time series of demands, as a function of flow, or using a user-specified general expression. By default, demands do not return any of their water back to the routing network. However, each demand can also be linked to a return flow which may be directed to the same river reach as it was withdrawn from, any other reach in the watershed, or sent to an HRU group as irrigation (equivalent to rainfall). Each water demand is assigned a penalty for not being satisfied (default=0.5 sm^{-3}). Any management goal or other demand with a higher penalty will get preferential access to water subject to environmental flow minima or mass balance requirements (i.e., satisfaction of positive or zero flows at every point in the stream network).

```
:WaterDemand [SBID] [demand_ID] [demand_name]
  :DemandTimeSeries
    :AnnualCycle [J F M A M J J A S O N D]
  :EndDemandTimeSeries
  :ReturnTimeSeries
    :AnnualCycle [J F M A M J J A S O N D]
  :EndReturnTimeSeries
  :ReturnFraction [fraction]
  :Penalty [magnitude]
:EndWaterDemand
```

This designates a demand just upstream of the outlet of the reach associated with subbasin `SBID`. The water demand must be given a unique `demand_ID` and `demand_name`. Within the command block, the demand magnitude must be specified as either a time series (`:DemandTimeSeries`), fraction of in-stream flow (`:DemandFlowFraction`), or general expression (`:DemandExpression`). Optionally, a return flow may be specified as a time series (`:ReturnTimeSeries`), fraction of delivered demand (`:ReturnFraction`), of general expression (`:ReturnExpression`). If both the return fraction and time series/expression are The destination of return flow defaults to the location of this demand. However, it can be overridden such that the delivery is applied to another subbasin reach (using the `:ReturnDestination` command). By default, delivered flow does not return to the watershed.

```
:ReservoirWaterDemand [SBID] [demand_ID] [demand_name]
  :DemandTimeSeries
    :AnnualCycle [J F M A M J J A S O N D]
  :EndDemandTimeSeries
:EndReservoirWaterDemand
```

This command operates identically to a `:WaterDemand` command block, except the demand is drawn from the reservoir associated with `SBID` rather than the subbasin channel outlet, and any return flows are likewise returned to the reservoir by default. It may include all of the internal commands listed below.

The following commands may be only used within a `:WaterDemand` - `:EndWaterDemand` block or `:ReservoirWaterDemand` - `:EndReservoirWaterDemand` block:

- `:DemandTimeSeries` or `:DemandFlowFraction`
- `:Penalty`
- `:ReturnTimeSeries`
- `:ReturnFraction`

- :ReturnDestination
- :ResetDate
- :IsUnrestricted

These commands are described below.

```
:DemandTimeSeries
  :AnnualCycle [J F M A M J J A S O N D]
:EndDemandTimeSeries
```

Specifies the water demand as a withdrawal rate in units of m^3/s. Typically specified using the // :AnnualCycle command for brevity, the default time series or :AnnualPattern formats documented in appendix A.4 may also be used, though a :RedirectToFile would be typically used in the former case. Cannot be used in conjunction with the :DemandFlowFraction command.

```
:DemandFlowFraction [fraction]
```

Specifies the water demand as a fraction of the total flow at the start of the time step, expressed as a decimal. For instance, if 0.1 % of the streamflow was to be diverted through a demand, fraction would be 0.001. Cannot be used in conjunction with the :DemandTimeSeries command. Must be inside a :Demand command block.

```
:DemandExpression !dxxx = [expression]
```

An arbitrary expression for the demand as a function of any value in the model (see description of :Expression later in this appendix section). The left hand of the expression should be of the form !dxxx, where xxx is the subbasin ID of *this* demand.

```
:Penalty [penalty]
```

Sets the penalty for unmet demand of the water demand. The penalty value must be positive, with zero penalty indicating that there is no consequence for unmet demand. Demands with higher penalties will effectively be given higher priority over other demands in the same system.

```
:ReturnTimeSeries
  :AnnualCycle [J F M A M J J A S O N D]
:EndReturnTimeSeries
```

Specifies the maximum quantity of return flow as a flow rate in units of m^3/s. Typically specified using the :AnnualCycle command for brevity, the default time series or :AnnualPattern formats documented in appendix A.4 may also be used, though a :RedirectToFile would be typically used in the former case. Cannot be used in conjunction with the :DemandFlowFraction command. If used in conjunction with a :ReturnFraction command, the actual return flow will be $Q_r = \min(\alpha * Q_d, Q_r^{spec})$, where α is the return fraction, Q_d is the delivered demand, and Q_r^{spec} is the return flow as specified by this time series. Must be inside a :Demand command block.

```
:ReturnFraction [fraction]
```

Specifies the maximum quantity of return flow as a percentage of delivered water. Must be between 0 and 1. If used in conjunction with a :ReturnTimeSeries command, the actual return flow will be

$Q_r = \min(\alpha * Q_d, Q_r^{spec})$, where α is the specified return fraction, Q_d is the delivered demand, and Q_r^{spec} is the return flow as specified by this time series. If not used in conjunction with a `:ReturnTimeSeries` command, the actual return flow will be $Q_r = \alpha * Q_d$. Must be inside a `:Demand` command block.

```
:ReturnDestination [SBID]
```

Specifies the subbasin reach where the return flow is directed, if the return fraction is greater than zero. The default is the same basin as the flow. The subbasin is referred to by the id `SBID`. For a reservoir, the same syntax is used, because the return flow will be added at the outlet of the reach draining into the reservoir, which is the same as being added as a direct inflow to the reservoir. Cannot be used in conjunction with the `:IrrigationDestination` command. This is often used to represent municipal water use, where water is delivered to a water treatment plant with some percentage returned via a wastewater treatment plant at another location in the stream network. Must be inside a `:Demand` command block.

Advice

Flow diversions from one water body to another may be simply built by creating a water demand with a return subbasin destination other than the source of the withdrawal.

A.6.3 Optional Demand Modification Commands

The following commands are used *outside* of a `:WaterDemand` command block or `:ReservoirWaterDemand` command block, typically to modify demand properties from their default values within a specific scenario. For instance, all of the water demand default properties could be specified in a single .rvm file, with another .rvm file added via a (easily commented out) `:RedirectToFile` command which modifies these defaults. Importantly, all of the below commands require that the demands already be defined at an earlier point in the master .rvm file.

```
:DemandMultiplier [demand_ID] [mult]
```

Multiplies demands from the water demand identified by `demand_ID` by the (positive) value `mult` for all model time steps. The `demand_ID` can be the integer demand identifier or the demand name, as specified in the `:WaterDemand` demand command in the .rvm file or the `:IrrigationDemand`, `:WaterDemand`, or `:ReservoirExtraction` time series commands in the .rvt file.

```
:DemandPenalty [demand_ID] [penalty]
```

Overrides or sets the penalty for unmet demand of the water demand identified by `demand_ID`, usually called after the demand has been defined in a `:WaterDemand` command block using the `:Penalty` command. The `demand_ID` can be the integer demand identifier or the demand name. The penalty value must be positive and usually between 0 and 10, with zero penalty indicating that there is no consequence for unmet demand.

```
:DemandResetDate [demand_ID] [julian_date]
```

Sets the date at which cumulative delivery for the year is reset to zero for the water demand identified by `demand_ID`. By default, this is a `julian_date` of zero, i.e., Jan 1. The demand specified by `demand_ID` must already have been defined a `:WaterDemand` command block. This overrides the value specified by the `:ResetDate` in the demand's `:WaterDemand` command block.

```
:DemandIsUnrestricted [demand_ID]
```

Indicates that a water demand is unconstrained by environmental flow minima constraints, i.e. that a water demand will only be unsatisfied if the river/reservoir dries out (a mass balance constraint) or if other management goals with larger flow penalties take precedence. The demand specified by `demand_ID` must already have been defined a `:WaterDemand` command block.

```
:DemandGroup [dgroupname]
  [demand_ID1] [demand_ID2] ... [demand_IDN]
:EndDemandGroup
```

Defines a demand group named `dgroupname` as a collection of water demands. The `demand_ID` can be the integer demand identifier or the demand name, as specified in the `:IrrigationDemand`, `:WaterDemand`, or `:ReservoirExtraction` time series commands. Demand groups can be used to collectively adjust demand from all group members.

```
:DemandGroupMultiplier [dgroupname] [multiplier]
```

Multiplies demands of all members of demand group `dgroupname` by the (positive) value `multiplier` for all model time steps.

A.6.4 Optional Constraint/Goal Specification Commands

Management constraints and goals are defined by general expressions composed of numerical values, named numerical constants, model state variables such as streamflow or reservoir stage, user-specified decision variables such as the total flow from four select basins, time series, lookup tables, and historical values of state variables. These expressions can collectively represent very general water management rules.

```
:NamedConstant [cname] [value]
```

Creates a named constant with name `cname` that can be used in constraint or goal expressions. Suggested naming convention is to name constants starting with a lowercase c, e.g., cPowerMultiplier or cRegulatoryLimitPct.

```
:LookupTable [name]
  N
  x1 y1
  x2 y2
  ...
  xN yN
:EndLookupTable
```

Creates a lookup table comprised of `N` pairs of x-values (the input to the lookup table) and y-values (the result of the lookup table, interpolated for any input value x). These lookup tables can be referred to in general expressions using `@lookup(name,x)`, which will return the interpolated y value corresponding to an input x value. If x is greater than `xN`, the linear segment between `xN` and `xN-1` is assumed to extend to any value. If x is less than `x1`, then the output is the value `y1`.

```
# this command can appear in an .rvt file OR an .rvm file:

:UserTimeSeries [name]
  [date yyyy-mm-dd] [time hh:mm:ss.0] [time interval (d)] [N]
  v_1
  v_2
  ...
  v_N
:EndUserTimeSeries

# or, for monthly varying values:

:UserTimeSeries [name]
  :AnnualCycle [J F M A M J J A S O N D]
:EndUserTimeSeries

# or, for specifying annual patterns:

:UserTimeSeries [name]
  :AnnualPattern [STEP or INTERPOLATE]
    [mm-dd] [v_1]
    [mm-dd] [v_2]
    [mm-dd] [v_3]
    [mm-dd] [v_4]
  :EndAnnualPattern
:EndUserTimeSeries
```

Creates a named time series of values. This time series can be referred to in general expressions using `@ts(name,n)`, where `n` is a time shift from the current time step, i.e., `n`=0 corresponds to the current time step, `n`=-3 is three time steps in the past, and `n`=7 is seven time steps in the future. See the descriptions of the `:AnnualCycle` and `:AnnualPattern` commands in appendix A.4.

```
:DefineWorkflowVariable [wv_name] = [expression]
# e.g.,
:DefineWorkflowVariable vBDeliv = !D121[-1] + !D122[-1] + !D134[-1]
:DefineWorkflowVariable vConv130= @lookup(Q130_lookup,!Q130[-1])
```

Creates and defines a workflow variable named `wv_name` and defines it using an expression which does not include decision variables (syntax rules for this expression are defined under the `:Expression` command definition below). These workflow variables are used to build complex (but readable) management rules. Unlike decision variables, they are calculated at the start of each routing time step prior to solving the demand problem. As such, their definitions cannot include any decision variables which are a by-product of optimization. They therefore can only be assembled from historical variables, but are preferred over decision variables because 1) they do not increase the computational cost of solving the optimization problem and 2) can be included in non-linear expressions such as lookup tables. Suggested naming convention is that the control variables should be nouns and start with a lowercase v, e.g., vTotalFlowDemand or vFlowToPowerConversion. If other workflow or design variables are used in the expression defining this variable, they need to be defined prior to this variable in the input file. Workflow variables are calculated in the order they are defined in the input file.

```
:WorkflowVarDefinition [wv_name]
  :OperatingRegime [nameA]
     :Expression [wv_name] = [expression]
     :Condition [condition]
  :EndOperatingRegime
  :OperatingRegime [nameB]
     :Expression [wv_name] = [expression]
  :EndOperatingRegime
:EndWorkflowVarDefinition
```

For more complex workflow variable rules where the definition of the variable is controlled by the current system state. The syntax is identical to the `:ManagementGoal` described below, except the expressions must begin with `wv_name =`, using the same name as appears in the first line of the command.

```
:DefineDecisionVariable [dv_name] = [expression]
# e.g.,
:DefineDecisionVariable vBobsDelivered = !D121 + !D122 + !D134
:DefineDecisionVariable vNelsonCreekQ  = !Q435
:DefineDecisionVariable vPowerConv13   = @lookup(Q13_lookup,!Q13[-1])
```

Creates and defines a decision variable named `dv_name` and defines it using a linear expression (syntax rules for this expression are defined under the `:Expression` command definition below). These decision variables are tracked by the model and reported in DemandOptimization.csv. They can also themselves be used to build complex (but linear) management rules. Suggested naming convention is that the decision variables should be nouns and start with a lowercase v, e.g., vTotalFlowDemand or vFlowToPowerConversion.

```
:DeclareDecisionVariable [dv_name]
```

Creates a decision variable named `dv_name` which may later be manipulated with management goals and constraints, but may not have a fixed definition. These decision variables are tracked by the model and reported in DemandOptimization.csv. They can also themselves be used to build complex (but linear) management rules. Suggested naming convention is that the decision variables should be nouns and start with a lowercase v, e.g., vTotalFlowDemand or vFlowToPowerConversion. User specified decision variables with complex definitions conditional on system state should first be declared, then their definition specified using a management condition.

Advice

Workflow variables and decision variables are often used in similar ways, but have a key difference: decision variables are calculated as part of the optimization problem in each time step, whereas workflow variables are calculated at the start of the time step. This also means that workflow variables may not be a function of decision variables, whereas decision variables can be a function of workflow variables. Lastly, workflow variables may be used within non-linear expressions such as lookup tables, min/max functions, or as the product of two variables. Decision variables are limited such that management goals can only be constructed of linear combinations of decision variables (including Raven state variables).

```
:DecisionVariableBounds [dv_name] [lower] [upper]
```

Provide upper and lower bounds to a user-specified decision variable already defined using the `:DefineDecisionVariable` or `:DeclareDecisionVariable` command. These bounds act as constraints in the model, and the decision variable cannot take on values outside of this range. By default, decision variables have the lower bound 0 and upper bound ∞.

```
# for a single operating regime:
:ManagementGoal [goal_name]
  :Expression   [expression]
  :Condition    [condition1]
  ...
  :Condition    [conditionN]
  :Penalty      [value1] {value2}
:EndManagementGoal

# OR, using multiple operating regimes:
:ManagementGoal  [goal_name]
  :OperatingRegime [name]
    :Expression  [expression]
    :Condition   [condition1]
    :Condition   [condition2]
    ...
    :Condition   [conditionN]
  :EndOperatingRegime
  :OperatingRegime [name]
    :Expression  [expression2]
  :EndOperatingRegime
  :OperatingRegime [name] #default-  no conditions
    :Expression  [expression2]
  :EndOperatingRegime
  :Penalty       [penalty1] {penalty2}
:EndManagementGoal
```

Define a water management goal defined by the expression `expression`, which must be linear with respect to decision variables. The expression may be an equality or inequality. For the single operating regime case, the goal expression is only applied if all of the conditionals are satisfied, and is always applied if no conditional statements are present. For the case of several operating regimes, the expression corresponding to first operating regime whose conditions are all satisfied will be applied. Therefore, the operating regimes define sets of conditions defining when different management goals are active. Every operating regime must have one `:Expression` command within it. The naming convention for goals is usually a verb, e.g., SetMaxDixonReservoirStage or AdjustDemand.

The specified penalty terms are the penalty multipliers associated with violating the goal expression, such that the magnitude of the penalty will be:

- $P_1 \cdot (\text{RHS} - \text{LHS})$ if the expression is of the form LHS $>$ RHS,
- $P_1 \cdot (\text{LHS} - \text{RHS})$ if the expression is of the form LHS $<$ RHS, and
- $P_1 \cdot \max(\text{LHS} - \text{RHS}, 0) + P_2 \cdot \max(\text{RHS} - \text{LHS}, 0)$ if the expression is a target of the form LHS $=$ RHS. P_1 (`penalty1`) is the penalty for exceeding the target right hand side, and P_2 (`penalty2`) is the penalty for being below the target right hand side.

Note that the second penalty is only required if the goal expression is an equality. Syntax for the `:Expression`, `:OperatingRegime`, and `:Constraint` commands are described below.

```
# for a single operating regime:
:ManagementConstraint [constraint_name]
  :Expression [expression]
  :Condition [condition1]
  :Condition [condition2]
  ...
  :Condition [conditionN]
:EndManagementConstraint
```

Define a water management constraint defined by the expression `expression`, which must be linear with respect to decision variables. The expression may be an equality or inequality. The constraint is only applied if *all* of the conditionals are satisfied, and is always applied if no conditional statements are present (alternatively, multiple operating regimes may be applied as done above in the `:ManagementGoal` example).

The difference between a constraint and goal is that the constraint must be satisfied, while the optimization algorithm will try its best to satisfy a goal. For this reason, constraints should be used sparingly as they may over-constrain the optimization problem such that there is no feasible solution. For instance, simultaneously enforcing a maximum reservoir stage constraint and maximum reservoir outflow constraint could be impossible if the reservoir is already at its max stage and there is a large inflow rate: either the stage must increase or the reservoir must be allowed to release more water. It is better to treat one or both of these constraints as goals, where exceeding the maximum reservoir outflow or stage is discouraged via a penalty, but allowed to happen if it must to preserve mass balance. Syntax for the `:Expression`, `:OperatingRegime`, and `:Condition` commands are described below.

> When in doubt, express your management intent as a goal rather than constraint. Constraints that unintentionally conflict with mass balance (or each other) will lead to infeasible solutions and stoppage of the model.

```
:Expression [expression]
```

The `:Expression` command must be within a `:ManagementConstraint`, `:ManagementGoal`, or `:WorkflowVarDefinition` command block, and will define an equality or inequality relationship between one or more variables. Each expression must include one or more decision variables and the expression must be linear with respect to decision variables. Linearity means that 1) decision variables are not included within a function (such as a lookup table or min/max function) and 2) the product or quotient of two or more decision variables is not present, and 3) decision variables do not appear in the divisor of a term. Within these rules, the expression can be built with nearly arbitrary complexity by multiplying, adding, subtracting or dividing the following single terms :

- `vConstant`
  named constants defined using the `:NamedConstant` command.
- `@ts(tseries,n)`
  a time series defied using the `:TimeSeries` command in the .rvt file. Here, `tseries` is the name of the time series and `n` is the number of time steps prior to the current time step if negative, or the number of time steps after the current time step if positive.

- `@lookup(table,exp)`
  a lookup table defined using the `:LookupTable` command. Here, `table` is the name of the lookup table and `exp` is a single-term input expression. The expression cannot be a decision variable.
- `@min(exp1,exp2)`
  a function evaluating the minimum of the single-term input expressions `exp1` and `exp2`, e.g., `@min(!Q130,10.0)`. Neither expression can be a decision variable.
- `@max(exp1,exp2)`
  a function evaluating the maximum of the single-term input expressions `exp1` and `exp2`, e.g., `@max(!D130,!D120)`. Neither expression can be a decision variable.
- `@SB_var(state var,SBID)`
  a function returning the mean value of a state variable (from table D.1) in the subbasin with a specific subbasin ID (`SBID`).
- `@HRU_var(state var,HRUID)`
  a function returning the mean value of a state variable (from table D.1) in the HRU with a specific ID (`HRUID`).
- Unit conversion constants
  reserved constants used to multiply or divide quantities to convert units. Current units conversions supported include `ACREFTD_TO_CMS`, `CMS_TO_ACREFTD`, `MM_TO_INCH`, `INCH_TO_MM`, `FEET_TO_METER`, `METER_TO_FEET`, `CMS_TO_CFS`, `CFS_TO_CMS`, `ACREFT_TO_M3`, and `M3_TO_ACREFT`. These are reserved keywords, and therefore cannot be used as named constants.
- Internal state variables within Raven (treated as decision variables within the management optimization problem) can be referred to using the `!` notation:
  - `!Qxxx` is the stream or reservoir (i.e., subbasin) outflow in the subbasin with ID `xxx` [m^3/s]
  - `!hxxx` is the absolute reservoir or lake stage in the subbasin with ID `xxx` [m]
  - `!Ixxx` is the inflow to a reservoir in the subbasin with ID `xxx` [m^3/s]
  - `!qxxx` is the rate of change in reservoir outflow from the subbasin with ID `xxx` [$m^3/s/d$]
  - `!Dxxx` is the satisfied/delivered demand with demand ID `xxx` (demand IDs can refer to stream or reservoir demands) [m^3/s]

  These same variables may likewise be referred to using the subbasin, demand, or reservoir/lake name by adding a period after the state variable letter
  - `!Q.subname` (e.g., `!Q.NelsonCreek`)
  - `!h.resname` (e.g., `!h.PittsDam`)
  - `!I.resname`
  - `!q.resname`
  - `!D.demand_name` (e.g., `!D.GrahamRanchAllocation`)

  Lastly, the following notation can be used to refer to historical values of state variables:
  - `!Q.RavenRiver[-2]` refers to the streamflow in the Raven river subbasin from two time steps ago. There is a limit to how far back the history is stored, defined by the `:LookbackDuration` command. No spaces can be within the square brackets.
  - `!Qxxx[-n]` refers to the streamflow in river with subbasin ID `xxx` from `n` time steps ago.

Supported history variables include `!Q`, `!D`, `!I`, `!h`, `$B`, and `$E`.

- Other internal Raven variables within Raven can be referred to using the `$` notation:
  - `$dxxx` is the demand target with demand ID `xxx` (demand IDs can refer to stream or reservoir extractions)[m3/s]
  - `$Cxxx` is the cumulative satisfied/delivered demand with demand ID `xxx`, accumulated since the demand reset date for that demand [m3]
  - `$Bxxx` is the specified basin inflow hydrograph to the subbasin with ID `xxx` (from a `:BasinInflowHydrograph` time series) [m]
  - `$Exxx` is the environmental minimum flow in the subbasin with ID `xxx` (from a `:EnvironmentalMinFlow` time series) [m3/s/d]

  The `$B` or `$E` variables can be followed with the `[n]` suffix, such that past or future values of these two time series may be used in an expression

The above syntax can also be used elsewhere in rvm file commands, including within the `:Condition` command (for building general conditions), the `:DefineDecisionVariable` command, and the `:DefineWorkflowVariable` command.

Valid example expressions:

```
:Expression !Q120 + !Q181 = @ts(Q130_power_target,0) * vPowerConvert
```

Example invalid expressions:

```
# you cannot include product of two state variables (non-linear):
:Expression !Q120 * !Q181 = 40

# parentheses are not yet supported:
:Expression !Q120 < (!Q150 + !Q141) * cVal
# instead, use:
:Expression !Q120 < !Q150 * cVal + !Q141 * cVal

# a decision variable can't be within a lookup table (non-linear):
:Expression !Q181 = @lookup(myLUT,!Q120)

# nested functions are not yet supported:
:Expression !d181 = @lookup(myLUT,@ts(myTS,0))
# instead, use:
:DefineWorkflowVariable currTS = @ts(myTS,0)
:Expression !d181 = @lookup(myLUT,currTS)
```

If an expression refers to a state variable which is in a disabled portion of the model domain, the entire constraint/goal will be ignored.

```
:Condition [condition] [comparator] [value] {bound}
```

One or more conditional statements may be included within a `:ManagementGoal` or `:ManagementConstraint` command block, optionally within an `:OperatingRegime` command block. If multiple conditional statements are included, then all of the conditions have to be satisfied

within a given time step in order for the goal or constraint to be applied. If multiple operating regimes are specified, the expression within first operating regime whose conditions are all satisfied is applied.

The conditional statements are in one of the following forms:

```
:Condition [variable] > [expression]
:Condition [variable] < [expression]
:Condition [variable] = [expression]
:Condition [variable] IS_NOT_EQUAL_TO [value]
:Condition [variable] IS_BETWEEN [lower bound] [upper bound]
```

Where `expression` uses the same syntax as the `:Expression` command defined above. Here, the `variable` term can be one of:

- `DATE` - dates in the form yyyy-mm-dd
  e.g., `:Condition DATE IS_BETWEEN 1975-01-02 2010-01-02`
- `DAY_OF_YEAR` - Julian dates, from 0 (Jan 1) to 364 (Dec 31)
- `MONTH` - month as an integer, from 1 (Jan) to 12 (Dec)
- `YEAR` - year, in yyyy format
  e.g., `:Condition YEAR IS_BETWEEN 1977 2025`
- A simulated internal state variable, in the form `!Qxxx` or `!hxxx` as defined above in the `:Expression` command
- A user-specified decision variable, created using the `:DeclareDecisionVariable` or `:DefineDecisionVariable` command

```
:Penalty [penalty] {penalty_over}
```

One penalty command may be included within a `:ManagementGoal` (but not a `:ManagementConstraint`) command block. If not included, the penalty defaults to a value of 1.0. The optional `penalty_over` term should be included only for equalities where exceedance of a target should be penalized more (or less) than falling short of the target.

Penalties are used for prioritization of management goals - by definition, goals which incur a larger penalty will be interpreted by the algorithm as more important than goals with lesser penalties. Penalties must be greater than zero. Very large values for penalties (e.g., >100000) may lead to instabilities in the linear programming solver.

Note that penalties magnitudes are implicitly tied to the units of the `:Expression` statement as noted above when discussing the management goal command block. If you write a management goal in terms of reservoir stage, for instance, it may be useful to multiply the magnitude of the penalty by the area of the reservoir (in m^2) divided by 86400 (sec/day). This is done automatically for all standard reservoir operation time series, but can also be enforced using the `:UseStageUnitsCorrection` command.

```
:UseStageUnitsCorrection [SBID]
```

This adjusts the units of a penalty if the management goal or constraint expression is expressed in units of stage instead of discharge. It must be inside a `:ManagementGoal` command block. Otherwise, it is assumed the units (and therefore the penalties incurred) of all goal expressions are in units of discharge.

```
:OverrideStageDischargeCurve [SBID]
```

Because outflows to reservoirs are by default handled using a constraint (the stage discharge curve), the constraint has to be disabled if a management goal (for instance, a target stage) is instead intended to be used. This command is typically associated with - but not contained within - one or more `:ManagementGoal` expressions that will override the default behaviour of a reservoir. Here, the `SBID` is the subbasin in which the reservoir is located. Suggested practice is to list all of these overrides at the top of the .rvm file so that it is clear which reservoirs are managed vs. those handled using stage-discharge relations only.

> It is important that if `:OverrideStageDischargeCurve` command is used, a goal or constraint must be applied to the stage or discharge of a reservoir for ANY possible conditions, otherwise the reservoir outflow is unconstrained and strange results may occur.

```
:LoopThrough [looptype] [groupname]
  [.rvm commands]
:EndLoopThrough
# OR
:LoopThrough LIST [size]
  [.rvm commands]
:EndLoopThrough
```

This command block allows for looping through sets of .rvm commands multiple times. The `looptype` is one of `SB_GROUP`, `DEMAND_GROUP`, or `LIST`.

If the `looptype` is `SB_GROUP`, the loop iterates through all of the subbasins in the subbasin group named `groupname`; any appearance of the wildcard ID will be replaced with the subbasin ID. Likewise, any appearance of the wildcard $NAME$ will be replaced with the subbasin name, as indicated in the .rvh file.

If the `looptype` is `DEMAND_GROUP`, the loop iterates through all of the demands in the demand group named `groupname`; any appearance of the wildcard ID will be replaced with the demand ID. Likewise, any appearance of the wildcard $NAME$ will be replaced with the demand name.

Lastly, if the `looptype` is `LIST`, the loop iterates through a vector of size `size`. Wildcards are handled through the `:LoopVector` command below.

There is no limit to the number of commands allowed within the `:LoopThrough` command block. However, care must be taken to not generate duplicate management goals; the wildcards are intended to be used to differentiate between contents of each loop. Nested loops are not supported.

```
:LoopVector $wildcard$ s1 s2 s3 ... sN
```

Located at the top of a `:LoopThrough`-`:EndLoopThrough` command block, the `:LoopVector` command defines a wildcard string that is replaced by string items within the vector of values (`s1`..`sN`). The wildcard must always be a string between two dollar signs, e.g., res_stage. The reserved wildcards ID and $NAME$ cannot be used. The size of the vector must be equal to the size of the subbasin group, demand group, or list being looped through. Everywhere the wildcard appears in the remainder of the `:LoopThrough`-`:EndLoopThrough` command block, the wildcard will be replaced by the string item corresponding to the current loop iteration.

A.7 Live file (.rvl)

The live file is intended for direct synchronous coupling of Raven with another model or software tool which would require regular information exchange with Raven during operation. Examples of such tools include reservoir optimization tools, software which represents dynamic land use or forestry practices, hydraulic models, or glacier mass balance models. Typically used in conjunction with the `:CallExternalScript` command, the live file (*.rvl) is read if the `:ReadLiveFile` command is included in the .rvi file. By default this file will be read at the start of every time step just prior to the mass balance calculations and enables external codes to dynamically modify Raven state variables, simulated flows, parameters, and landscape classification.

An example .rvl file is shown below.

```
:LandUseChange 2402 URBAN
:LandUseChange 2417 URBAN
:LandUseChange 2483 URBAN
```

This file would change the land use of HRUs 2402, 2417, and 2483 from their previous land use to URBAN. The URBAN land use parameters will now be used to represent the hydrologic response of these HRUs.

A.7.1 Commands

```
:VegetationChange [HRU_ID] [new vegetation class name]
```

Converts HRU indicated by `HRU_ID` from its previous vegetation class to the one specified. Useful for representing agricultural management.

```
:GroupVegetationChange [HRU_Group_Name] [new vegetation class name]
```

Converts all HRUs in the HRU group `HRU_Group_Name` from its previous vegetation class to the one specified.

```
:LandUseChange [HRU_ID] [new land use class name]
```

Converts HRU indicated by `HRU_ID` from its previous land use class to the one specified.

```
:GroupLandUseChange [HRU_Group_Name] [new land use class name]
```

Converts all HRUs in the HRU group `HRU_Group_Name` from its previous land use class to the one specified.

```
:HRUTypeChange [HRU_ID] [new HRU type]
```

Converts HRU indicated by `HRU_ID` from its previous type to the one specified.

```
:HRUTypeChange [HRU_ID] [new HRU type]
```

Converts all HRUs in the HRU group `HRU_Group_Name` from their previous type to the one specified.

```
:GroupLandUseChange [HRU_Group_Name] [new land use class name]
```

Converts all HRUs in the HRU group `HRU_Group_Name` from its previous land use class to the one specified.

```
:SetStreamflow [SBID] [Q]
```

Changes the outflow from subbasin with subbasin index `SBID` to the discharge Q in m^3/s.

```
:SetReservoirStage [SBID] [stage]
```

Changes the stage in the reservoir within subbasin `SBID` to the stage indicated (in m).

```
:RepopulateHRUGroup [group_name]
  [HRUID_1] [HRUID_2] ... [HRUID_N]
:EndRepopulateHRUGroup
```

Uses the same format as `:HRUGroup` command in the .rvh file. Allows dynamic repopulation of HRU groups defined in the .rvi file and/or earlier populated in the .rvh file. The repopulation command may be immediately followed by (e.g.,) a `:GroupLandUseChange` command call to dynamically adjust land use change externally via a script or call to an external model. For instance:

```
:RepopulateHRUGroup DeglaciatingHRUs
  1203, 1204, 1206, 1212
:EndRepopulateHRUGroup
:GroupLandUseChange DeglaciatingHRUs BARREN
```

A.8 Ensemble file (.rve)

The ensemble file is intended for controlling the operation of RAVEN when running in ensemble mode. Each ensemble mode uses different commands, and so this section is divided into subsections based upon these modes. The ensemble file is only read if the following command is found in the .rvi file:

```
:EnsembleMode [ens_mode] [num_members]
```

where `ens_mode` is the ensemble mode (currently only `ENSEMBLE_ENKF`) and `num_members` is the number of ensemble members.

All ensemble methods require the following command:

```
:OutputDirectoryFormat [directory, with wildcard]
# e.g.,
:OutputDirectoryFormat ./out/ens_*
```

This command defines the output directory setup, as each ensemble member will preferably be directed to its own directory. Any asterisks in the directory name will be replaced with the ensemble member number, from 1 to N. The directory should not have a final backslash appended. In general, it is desirable to have this as a subdirectory to the base output directory (i.e., './out' in the above example). The RavenErrors.txt and ensemble-specific summary output will be sent to the base output directory, with all other output directed to the subfolder. No spaces are allowed in the directory.

A.8.1 EnKF ensemble commands

```
:EnKFMode {mode}
```

This specifies the mode of EnKF operation, as outlined in section 6.4. Here, "mode" can be one of the following:

- `ENKF_SPINUP` - in spinup mode, each ensemble member uses the same initial conditions from the standard .rvc file. Perturbations are applied to forcings. EnKF is applied at the end of the simulation and generates a solution_EnKF.rvc file in each ensemble member folder.
- `ENKF_CLOSED_LOOP` - in closed loop mode, each ensemble member uses initial conditions from an EnKF-adjusted previous model run (from the solution_EnKF.rvc file). Perturbations are applied to forcings. EnKF is applied at the end of the simulation and generates a solution_EnKF.rvc file in each ensemble member folder.
- `ENKF_FORECAST` - in forecast mode, each ensemble member uses initial conditions from an EnKF-adjusted previous model run (from the solution_EnKF.rvc file). Perturbations are NOT applied to forcings, and EnKF is not applied. The only thing that varies between forecast members is the initial conditions set.
- `ENKF_OPEN_LOOP` - same as `ENKF_CLOSED_LOOP`, except without use of EnKF-updated states. Instead, initial conditions are read from the member-specific solution.rvc file for each member.
- `ENKF_OPEN_FORECAST` - same as `ENKF_FORECAST`, except without use of EnKF-updated states.

All modes assume that the time stamp on the solution.rvc and solution_EnKF.rvc are the same as the start time of the simulation. Only the closed loop and spinup modes actually apply Kalman Filter to their end model states.

```
:SolutionRunName [runname]
```

(Optional) The `runname` argument specifies the run name used in the previous ensemble run, which will be read as initial conditions for each member in closed-loop mode (users may wish to change run names between ensemble runs to prevent overwriting of results). If not included, it is assumed the run name is the same as the run name specified using the `:RunName` command in the .rvi file. Should not be used in conjunction with `:EnsembleRVCFormat`, which will override this command.

```
:EnsembleRVCFormat [filename, with wildcard]
# e.g.,
:EnsembleRVCFormat ./EnKFsols/runname_solution_EnKF_*.rvc
# or
:EnsembleRVCFormat ./input/ens_*/solution_EnKF.rvc
```

(Optional) This command defines the solution file read by each ensemble member, if these should vary by member. Any asterisks in the file path name will be replaced with the ensemble member number, from 1 to N. The file extension .rvc should be included in the filename. No spaces are allowed in the filename.

The `filename` argument specifies the .rvc filenames to be used for initial conditions, typically generated from a previous ensemble run. If not included, it is assumed the solution files are read from the output directories specified using the `:OutputDirectoryFormat` command OR the base .rvc file `model-name.rvc`. This command is only used to override this default behavior, usually because a forecasting system is running from alternative saved ensemble states. In the case of the EnKF model configuration, the source of the initial conditions file is by default determined by the `:EnKFMode`, but will be overridden by this command unless the mode is `ENKF_SPINUP`, in which case the single base .rvc is used for initial conditions. This command should not be used in conjunction with `:SolutionRunName`, as it will override the provided runname.

```
:ExtraRVTFilename [file.rvt]
```

(Optional) The contents of the specified .rvt is used in addition to the base and forecast .rvt files. If an asterisk is included in the filename, this is used as a wildcard character and replaced with the ensemble member number, from 1 to N. This allows for individual members to have access to ensemble member-specific time series inputs. This file must be specified with the .rvt extension. Relative or absolute path names are allowed.

```
:ForcingPerturbation [forcing] [dist] [p1] [p2] [adj] {HRU_grp}
```

This defines which forcings should be perturbed for each ensemble member prior to t_0 and how they should be perturbed. Here, `forcing` is the forcing type (must be one of the terms in table D.2), `dist` is one of the distributions found in table A.8, with parameters `p1` and `p2` fully defining the distribution, `adj` is one of `ADDITIVE` or `MULTIPLICATIVE` depending upon whether the perturbation is additive or multiplicative. Optionally, the HRU group may be specified using the `HRU_grp` term, in which case the perturbation is only applied locally. Additive pertubations should typically have a mean of zero, while multiplicative perturbations should have a mean of one. Raven supports an arbitrary number of forcing perturbations in the model.

```
:AssimilatedState [state] {group_name}
```

This command determines which model states are actually updated during EnKF data assimilation. The

Table A.8: Probability distributions supported by Raven. Units assume the variable of interest is in units of L.

tag	Definition	parameter 1	parameter 2
DIST_UNIFORM	uniform distribution	left bound [L]	right bound [L]
DIST_NORMAL	normal distribution	mean [L]	std. deviation [L]
DIST_GAMMA	Gamma distribution	shape [-]	scale [1/L]

command `state` is one of the state variable types in table D.1, and must exist in the model. The special state tags `STREAMFLOW` and `RESERVOIR_STAGE` may be used to update in-stream flows and reservoir/lake levels, in which case `group_name` must be a subbasin group. Otherwise, `group_name` is assumed to be an HRU group name. Raven does not yet support assimilation of transported constituent states.

```
:AssimilateStreamflow [SBID}
```

This command (which can also optionally appear in the .rvt file) indicates which streamflow observational time series are used in EnKF assimilation. Each stream gauge to be used in assimilation should appear in its own `:AssimilateStreamflow` command. The subbasin index `SBID` must exist in the model and should also have a `HYDROGRAPH`-type `:ObservationData` time series associated with it.

```
:ObservationErrorModel [state] [dist] [p1] [p2] [adj]
```

This command defines the observational error model for each observed state. The parameter `state` is the model state associated with the observation data, which much correspond to an entry in table D.1 (currently, only `STREAMFLOW` is supported). The command entry `dist` is one of the probability distributions found in table A.8, with parameters `p1` and `p2` fully defining the error distribution, `adj` is one of `ADDITIVE` or `MULTIPLICATIVE` depending upon whether the perturbation is additive or multiplicative (in most cases this should be additive with a mean of zero). Users should typically have one observational error model command for each observation type used in assimilation. If this command is not included for a given observed state, the error model is a Dirac distribution centred at zero (i.e., it assumes no observation error).

Appendix B

Output Files

B.1 Standard Output Files

By default, output files are in comma-delimited format, and can be readily opened up in Excel or R for post-processing, visualization, and analysis. Most of these files will alternatively be generated in NetCDF (*.nc) format if the command `:WriteNetCDFFormat` command is used in the .rvi file. Available output files include:

- `WatershedStorage.csv/.nc` (created by default)
  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. Note that it is normal that the Ponded Water and Surface Water stores are empty throughout the simulation, because these are temporary stores emptied out by the end of each time step. Created by default.

- `Hydrographs.csv/.nc` (created by default)
  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 simulated hydrograph. If the `:WriteLocalFlows` command is included in the .rvi file, local subbasin contributions to the outlet hydrograph for that subbasin are reported. Created by default.

- `RavenErrors.txt` (always created)
  A text file outlining model input errors, warnings, and advisories for the user.

- `ForcingFunctions.csv/.nc` (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.

- `Diagnostics.csv` (optional)

A comma-delimited file reporting the quality of fit between model and supplied observations. Created if observations are present and the `:EvaluationMetrics` command is used.

- `WatershedMassEnergyBalance.csv/.nc` (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/.nc` (optional)
  A comma-delimited file reporting the instantaneous stage of all simulated reservoirs where the corresponding subbasin is specified as 'gauged' in the .rvh file. Created automatically if reservoirs are present in the model.

- `Demands.csv/.nc` (optional)
  A comma-delimited file containing the time series of irrigation demand, environmental minimum flow, actual flow, and unmet demand for all subbasins specified as 'gauged' in the .rvh file. The output is all instantaneous, i.e., the reported values refer to snapshots in time. Created if `:WriteDemandFile` command is included in .rvi file.

- `WaterLevels.csv` (optional)
  A comma-delimited file containing the time series of stream water levels at subbasin outlets, or just upstream of reservoirs in the case of subbasins with a reservoir. Only reported at subbasins specified as 'gauged' in the .rvh file. Created if `:WriteWaterLevels` command is included in the .rvi file.

- `SubbasinProperties.csv` (optional)
  A comma-delimited file containing key calculated properties of all subbasins, including drainage area, catchment routing, and channel routing properties such as celerity and diffusivity. Created if `:WriteSubbasinFile` is included in the .rvi.

- `{constituent}concentrations.csv/.nc` (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.

- `Temperatures.csv` (optional)
  A comma-delimited file reporting the instantaneous watershed-averaged water temperature in all water storage units. Created automatically if thermal transport is included in the model.

- `{constituent}pollutograph.csv/.nc` (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.

- `StreamTemperatures.csv/.nc` (optional)
  A comma-delimited file reporting the instantaneous outflow water temperature from all gauged subbasins. Created automatically if thermal transport is included in the model.

- `{constituent}MassLoadings.csv` (optional)
  A comma-delimited file reporting the instantaneous mass loading (in kg/d) from the outlets of

gauged subbasins. Only generated if the :WriteMassLoadings command is used in the .rvi file. For synthetic tracers, this corresponds to the fractional flow (in m3/s) tagged with the tracer.

If the :RunName parameter is specified in the .rvi file, this run name is added as a prefix to the above filenames.

B.2 Custom Outputs

A variety of custom outputs of any state variable or mass flux in the model may be generated using the :CustomOutput command. See section A.1.4 for details. Note that for CONTINUOUS time aggregation, these custom outputs report instantaneous fluxes/states, and the forcing function rates (e.g., rainfall) will be period-ending, i.e., the average precipitation rate for the time step preceding the time stamp is reported. If the aggregation is MONTHLY, YEARLY, or WYEARLY, the variables reported are calculated over the entire corresponding period.

B.3 NetCDF Output Format

The .nc output hydrographs, reservoir stages, water level, forcing functions, transport outputs, and custom output files are generated if the :WriteNetCDFFormat command is used. Currently these are the only NetCDF-format outputs available; other outputs will still be generated in .csv format.

The NetCDF files written are compatible with NetCDF version 4.0 and greater. The contain an unlimited dimension for time. Depending upon the output file, other dimensions may include the number of sub-basins with simulated outflow nbasin_sim or the number of basins with observed outflows nbasin_obs. All floating-point variables are written in double precision. Multiple attributes are available for each output variable, such as units, long_name, _FillValue, and/or missing_value.

The header of an example Hydrographs.nc containing the results of a simulation with 2 sub-basins and streamflow observations for one sub-basin starting at Oct 1st, 1991 looks like:

```
netcdf Hydrographs {
dimensions:
  time       = UNLIMITED ;
  nbasin_sim = 2 ;
  nbasin_obs = 1 ;
variables:
  double time(time) ;
    time:units              = "days since 1991-10-01 00:00:00" ;
    time:calendar           = "gregorian" ;
  double precip(time) ;
    precip:units            = "mm d**-1" ;
    precip:long_name        = "Precipitation" ;
    precip:_FillValue       = -9999. ;
    precip:missing_value    = -9999. ;
  string basin_name_sim(nbasin_sim) ;
    basin_name_sim:long_name = "ID of sub-basins with simulated outflows" ;
  double q_sim(time, nbasin_sim) ;
    q_sim:long_name         = "Simulated outflows" ;
    q_sim:units             = "m**3 s**-1" ;
    q_sim:_FillValue        = -9999. ;
```

```
    q_sim:missing_value      = -9999. ;
  string basin_name_obs(nbasin_obs) ;
    basin_name_obs:long_name = "ID of sub-basins with observed outflows" ;
  double q_obs(time, nbasin_obs) ;
    q_obs:long_name          = "Observed outflows" ;
    q_obs:units              = "m**3 s**-1" ;
    q_obs:_FillValue         = -9999. ;
    q_obs:missing_value      = -9999. ;
}
```

Appendix C

Using Raven with Deltares FEWS

Raven 3.1 and above works seamlessly with the Deltares Flood Early Warning System (FEWS) software program. A schematic of the connections between Raven and FEWS is shown in figure C.1. In this setup, Raven and FEWS communicate entirely via NetCDF files, so this only works with the FEWS version of the Raven software.

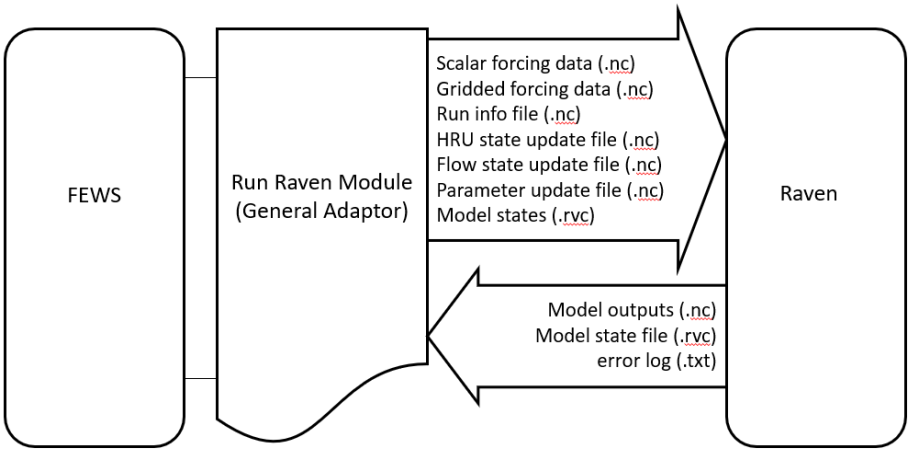

Figure C.1: The coupling between FEWS and Raven

C.1 Setup

The FEWS directory structure should be setup as follows:

- ./input/
  - FEWS-generated model input. NetCDF files which include the meteorological forcings and input time series
  - FEWS-generated state update and parameter update files (format defined below)
- ./output/
  - The Raven output directory. FEWS will read model output from here.
- ./work/

  - The base model .rvi, .rvh, .rvp, .rvc, and .rvp files

- run_info.nc

To port an existing Raven model to use the FEWS framework, the first step is to include the following command block in the header of the .rvi file:

```
:DeltaresFEWSMode
:WriteNetCDFFormat
:OutputDirectory        ../output/
:FEWSRunInfoFile        ../run_info.nc
:FEWSStateInfoFile      ../input/state_update.nc
:FEWSBasinStateInfoFile ../input/flow_update.nc
:FEWSParamInfoFile      ../input/param_update.nc
```

This ensures that all output is generated in NetCDF format, the output is directed to the correct location in the file structure, and that Raven can find the FEWS-generated run information, state update, and parameter update files. The `:DeltaresFEWSMode` ensures that missing data in the first time step of meteorological inputs is ignored.

All meteorological forcing inputs and time series (such as reservoir controls or observed flow data) should be generated by FEWS into the 'input' folder. Raven can read this input via the use of `:ReadFromNetCDF`, `:GriddedForcing`, and `:StationForcing` commands in the .rvt, as documented in appendix A.4. No specific naming convention for time series variables is required, as these are specified in the .rvt file.

To support run control by FEWS, Raven reads three special FEWS-generated input files, in addition to the standard .rv files:

- **Run Information File (run_info.nc)** -
  specifies the simulation start date and end date; also provides access to some basic run control (equivalent to `:RunName`, `:Mode`, `:NoisyMode`, and other basic I/O commands).
- **State Update File (state_update.nc)** -
  used to override initial model states, including SWE and soil moisture in all HRUs
- **Basin State Update File (flow_update.nc)** -
  used to override initial basin-linked model states, including discharge and lake/reservoir stage in all subbasins.
- **Parameter Update File (param_update.nc)** -
  used to override model parameters; including global, land use, soil, vegetation, subbasin or gauge parameters.

The content of these files overrides any existing commands in the .rvi, .rvp, or .rvc files. The required format of these files is indicated in the following section.

C.2 FEWS adaptor files

C.2.1 Run information file

The run information file, with filename indicated with the `:FEWSRunInfoFile` command, overrides some basic commands in the .rvi file for run management. The following variables in the NetCDF file will be processed:

- start_time
  A double variable named start_time with units in standard NetCDF format (e.g., 'minutes since YYYY-MM-DD...'). The time zone will be ignored and assumed to be in the same time zone as the model. This will override the simulation start date indicated by the :StartDate command.
- end_time
  A double variable named end_time with units in standard NetCDF format (e.g., 'minutes since YYYY-MM-DD...'). The time zone will be ignored and assumed to be in the same time zone as the model. This will override the simulation end date indicated by the :EndDate or :Duration commands.
- properties
  A multi-attribute character variable named properties. The following attributes within properties will be read:
  - RunName
    Corresponds to the Raven command :RunName. A string.
  - BlockRavenWarnings
    Equivalent to the Raven command :SuppressWarnings. Must be true (suppress) or false (don't suppress).
  - BlockRavenCustomOutput
    Disables all writing of custom outputs if true. Must be true or false.
  - NoisyMode
    Equivalent to the Raven command :NoisyMode command if true. Must be true or false.
  - SilentMode
    Equivalent to the Raven command :SilentMode command if true. Must be true or false.
  - Mode
    Equivalent to the Raven command :Mode command if true. Must be a single character. Because this gets processed AFTER reading of the .rvi file, all :IfModeEquals command blocks in the .rvi file will not properly register; in such cases, the mode should be supplied at the command line or manually rather than via the run information file.
  - AssimilateStreamflow
    Equivalent to the :AssimilateStreamflow command in the .rvi file if true. Must be true or false.
  - EnKFMode Equivalent to the :EnKFMode command in the .rve file. A string. Must be one of ENKF_SPINUP, ENKF_CLOSED_LOOP, ENKF_OPEN_LOOP, ENKF_FORECAST, or ENKF_OPEN_FORECAST.

C.2.2 State update file

The state update file, with filename indicated with the :FEWSStateInfoFile command in the .rvi file, overrides some basic commands in the .rvc file model for direct initialization by FEWS.

The state update file MUST include:

- the dimensions time (size of the time vector) and stations (size of the HRU vector), where the number of 'stations' must be equal to or fewer than the number of HRUs in the Raven model.

- the character variable station_id, with dimensions (stationsxchar_leng_id). This vector of character strings encodes the list of HRUs which may be modified in state; these HRU IDs must be present in the :HRUs command block inside the .rvh file. Any unrecognized HRU IDs in this variable will cause an error to be thrown.
- the double variable time with dimension (time). This vector, with NetCDF time units (e.g., 'minutes since YYYY-MM-DD...', must be ordered and have one of its entries correspond to the simulation start time (typically the one indicated in the run information file).

In addition, the state update file can include any number of (time x station) float array variables which store model states at a number of times. The names of these variables must correspond to state variable names in the model (e.g., SOIL[0] or |SNOW|, see table D.1). All names not in the model will be ignored, as will all other variables in the update file. If the state variable name is present in the model, the row of values corresponding to the model start time will be used to re-initialize the state variables prior to model simulation, acting as initial conditions to the model. Only the HRUs included in the station_id list will be updated, and if the value is blank, updating will not be performed. The column indices of the array are assumed to align with that of the station_id variable, i.e., if the 3rd item in the station_id variable is '1001', then the 3rd state variable value in the state array will be used to update HRU 1001.

Note that only HRU-linked state variables will be adjusted using this file; basin-linked state variables (such as stream discharge and reservoir stage) must be present in a separate file.

C.2.3 Subbasin State update file

This is similar to the state update file, but is used to initialize states which are linked to subbasins, such as stream discharge or reservoir stage. The subbasin state update file, with filename indicated with the :FEWSBasinStateInfoFile command in the .rvi file, overrides some basic commands in the .rvc file model for direct initialization by FEWS.

The state update file MUST include:

- the dimensions time (size of the time vector) and stations (size of the subbasin vector), where the number of 'stations' must be equal to or fewer than the number of subbasins in the Raven model.
- the character variable station_id, with dimensions (stationsxchar_leng_id). This vector of character strings encodes the list of subbasins which may be modified in state; these subbasin IDs must be present in the :SubBasins command block inside the .rvh file. Any unrecognized subbasin IDs in this variable will cause an error to be thrown.
- the double variable time with dimension (time). This vector, with NetCDF time units (e.g., 'minutes since YYYY-MM-DD...', must be ordered and have one of its entries correspond to the simulation start time (typically the one indicated in the run information file).

In addition, the state update file can include any number of (time x station) float array variables which store model states (flows or stages) at a number of times. The names of these variables must be one of: BASIN_OUTFLOW or RESERVOIR_STAGE. All other variables in the update file will be ignored. The row of values corresponding to the model start time will be used to re-initialize the state variables prior to model simulation, acting as initial conditions to the model. Only the subbasins included in the station_id list will be updated, and if the value is blank, updating will not be performed. If the stage is supplied for a subbasin without a reservoir, it will be ignored. The column indices of the array are assumed to align with that of the station_id variable, i.e., if the 3rd item in the station_id variable is '301', then the 3rd state variable value in the state array will be used to update subbasin 301.

C.2.4 Parameter update file

The parameter update file, with filename indicated with the :FEWSParamInfoFile command in the .rvi file, overrides parameter values in the .rvp file model for direct parameter override by FEWS.

The parameter update file MUST include:

- the dimensions time (size of the time vector)
- the double variable time with dimension (time). This vector, with NetCDF time units (e.g., 'minutes since YYYY-MM-DD...', must be ordered and have one of its entries correspond to the simulation start time (typically the one indicated in the run information file).

In addition, the parameter update file can include any number of 1-D array variables of size time which store parameter values at a number of times. The names of these variables in the NetCDF file will be of the format PARAM_in_CLASS, where PARAM is a valid parameter name and CLASS is one of:

- A valid soil class, vegetation class, land use class name which exists in the .rvp file;
- The string GLOBALS, if the parameter is a global parameter;
- A valid gauge name which is specified in the .rvt file;
- An integer value corresponding a subbasin ID in the .rvh file; OR
- The string SUBBASINxxx where xxx is a subbasin ID

Example valid variable names include ADIABATIC_LAPSE_in_GLOBALS, BASEFLOW_COEFF_in_SILT, MANNINGS_N_in_SUBBASIN12, MANNINGS_N_in_12, RAINFALL_CORR_in_X3054, or IMPERM_FRAC_in_URBAN. All names not in the model will be ignored, as will all other variables in the update file. If the parameter is in the model, the parameter value corresponding to the model start time will override the value as specified in the .rvp or .rvt file. If the value is blank, parameter updating will not be performed.

C.2.5 Management Time Series File

The water management time series file, with filename indicated with the :FEWSManagementInfoFile command in the .rvi file, supplies user-specified time series for use with FEWS, and is the equivalent of the :UserTimeSeries command in the .rvm file.

The management file MUST include:

- the dimensions time (size of the time vector)
- the double variable time with dimension (time). This vector, with NetCDF time units (e.g., 'minutes since YYYY-MM-DD...', must be ordered and have one of its entries correspond to the simulation start time (typically the one indicated in the run information file).

In addition, the management update file can include any number of 1-D array variables of size time which store arbitrary management time series. The names of these variables in the NetCDF file will be of the format UserTimeSeries_NAME, where NAME is the name of the time series as used in management commands within the .rvm file(s).

Appendix D

Reference Tables

Table D.1: All state variables currently available in Raven. This list of state variables is supported by the :HydroProcesses commands and :CustomOutput commands, amongst others.

State Variable	[units] Description
Required Water Storage Variables	
SURFACE_WATER	[mm] streams/rivulets - routed to outlet via in-catchment routing
ATMOSPHERE	[mm] atmosphere : receives 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
DEPRESSION	[mm] depression/surface storage
WETLAND	[mm] deep depression storage
LAKE_STORAGE	[mm] lake storage
SNOW	[mm] frozen snow storage (as mm SWE : snow water equivalent)
SNOW_LIQ	[mm] snow liquid water content
GLACIER	[mm] glacier melt/reservoir storage
GLACIER_ICE	[mm] glacier ice - typically assumed to be infinite reservoir.
Convolution storage	
CONVOLUTION	[mm] storage of water en route - for conceptual models with convolution
CONV_STOR	[mm] convolution sub-storage - 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 or MJ/m2] Cold content of snowpack
GLACIER_CC	[C] cold content of glacier
SOIL_TEMP	[C] temperature of soil
CANOPY_TEMP	[C] temperature of 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_AGE	[d] snow age, in days
SNOW_ALBEDO	[-] snow surface albedo
CROP_HEAT_UNITS	[-] cumulative crop heat units
Diagnostic Variables	
AET	[mm/d] actual evapotranspiration
RUNOFF	[mm/d] net runoff to surface water (includes baseflow)
Memory Variables	
CUM_INFIL	[mm] cumulative infiltration to topsoil
CUM_SNOWMELT	[mm] cumulative snowmelt
Transport Variables	
CONSTITUENT	[mg/m^2] mass density of chemical species or tracer
CONSTITUENT_SRC	[mg/m^2] chemical species or tracer cumulative source
CONSTITUENT_SW	[mg/m^2] chemical species dumped to surface water
CONSTITUENT_SINK	[mg/m^2] chemical species or tracer cumulative sink (e.g., decay)
![CONST]|[SV]	[mg/l] concentration of CONST in water store SV
(e.g.) !SNOWMELT|SOIL[1]	[mg/l] SNOWMELT concentration in SOIL[1]

Table D.2: All forcing functions currently available in Raven. This list of forcing functions is supported by the `:Data`, `:GriddedForcing`, `:MultiData`, `:CustomOutput`, and `:GaugeMultiData` 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]
RECHARGE	groundwater recharge rate over time step [mm/d]
TEMP_AVE	average air temp over time step/data interval [$^\circ$C]
TEMP_DAILY_AVE	average air temp over day (0:00-24:00) [$^\circ$C]
TEMP_MIN/TEMP_DAILY_MIN	minimum air temperature over day (0:00-24:00)[$^\circ$C]
TEMP_MAX/TEMP_DAILY_MAX	maximum air temperature over day (0:00-24:00)[$^\circ$C]
TEMP_MONTH_MAX	maximum air temp during month [$^\circ$C]
TEMP_MONTH_MIN	minimum air temp during month [$^\circ$C]
TEMP_MONTH_AVE	average air temp during month [$^\circ$C]
TEMP_AVE_UNC	uncorrected daily average air temp [$^\circ$C]
TEMP_MAX_UNC	uncorrected daily min air temp [$^\circ$C]
TEMP_MIN_UNC	uncorrected daily max air temp [$^\circ$C]
AIR_DENS	air density [kg/m^3]
AIR_PRES	air pressure [kPa]
REL_HUMIDITY	relative humidity [0..1]
ET_RADIA	uncorrected extraterrestrial shortwave radiation [MJ/m^2/d]
SHORTWAVE/SW_RADIA	Incoming shortwave radiation (uncorrected for albedo) [MJ/m^2/d]
SW_RADIA_NET	net shortwave radiation (albedo corrected) [MJ/m^2/d]
LW_RADIA_NET	net longwave radiation [MJ/m^2/d]
LW_INCOMING	incoming longwave radiation [MJ/m^2/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 [-]

Table D.3: All subbasin parameters currently available in Raven. These parameters may be specified in the :SubBasinProperties command in the .rvh file.

Parameter	[units] Description
In-catchment Routing Parameters	
TIME_TO_PEAK	[d] The time to peak of the unit hydrograph
TIME_CONC	[d] The time of concentration of the unit hydrograph
TIME_LAG	[d] The time lag of the unit hydrograph
NUM_RESERVOIRS	[-] The number of reservoirs used in the ROUTE_RESERVOIR_SERIES method
RES_CONSTANT	[1/d] A linear reservoir constant used to generate the unit hydrograph
GAMMA_SHAPE	[-] The Gamma unit hydrograph shape factor (for ROUTE_GAMMA method)
GAMMA_SCALE	[1/d] The Gamma unit hydrograph scale factor (for ROUTE_GAMMA method)
In-channel Routing Parameters	
Q_REFERENCE	[m^3/s] reference flow for the reach
MANNINGS_N	[-] Manning's coefficient for the reach; overrides channel profile value
SLOPE	[-] Slope for the reach; overrides channel profile value
DIFFUSIVITY	[m^3/s] Diffusivity for reach, overrides channel default value
CELERITY	[m/s] Celerity of flood wave in reach, overrides channel default value
Global parameters	
RAINSNOW_TEMP	[$^\circ C$] rain/snow halfway transition temperature
RAINSNOW_DELTA	[$^\circ C$] range of rain-snow transition zone (about RAINSNOW_TEMP)
SNOW_SWI	[-] water saturation fraction of snow
Other Parameters	
RAIN_CORR	[0..1] rain correction factor for subbasin (multiplier)
SNOW_CORR	[0..1] snow correction factor for subbasin (multiplier)
GAMMA_SHAPE_MULTIPLIER	shape factor multiplier, adjusts Raven-estimated values
TIME_TO_PEAK_MULTIPLIER	time to peak multiplier, adjusts Raven-estimated values

Table D.4: Required parameters for all model operation options. The asterisk* denotes the default algorithm for each method.

Option	Algorithms	Required Parameters
Interpolation		
:Interpolation	INTERP_FROM_FILE	:GaugeWeightsTable required
	INTERP_AVERAGE_ALL	-
	INTERP_NEAREST_NEIGHBOR*	-
	INTERP_INVERSE_DISTANCE	-
Routing		
:Routing	ROUTE_NONE ROUTE_DIFFUSIVE_WAVE* ROUTE_PLUG_FLOW ROUTE_STORAGE_COEFF ROUTE_MUSKINGUM ROUTE_MUSKINGUM_LAGGED ROUTE_MUSKINGUM_CUNGE ROUTE_HYDROLOGIC	Channel Geometry, Mannings n (ALL but ROUTE_NONE)
:CatchmentRoute	ROUTE_DUMP*	-
	ROUTE_GAMMA_CONVOLUTION	TIME_TO_PEAK or GAMMA_SCALE [GAMMA_SHAPE optional]
	ROUTE_TRI_CONVOLUTION	TIME_CONC, TIME_TO_PEAK
	ROUTE_RESERVOIR_SERIES	NUM_RESERVOIRS, RES_CONSTANT
	ROUTE_EXPONENTIAL	RES_CONSTANT
Evaporation		
:Evaporation	PET_NONE	-
	PET_FROMFILE	[gridded data or time series at gauge]
	PET_FROMMONTHLY [HBV]	:MonthlyAveEvaporation, :MonthlyAveTemperature
	PET_MONTHLY_FACTOR	FOREST_PET_CORR, FOREST_COVERAGE, :MonthlyEvapFactor
	PET_PENMAN_MONTEITH	MAX_HEIGHT,RELATIVE_HT,MAX_LAI, RELATIVE_LAI SPARSENESS, MAX_LEAF_COND
	PET_PENMAN_COMBINATION	MAX_HEIGHT,RELATIVE_HT
	PET_HAMON	-
	PET_HARGREAVES	:MonthlyMaxTemperature and :MonthlyMinTemperature
	PET_HARGREAVES_1985*	-
	PET_OUDIN	
	PET_TURC_1961	-
	PET_SIMPLE33 (Valiantzas et al 2006)	-
	PET_SIMPLE39 (Valiantzas et al 2006)	-
	PET_GRANGERGRAY	-
	PET_MOHYSE	MOHYSE_PET_COEFF
	PET_MAKKINK_1957	-
	PET_PRIESTLEY_TAYLOR	-
:OW_Evaporation	Same as :Evaporation	-
:OroPETCorrect	OROCORR_NONE*	-
	OROCORR_SIMPLELAPSE	
	OROCORR_HBV	
Radiation		
:SWRadiationMethod	SW_RAD_DATA	[gridded data or time series at gauge]
	SW_RAD_DEFAULT*	SLOPE, ASPECT
	SW_RAD_UBCWM	
:LWRadiationMethod	LW_RAD_DATA	[gridded data or time series at gauge]
	LW_RAD_DEFAULT*	FOREST_COVERAGE
	LW_RAD_UBC	FOREST_COVERAGE
:CloudCoverMethod	CLOUDCOV_NONE*	-
	CLOUDCOV_DATA	[gridded data or time series at gauge]
	CLOUDCOV_UBC	-

Table D.5: Required parameters for all model operation options (continued)

Option	Algorithms	Required Parameters
Precipitation		
:RainSnowFraction	RAINSNOW_DATA	[gridded data or time series at gauge]
	RAINSNOW_DINGMAN	RAINSNOW_TEMP
	RAINSNOW_HBV	RAINSNOW_TEMP, RAINSNOW_DELTA
	RAINSNOW_HARDER	-
	RAINSNOW_UBC	RAINSNOW_TEMP, RAINSNOW_DELTA
:PrecipIceptFract	PRECIP_ICEPT_USER	RAIN_ICEPT_PCT, SNOW_ICEPT_PCT
	PRECIP_ICEPT_LAI	RAIN_ICEPT_FACT,SNOW_ICEPT_FACT
	PRECIP_ICEPT_EXPLAI	-
:OroPrecipCorrect	OROCORR_NONE*	-
	OROCORR_UBC	:UBCPrecipLapseRates
	OROCORR_HBV	-
	OROCORR_SIMPLELAPSE	-
Temperature		
:OroTempCorrect	OROCORR_NONE*	-
	OROCORR_UBC	:UBCTempLapseRates,:ReferenceMaxTemperatureRange, ADIABATIC_LAPSE,WET_ADIABATIC_LAPSE
	OROCORR_HBV	ADIABATIC_LAPSE
	OROCORR_SIMPLELAPSE	ADIABATIC_LAPSE
Energy		
:PotentialMeltMethod	POTMELT_DEGREE_DAY*	MELT_FACTOR, DD_MELT_TEMP (opt)
	POTMELT_RESTRICTED	MELT_FACTOR, DD_MELT_TEMP (opt)
	POTMELT_DATA	[gridded data or time series at gauge]
	POTMELT_EB [Dingman]	-
	POTMELT_USACE	WIND_EXPOSURE
	POTMELT_HMETS	MIN_MELT_FACTOR, MAX_MELT_FACTOR, DD_AGGRADATION
	POTMELT_HBV	MIN_MELT_FACTOR,HBV_MELT_ASP_CORR, FOREST_COVERAGE,HBV_MELT_FOR_CORR, MELT_FACTOR
	POTMELT_UBC	MIN_SNOW_ALBEDO, FOREST_COVERAGE,ASPECT, :UBCSouthSWCorr,F0ERGY,:UBCNorthSWCorr
:SubdailyMethod	SUBDAILY_NONE*	-
	SUBDAILY_SIMPLE	-
	SUBDAILY_UBC	-
Atmospheric Variables		
:WindspeedMethod	WINDVEL_CONSTANT*	-
	WINDVEL_DATA	[gridded data or time series at gauge]
	WINDVEL_UBC	:UBCTempLapseRates (P0TEDL,P0TEDU, MAX_RANGE_TEMP), FOREST_COVERAGE
:RelativeHumidityMethod	RELHUM_CONSTANT*	-
	RELHUM_DATA	[gridded data or time series at gauge]
	RELHUM_MINDEWPT	-
:AirPressureMethod	AIRPRESS_BASIC*	-
	AIRPRESS_UBC	-
	AIRPRESS_DATA	[gridded data or time series at gauge]
	AIRPRESS_CONST	-
Temporal Interpolation		
:MonthlyInterpolationMethod	MONTHINT_UNIFORM	-
	MONTHINT_LINEAR_FOM	-
	MONTHINT_LINEAR_MID*	-
	MONTHINT_LINEAR_21	-

Appendix E

Database Files

The following section provides an overview of the .dat database files distributed with Raven source. These files contain information that is independent of the compiled model code but may be useful to users and pre/post-processing tools. This includes parameter ranges and default values (RavenParameters.dat), mapping of parameters used in each algorithm (RavenAlgParams.dat), and the mapping of process connection for each algorihtm (RavenProcessConnections.dat). These files are distributed with the Raven source code, and an unofficial version is also distributed with the RavenR package. These files are used by a number of RavenR routines.

A preview of the format for each of these three files is provided in the following sections. Note that these master files are to date incomplete and will be updated with each new addition to algorithms and parameters in Raven, and should not be treated as comprehensive files as of Raven v3.5.

E.1 RavenAlgParams.dat

The tabular file RavenAlgParams.dat contains a list of all of the parameter requirements of the Raven hydrologic process algorithms and other options. Each row corresponds to one parameter requirement for one algorithm. Example:

```
# algorithm name | algorithm type | parameter name | parameter class

PRECIP_RAVEN         Precipitation  FOREST_COVERAGE             LULT
PRECIP_RAVEN         Precipitation         POROSITY             SOIL
CANEVP_RUTTER           CanopyEvap  FOREST_COVERAGE             LULT
CANEVP_RUTTER           CanopyEvap     MAX_CAPACITY             LULT
CANEVP_RUTTER           CanopyEvap   TRUNK_FRACTION             LULT
...
SOILEVAP_HBV       SoilEvaporation   FIELD_CAPACITY             SOIL
SOILEVAP_HBV       SoilEvaporation         SAT_WILT             SOIL
SOILEVAP_HBV       SoilEvaporation  FOREST_COVERAGE             LULT
SOILEVAP_HBV       SoilEvaporation   PET_CORRECTION             SOIL
...
ROUTE_HYDROLOGIC     RoutingMethod     CHANNEL_GEOM         SUBBASIN
ROUTE_HYDROLOGIC     RoutingMethod       MANNINGS_N         SUBBASIN
...
```

E.2 RavenParameters.dat

The RavenParameters.dat is a tabular file which contains a complete list of all of the parameters in Raven, their parameter class, their units, a default parameter value and a reasonable range. Missing values are indicated with a -9999 flag. Other parameters may also have arbitrarily large numbers to indicate that an upper bound should be determined by the user. These values should be reviewed prior to use in calibration or before use as default values in a model.

```
#parameter name, class type, units, Autogen, default,  min,  max
FOREST_COVERAGE         LULT [0..1]    False        0     0     1
LAKE_PET_CORR           LULT [0..1]    False      1.0   0.0   2.0
...
POROSITY                SOIL [0..1]     True      0.4   0.0   1.0
VIC_ALPHA               SOIL    [-]    False    -9999 -9999 -9999
VIC_ZMAX                SOIL   [mm]    False    -9999   0.0 -9999
VIC_ZMIN                SOIL   [mm]    False    -9999   0.0 -9999
...
TIME_CONC           SUBBASIN    [d]    False        1     0   200
TIME_TO_PEAK        SUBBASIN    [d]    False        1     0   100
TIME_LAG            SUBBASIN    [d]     True      0.0   0.0   5.0
...
```

E.3 RavenProcessConnections.dat

The RavenProcessConnections.dat is a tabular file that specifies the list of 'from' and 'to' compartments for all of the algorithms in Raven. Each line corresponds to one connection.

```
#algorithm,        process,         from,              to
#------------------------------------------------------------
# PRECIPITATION
#------------------------------------------------------------
PRECIP_RAVEN Precipitation        ATMOS_PRECIP           SNOW
PRECIP_RAVEN Precipitation        ATMOS_PRECIP   PONDED_WATER
PRECIP_RAVEN Precipitation        ATMOS_PRECIP     DEPRESSION
PRECIP_RAVEN Precipitation        ATMOS_PRECIP        WETLAND
...
#------------------------------------------------------------
# SOIL PROCESSES
# -----------------------------------------------------------
INF_RATIONAL  Infiltration    PONDED_WATER        SOIL[0]
INF_RATIONAL  Infiltration    PONDED_WATER  SURFACE_WATER
INF_SCS       Infiltration    PONDED_WATER        SOIL[0]
INF_SCS       Infiltration    PONDED_WATER  SURFACE_WATER
...
#------------------------------------------------------------
# CONCEPTUAL PROCESSES
#------------------------------------------------------------
FLUSH_RAVEN   Flush         USER_SPECIFIED USER_SPECIFIED
RAVEN_DEFAULT LateralFlush  USER_SPECIFIED USER_SPECIFIED
SPLIT_RAVEN   Split         USER_SPECIFIED USER_SPECIFIED
...
```

Appendix F

Template Files

The following section provides template .rvi files. Note that for these files and for custom model configurations, the `:CreateRVPTemplate` command in the .rvi file (see section A.1.3) can be used to generate an empty rvp file which can be populated with parameter values by the user.

F.1 UBC Watershed Model 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

# --Hydrologic 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]
:HydrologicProcesses
   :SnowAlbedoEvolve  SNOALB_UBCWM
   :SnowBalance       SNOBAL_UBCWM     MULTIPLE      MULTIPLE
   # moves snowmelt to fast runoff
   :Flush             RAVEN_DEFAULT    PONDED_WATER  INT_SOIL2
     :-->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
   # from infiltration to routing
   :Flush             RAVEN_DEFAULT    SURFACE_WATER INT_SOIL2
     :-->Conditional    HRU_TYPE IS_NOT LAKE
   :GlacierInfiltration GINFIL_UBCWM   PONDED_WATER  MULTIPLE
   # soils really used as routing stores
   :Percolation       PERC_LINEAR_ANALYTIC INT_SOIL  INT_SOIL2
   :Percolation       PERC_LINEAR_ANALYTIC INT_SOIL2 INT_SOIL3
   :Baseflow          BASE_LINEAR      INT_SOIL3     SURFACE_WATER
   :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.

F.2 HBV-EC Emulation

To emulate the Environment Canada version of HBV-96 Hamilton et al. (2000); Lindström et al. (1997):

```
# ----------------------------------------------
# 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      ROUTE_TRI_CONVOLUTION

: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

# an oddity unique to HBV:
:LakeStorage SLOW_RESERVOIR

# --Hydrologic Processes-------------------------
:Alias       FAST_RESERVOIR SOIL[1]
:Alias       SLOW_RESERVOIR SOIL[2]

:HydrologicProcesses
  :SnowRefreeze      FREEZE_DEGREE_DAY  SNOW_LIQ        SNOW
  :Precipitation     PRECIP_RAVEN       ATMOS_PRECIP    MULTIPLE
  :CanopyEvaporation CANEVP_ALL         CANOPY          ATMOSPHERE
  :CanopySublimation CANEVP_ALL         CANOPY_SNOW     ATMOSPHERE
  :SnowBalance       SNOBAL_SIMPLE_MELT SNOW            SNOW_LIQ
    :Overflow        OVERFLOW_RAVEN     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
    :-->Conditional HRU_TYPE IS_NOT LAKE
  :SoilEvaporation   SOILEVAP_HBV       SOIL[0]         ATMOSPHERE
  :CapillaryRise     CRISE_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
  :LateralEquilibrate RAVEN_DEFAULT AllHRUs FAST_RESERVOIR 1.0
  :LateralEquilibrate RAVEN_DEFAULT AllHRUs SLOW_RESERVOIR 1.0
:EndHydrologicProcesses
```

See the Alouette2 tutorial example for a template .rvp file for HBV-EC emulation, indicating all required parameters.

```
# ------------------------------------------
# Raven Input file
# HBV-Light Emulation
# developed by Koen Jansen, Wageningen University
# ------------------------------------------
# --Simulation Details ------------------------
:StartDate    1991-10-01 00:00:00
:Duration     365
:TimeStep     1.0
:Method              ORDERED_SERIES
:SoilModel           SOIL_MULTILAYER  3

:Routing             ROUTE_NONE
:CatchmentRoute      ROUTE_TRI_CONVOLUTION
:Evaporation         PET_DATA
:RainSnowFraction    RAINSNOW_HBV
:PotentialMeltMethod POTMELT_DEGREE_DAY
:OroTempCorrect      OROCORR_HBV
:OroPrecipCorrect    OROCORR_HBV
:OroPETCorrect       OROCORR_HBV
:CloudCoverMethod    CLOUDCOV_NONE
:PrecipIceptFract    PRECIP_ICEPT_USER

# --Hydrologic Processes------------------------
:Alias TOPSOIL        SOIL[0]
:Alias FAST_RESERVOIR SOIL[1]
:Alias SLOW_RESERVOIR SOIL[2]
:HydrologicProcesses
  :SnowRefreeze      FREEZE_DEGREE_DAY  SNOW_LIQ       SNOW
  :Precipitation     PRECIP_RAVEN       ATMOS_PRECIP   MULTIPLE
  :SnowBalance       SNOBAL_SIMPLE_MELT SNOW           SNOW_LIQ
     :Overflow       OVERFLOW_RAVEN     SNOW_LIQ       PONDED_WATER
  :Infiltration      INF_HBV            PONDED_WATER   MULTIPLE
  :Flush             RAVEN_DEFAULT      SURFACE_WATER  FAST_RESERVOIR
  :-->Conditional HRU_TYPE IS_NOT LAKE
  :SoilEvaporation   SOILEVAP_HBV       TOPSOIL        ATMOSPHERE
  :CapillaryRise     CRISE_HBV          FAST_RESERVOIR TOPSOIL
  :Percolation       PERC_CONSTANT      FAST_RESERVOIR SLOW_RESERVOIR
  :Baseflow          BASE_POWER_LAW     FAST_RESERVOIR SURFACE_WATER
  :Baseflow          BASE_THRESH_POWER  FAST_RESERVOIR SURFACE_WATER
  :Baseflow          BASE_LINEAR            SLOW_RESERVOIR SURFACE_WATER
:EndHydrologicProcesses
```

F.3 GR4J Emulation

```
# -----------------------------------------------
# Raven Input file
# GR4J Emulation
# -----------------------------------------------
:StartDate           2000-01-01 00:00:00
:EndDate             2004-01-01 00:00:00
: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_NEIGE  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] 0.9
 :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.

F.4 Canadian Shield Configuration

A useful configuration in Canadian shield basins characterised by shallow soils atop rock, with ample exposed rock and lakes. Use the `:CreateRVPTemplate` command to generate the corresponding .rvp template file and determine what parameters are needed.

```
:StartDate           2003-10-01 00:00:00
:EndDate             2004-01-01 00:00:00
:TimeStep            1.0

:Method              ORDERED_SERIES
:InterpolationMethod NEAREST_NEIGHBOR

:SoilModel           SOIL_MULTILAYER  3

:Routing             ROUTE_DIFFUSIVE_WAVE
:CatchmentRoute      ROUTE_TRI_CONVOLUTION
:Evaporation         PET_HARGREAVES_1985
:OW_Evaporation      PET_HARGREAVES_1985
:SWCanopyCorrect     SW_CANOPY_CORR_STATIC
:RainSnowFraction    RAINSNOW_DINGMAN
:PotentialMeltMethod POTMELT_DEGREE_DAY
:PrecipIceptFract    PRECIP_ICEPT_LAI

:MonthlyInterpolationMethod MONTHINT_LINEAR_MID

:LakeStorage LAKE_STORAGE

# --Hydrologic Processes-------------------------
:Alias       SOIL0 SOIL[0]
:Alias       SOIL1 SOIL[1]
:Alias       SOIL2 SOIL[2]
:HydrologicProcesses
    :SnowRefreeze       FREEZE_DEGREE_DAY SNOW_LIQ      SNOW
    :Precipitation      PRECIP_RAVEN      ATMOS_PRECIP  MULTIPLE
    :CanopyEvaporation  CANEVP_MAXIMUM    CANOPY        ATMOSPHERE
    :CanopySublimation  CANEVP_MAXIMUM    CANOPY_SNOW   ATMOSPHERE
    :SnowBalance        SNOBAL_TWO_LAYER  MULTIPLE      MULTIPLE
    :Abstraction        ABST_FILL         PONDED_WATER  DEPRESSION
    :OpenWaterEvaporation OPEN_WATER_EVAP DEPRESSION    ATMOSPHERE
    :Infiltration       INF_HBV           PONDED_WATER  MULTIPLE
    :LakeRelease        LAKEREL_LINEAR    LAKE_STORAGE  SURFACE_WATER
    :SoilEvaporation    SOILEVAP_ROOT     SOIL0         ATMOSPHERE
    :Percolation        PERC_GAWSER_CONSTRAIN       SOIL0         SOIL1
    :Percolation        PERC_GAWSER_CONSTRAIN       SOIL1         SOIL2
    :Baseflow           BASE_THRESH_POWER SOIL0         SURFACE_WATER
    :Baseflow           BASE_THRESH_POWER SOIL1         SURFACE_WATER
    :Baseflow           BASE_THRESH_POWER SOIL2         SURFACE_WATER
:EndHydrologicProcesses
```

F.5 MOHYSE Configuration

A simple educational model developed at the Université du Québec à Montréal (Fortin and Turcotte, 2006) . Use the `:CreateRVPTemplate` command to generate the corresponding .rvp template file and determine what parameters are needed.

```
:StartDate              1958-08-01 00:00:00
:EndDate                2003-09-30 00:00:00
:TimeStep               1.0
:Method                 ORDERED_SERIES

:Routing                ROUTE_NONE
:CatchmentRoute         ROUTE_GAMMA_CONVOLUTION

:PotentialMeltMethod    POTMELT_DEGREE_DAY
:Evaporation            PET_MOHYSE
:RainSnowFraction       RAINSNOW_DATA
:DirectEvaporation

:SoilModel              SOIL_TWO_LAYER

:HydrologicProcesses
  :SoilEvaporation  SOILEVAP_LINEAR    SOIL[0]       ATMOSPHERE
  :SnowBalance      SNOBAL_SIMPLE_MELT SNOW          PONDED_WATER
  :Precipitation    RAVEN_DEFAULT      ATMOS_PRECIP  MULTIPLE
  :Infiltration     INF_HBV            PONDED_WATER  SOIL[0]
  :Baseflow         BASE_LINEAR        SOIL[0]       SURFACE_WATER
  :Percolation      PERC_LINEAR        SOIL[0]       SOIL[1]
  :Baseflow         BASE_LINEAR        SOIL[1]       SURFACE_WATER
:EndHydrologicProcesses
```

See the MOHYSE model example file distributed with the Raven tutorials for a template .rvp file for MOHYSE emulation, indicating all required parameters.

F.6 HMETS Configuration

HMETS is a relatively simple model developed at the École de technologie supérieure (Martel et al., 2017). You may use the `:CreateRVPTemplate` command to generate the corresponding .rvp template file and determine what parameters are needed.

```
:StartDate 1953-01-01  00:00:00
:EndDate   2009-12-31 00:00:00
:TimeStep   24:00:00

:PotentialMeltMethod     POTMELT_HMETS
:RainSnowFraction        RAINSNOW_DATA
:Evaporation             PET_OUDIN

:CatchmentRoute          ROUTE_DUMP
:Routing                 ROUTE_NONE

:SoilModel               SOIL_TWO_LAYER

:AllowSoilOverfill

:HydrologicProcesses
 :SnowBalance     SNOBAL_HMETS   MULTIPLE     MULTIPLE
 :Precipitation   RAVEN_DEFAULT  ATMOS_PRECIP MULTIPLE
 :Infiltration    INF_HMETS      PONDED_WATER MULTIPLE
   :Overflow      OVERFLOW_RAVEN SOIL[0]      CONVOLUTION[1]
 :Baseflow        BASE_LINEAR    SOIL[0]      SURFACE_WATER #interflow
 :Percolation     PERC_LINEAR    SOIL[0]      SOIL[1]       #recharge
   :Overflow      OVERFLOW_RAVEN SOIL[1]      CONVOLUTION[1]
 :SoilEvaporation SOILEVAP_ALL   SOIL[0]      ATMOSPHERE      #AET
 :Convolve        CONVOL_GAMMA   CONVOLUTION[0] SURFACE_WATER #surf. runoff
 :Convolve        CONVOL_GAMMA_2 CONVOLUTION[1] SURFACE_WATER #delay. runoff
 :Baseflow        BASE_LINEAR    SOIL[1]      SURFACE_WATER
:EndHydrologicProcesses
```

See the Salmon model example file distributed with the Raven tutorials for a template .rvp file for HMETS emulation, indicating all required parameters.

F.7 HYPR Configuration

HYPR is a version of HBV revised to support simulation on the Canadian Prairies (Ahmed et al., 2020). You may use the `:CreateRVPTemplate` command to generate the corresponding .rvp template file and determine what parameters are needed.

```
:StartDate          2002-10-01 00:00:00
:EndDate            2015-08-31 00:00:00
:Duration           4718
:TimeStep           24:00:00

# Model options
#------------------------------------------------------------------------
:CatchmentRoute      ROUTE_TRI_CONVOLUTION

:Evaporation         PET_FROMMONTHLY
:OW_Evaporation      PET_FROMMONTHLY
:SWRadiationMethod   SW_RAD_DEFAULT
:LWRadiationMethod   LW_RAD_DEFAULT
:RainSnowFraction    RAINSNOW_HBV
:PotentialMeltMethod POTMELT_HBV
:PrecipIceptFract    PRECIP_ICEPT_USER
:MonthlyInterpolationMethod  MONTHINT_LINEAR_21
:SoilModel           SOIL_MULTILAYER 3

# Soil Layer Alias Definitions
#------------------------------------------------------------------------
:Alias       FAST_RESERVOIR SOIL[1]
:Alias       SLOW_RESERVOIR SOIL[2]


# Hydrologic process order for HYPR Emulation
#------------------------------------------------------------------------
:HydrologicProcesses
  :SnowRefreeze      FREEZE_DEGREE_DAY  SNOW_LIQ        SNOW
  :Precipitation     PRECIP_RAVEN       ATMOS_PRECIP    MULTIPLE
  :CanopyEvaporation CANEVP_ALL         CANOPY          ATMOSPHERE
  :CanopySublimation CANEVP_ALL         CANOPY_SNOW     ATMOSPHERE
  :SnowBalance       SNOBAL_SIMPLE_MELT SNOW            PONDED_WATER
  :Infiltration      INF_HBV            PONDED_WATER    MULTIPLE
  :Flush             RAVEN_DEFAULT      SURFACE_WATER   PONDED_WATER
  :Abstraction       ABST_PDMROF        PONDED_WATER    DEPRESSION
  :Flush             RAVEN_DEFAULT      SURFACE_WATER   FAST_RESERVOIR
  :SoilEvaporation   SOILEVAP_HYPR      MULTIPLE        ATMOSPHERE
  :Baseflow          BASE_LINEAR        FAST_RESERVOIR  SURFACE_WATER
  :Baseflow          BASE_THRESH_STOR   FAST_RESERVOIR  SURFACE_WATER
:EndHydrologicProcesses
```

F.8 HYMOD/HYMOD2 Configuration

HYMOD is a popular conceptual model based upon the Probability Distributed Model (PDM) described in (Moore, 2007), and is documented in (e.g.,) (Wagener et al., 2001). The HYMOD2 model is a slightly revised variant documented by Roy et al. (2017). The user may use the `:CreateRVPTemplate` command to generate the corresponding .rvp template file and determine what parameters are needed.

```
:StartDate              2000-01-01 00:00:00
:EndDate                2001-01-01 00:00:00
:TimeStep               1.0
:Method                 ORDERED_SERIES

# Model options for HYMOD Emulation
#------------------------------------------------------------------------
:Routing              ROUTE_NONE
:CatchmentRoute       ROUTE_RESERVOIR_SERIES

:Evaporation          PET_HAMON
:OW_Evaporation       PET_HAMON
:SWRadiationMethod    SW_RAD_NONE
:LWRadiationMethod    LW_RAD_NONE
:CloudCoverMethod     CLOUDCOV_NONE
:RainSnowFraction     RAINSNOW_THRESHOLD
:PotentialMeltMethod POTMELT_DEGREE_DAY
:PrecipIceptFract     PRECIP_ICEPT_NONE

:SoilModel            SOIL_MULTILAYER 2

#------------------------------------------------------------------------
# Hydrologic process order for HYMOD/HYMOD2 Emulation
#
:HydrologicProcesses
  :Precipitation     PRECIP_RAVEN       ATMOS_PRECIP    MULTIPLE
  :SnowBalance       SNOBAL_SIMPLE_MELT SNOW            PONDED_WATER
  :Infiltration      INF_PDM            PONDED_WATER    MULTIPLE
  # 0.5 is the  HYMOD_ALPHA parameter
  :Flush             RAVEN_DEFAULT      SURFACE_WATER   SOIL[1]
0.5
  :SoilEvaporation   SOILEVAP_PDM       SOIL[0]         ATMOSPHERE
  # for HYMOD2, replace above with:
  # :SoilEvaporation SOILEVAP_HYMOD2    SOIL[0]         ATMOSPHERE
  :Baseflow          BASE_LINEAR        SOIL[1]         SURFACE_WATER
:EndHydrologicProcesses
```

F.9 AWBM Configuration

The Australian Water Balance Model (ABWM) is a low-parameter 4-compartment conceptual model described in Boughton (2004), here augmented to support simple treatment of snowmelt. The user may use the `:CreateRVPTemplate` command to generate the corresponding .rvp template file and determine what parameters are needed.

```
:StartDate              1994-01-01 00:00:00
:EndDate                1999-01-01 00:00:00
:TimeStep               1.0
:Method                 ORDERED_SERIES

# Model options for AWBM Emulation
#------------------------------------------------------------------------
:Routing              ROUTE_NONE
:CatchmentRoute       ROUTE_DUMP

:SWRadiationMethod    SW_RAD_NONE
:LWRadiationMethod    LW_RAD_NONE
:CloudCoverMethod     CLOUDCOV_NONE
:PrecipIceptFract     PRECIP_ICEPT_NONE

:Evaporation          PET_HAMON
:RainSnowFraction     RAINSNOW_THRESHOLD
:PotentialMeltMethod  POTMELT_DEGREE_DAY

:SoilModel            SOIL_MULTILAYER 4

#------------------------------------------------------------------------
# Hydrologic process order for AWBM Emulation
#
:HydrologicProcesses
  :Precipitation     PRECIP_RAVEN       ATMOS_PRECIP    MULTIPLE
  :SnowBalance       SNOBAL_SIMPLE_MELT SNOW            PONDED_WATER
  :Infiltration      INF_ABWM           PONDED_WATER    MULTIPLE
  :SoilEvaporation   SOILEVAP_ABWM      MULTIPLE        ATMOSPHERE
  :Baseflow          BASE_LINEAR        SOIL[3]         SURFACE_WATER
:EndHydrologicProcesses
```

Note that the baseflow parameter K in Boughton (2004) is equal to 1-`BASE_COEFF` here.

F.10 SAC-SMA Configuration

The Sacramento Soil Moisture Accounting model (SAC-SMA) is a spatially lumped model that divides the basin into lower and upper zones at different depths. The Raven implementation has seven soil layers, including layers for upper zone tension and free storage compartments, special storage corresponding to saturaed soil regions near a surface water body, and deep groundwater storage. The user may use the `:CreateRVPTemplate` command to generate the corresponding .rvp template file and determine what parameters are needed.

```
:StartDate              1954-01-01 00:00:00
:Duration               20819
:TimeStep               1.0
:Method                 ORDERED_SERIES

# Model options for SAC-SMA Emulation
#-----------------------------------------------------------------------
:PotentialMeltMethod     POTMELT_DEGREE_DAY
:RainSnowFraction        RAINSNOW_DATA
:Evaporation             PET_DATA
:CatchmentRoute          ROUTE_DUMP
:Routing                 ROUTE_NONE

:SoilModel               SOIL_MULTILAYER 7

:Alias UZ_T  SOIL[0]
:Alias UZ_F  SOIL[1]
:Alias LZ_T  SOIL[2]
:Alias LZ_PF SOIL[3]
:Alias LZ_PS SOIL[4]

#-----------------------------------------------------------------------
# Hydrologic process order for SAC-SMA Emulation
#
:HydrologicProcesses
  :SnowBalance           SNOBAL_SIMPLE_MELT   SNOW          PONDED_WATER
  :Precipitation         RAVEN_DEFAULT        ATMOS_PRECIP  MULTIPLE
  :SoilEvaporation       SOILEVAP_SACSMA      MULTIPLE      ATMOSPHERE
  :SoilBalance           SOILBAL_SACSMA       MULTIPLE      MULTIPLE
  :OpenWaterEvaporation  OPEN_WATER_RIPARIAN  SURFACE_WATER ATMOSPHERE
:EndHydrologicProcesses
```

F.11 Routing-only implementation

For a routing-only implementation, another model is used to generate the net runoff (all water reaching the surface water network, minus ET losses from water bodies, including baseflow). This is sent to Raven as a PRECIP input file, often as a gridded or station-based NetCDF input. Raven routes this water downstream through a network of rivers, lakes, and reservoirs.

```
:StartDate              2000-01-01 00:00:00
:EndDate                2010-12-31 00:00:00
:TimeStep               24:00:00

:CatchmentRoute         ROUTE_GAMMA_CONVOLUTION
:Routing                ROUTE_DIFFUSIVE_WAVE

:PrecipIceptFract       PRECIP_ICEPT_NONE
:PotentialMeltMethod    POTMELT_NONE
:SWRadiationMethod      SW_RAD_NONE
:OW_Evaporation         PET_NONE

:SoilModel              SOIL_ONE_LAYER

:HydrologicProcesses
  :Precipitation PRECIP_RAVEN   ATMOS_PRECIP  PONDED_WATER
  :Flush         RAVEN_DEFAULT  PONDED_WATER  SURFACE_WATER
:EndHydrologicProcesses
```

For an example routing-only setup, look at walkthrough number 6 of the Raven tutorials.

F.12 Blended v1 Configuration

The Blended model v1 configuration is a blended modelling approach in which five hydrologic process groups, specifically infiltration, quickflow, soil evapotranspiration, baseflow, and snow balance, have two or more process options. This blended model concept was introduced in Mai et al. (2020). The user may use the `:CreateRVPTemplate` command to generate the corresponding .rvp template file and determine what parameters are needed.

```
:StartDate    2000-01-01 00:00:00
:EndDate      2001-01-01 00:00:00
:TimeStep     1.0

:PotentialMeltMethod     POTMELT_HMETS
:RainSnowFraction        RAINSNOW_HBV
:Evaporation             PET_OUDIN

:CatchmentRoute          ROUTE_DUMP
:Routing                 ROUTE_NONE
:SoilModel               SOIL_MULTILAYER 3

:Alias DELAYED_RUNOFF CONVOLUTION[1]

:HydrologicProcesses
  :Precipitation   RAVEN_DEFAULT    ATMOS_PRECIP  MULTIPLE
  :ProcessGroup # infiltration group
    :Infiltration    INF_HMETS      PONDED_WATER  MULTIPLE
    :Infiltration    INF_VIC_ARNO   PONDED_WATER  MULTIPLE
    :Infiltration    INF_HBV        PONDED_WATER  MULTIPLE
  :EndProcessGroup CALCULATE_WTS 0.55556 0.5
  :Overflow        OVERFLOW_RAVEN    SOIL[0]      DELAYED_RUNOFF
  :ProcessGroup # quickflow group
    :Baseflow  BASE_LINEAR_ANALYTIC SOIL[0]       SURFACE_WATER
    :Baseflow  BASE_VIC             SOIL[0]       SURFACE_WATER
    :Baseflow  BASE_TOPMODEL        SOIL[0]       SURFACE_WATER
  :EndProcessGroup CALCULATE_WTS 0.55556 0.5
  :Percolation     PERC_LINEAR      SOIL[0]       SOIL[1]
  :Overflow        OVERFLOW_RAVEN   SOIL[1]       DELAYED_RUNOFF
  :Percolation     PERC_LINEAR      SOIL[1]       SOIL[2]
  :ProcessGroup #evaporation group
    :SoilEvaporation SOILEVAP_ALL      SOIL[0]    ATMOSPHERE
    :SoilEvaporation SOILEVAP_TOPMODEL SOIL[0]    ATMOSPHERE
  :EndProcessGroup CALCULATE_WTS  0.5
  :Convolve   CONVOL_GAMMA         CONVOLUTION[0] SURFACE_WATER
  :Convolve   CONVOL_GAMMA_2       DELAYED_RUNOFF SURFACE_WATER
  :ProcessGroup #baseflow group
    :Baseflow  BASE_LINEAR_ANALYTIC  SOIL[1]      SURFACE_WATER
    :Baseflow  BASE_POWER_LAW        SOIL[1]      SURFACE_WATER
  :EndProcessGroup CALCULATE_WTS  0.5
  :ProcessGroup #snow balance group
    :SnowBalance  SNOBAL_HMETS       MULTIPLE     MULTIPLE
    :SnowBalance  SNOBAL_SIMPLE_MELT SNOW         PONDED_WATER
    :SnowBalance  SNOBAL_HBV         MULTIPLE     MULTIPLE
  :EndProcessGroup CALCULATE_WTS 0.55556 0.5
:EndHydrologicProcesses
```

F.13 Blended v2 Configuration

The Blended model v2 configuration is a blended modelling approach in which five hydrologic process groups, specifically potential evapotranspiration, potential melt, quickflow, soil evapotranspiration, and baseflow, have two or more process options. This blended model concept was introduced in Mai et al. (2020), and the blended v2 configuration was produced by Chlumsky et al. (2024). The user may use the `:CreateRVPTemplate` command to generate the corresponding .rvp template file and determine what parameters are needed.

The blended v2 configuration below was developed for a lumped, daily model. Modifications for a semi-distributed, subdaily model are provided subsequently.

```
:StartDate    2000-01-01 00:00:00
:Duration     365
:TimeStep     1.0
:Method              ORDERED_SERIES

:SoilModel               SOIL_MULTILAYER 3

:Evaporation           PET_BLENDED
:BlendedPETWeights PET_GRANGERGRAY 0.55556 PET_HAMON 0.5 PET_PENMAN_MONTEITH

:PotentialMeltMethod           POTMELT_BLENDED
:BlendedPotMeltWeights POTMELT_HMETS 0.5 POTMELT_RESTRICTED

:RainSnowFraction        RAINSNOW_HBV        # RAINSNOW_DATA
:PrecipIceptFract        PRECIP_ICEPT_USER
:CatchmentRoute          ROUTE_DUMP
:Routing                 ROUTE_NONE

:Alias DELAYED_RUNOFF CONVOLUTION[1]

:HydrologicProcesses
  :Precipitation                RAVEN_DEFAULT           ATMOS_PRECIP
MULTIPLE
  :CanopyDrip                   CANDRIP_RUTTER          CANOPY
PONDED_WATER
  :Abstraction                  ABST_PERCENTAGE         PONDED_WATER
DEPRESSION
  :OpenWaterEvaporation         OPEN_WATER_EVAP         DEPRESSION
ATMOSPHERE
  :CanopyEvaporation            CANEVP_MAXIMUM          CANOPY
ATMOSPHERE
  :CanopySnowEvap               CANEVP_MAXIMUM          CANOPY_SNOW
ATMOSPHERE
  :Seepage                      SEEP_LINEAR             DEPRESSION
SOIL[1]
  :Infiltration                 INF_HMETS               PONDED_WATER
MULTIPLE
         :Overflow              OVERFLOW_RAVEN          SOIL[0]
DELAYED_RUNOFF
   :ProcessGroup #quickflow/interflow group
               :Baseflow        BASE_POWER_LAW          SOIL[0]
SURFACE_WATER
```

```
                :Baseflow        BASE_THRESH_POWER       SOIL[0]
SURFACE_WATER
  :EndProcessGroup CALCULATE_WTS 0.5
  :Percolation                   PERC_LINEAR             SOIL[0]
SOIL[1]         # recharge
  :CapillaryRise                 CRISE_HBV               SOIL[1]
SOIL[0]
  :Overflow                      OVERFLOW_RAVEN          SOIL[1]
DELAYED_RUNOFF
  :Percolation                   PERC_LINEAR             SOIL[1]
SOIL[2]         # loss to deep gw
  :CapillaryRise                 CRISE_HBV               SOIL[2]
SOIL[1]
  :ProcessGroup #evaporation/AET group
                :SoilEvaporation SOILEVAP_ALL            SOIL[0]
ATMOSPHERE
                :SoilEvaporation SOILEVAP_ROOT           SOIL[0]
ATMOSPHERE
                :SoilEvaporation SOILEVAP_SEQUEN         SOIL[0]
ATMOSPHERE
  :EndProcessGroup CALCULATE_WTS  0.55556 0.5
  :Convolve                      CONVOL_GAMMA
CONVOLUTION[0] SURFACE_WATER   # 'surface runoff'
  :Convolve                      CONVOL_GAMMA_2
DELAYED_RUNOFF SURFACE_WATER   # 'delayed runoff'
  :ProcessGroup #baseflow group
               :Baseflow         BASE_POWER_LAW          SOIL[1]
SURFACE_WATER
               :Baseflow         BASE_THRESH_POWER       SOIL[1]
SURFACE_WATER
  :EndProcessGroup CALCULATE_WTS  0.5
  :SnowBalance                   SNOBAL_HBV              MULTIPLE
MULTIPLE
:EndHydrologicProcesses
```

The modifications for a semi-distributed, subdaily blended v2 configuration include several minor adjustments. The first is to replace the catchment and in-channel routing with valid routines and also inlcude a subdaily method for temporal downscaling of forcings:

```
:CatchmentRoute         ROUTE_GAMMA_CONVOLUTION
:Routing                ROUTE_DIFFUSIVE_WAVE
:SubdailyMethod         SUBDAILY_SIMPLE
```

If orographic corrections are important, then adding the following is also recommended:

```
:OroTempCorrect         OROCORR_SIMPLELAPSE
:OroPrecipCorrect       OROCORR_HBV
```

The primary Hydrologic Processes block should be updated to the following, which removes the convolution routing for the faster convolution store, as is this practically replaced by catchment routing. The glacial processes (:GlacierMelt, :GlacierRelease, and glacial :Flush commands) may be removed if not rel-

evant.

```
:HydrologicProcesses
  :Precipitation                  RAVEN_DEFAULT           ATMOS_PRECIP
MULTIPLE
  :CanopyDrip                     CANDRIP_RUTTER          CANOPY
PONDED_WATER
  :Abstraction                    ABST_PERCENTAGE         PONDED_WATER
DEPRESSION
  :OpenWaterEvaporation           OPEN_WATER_EVAP         DEPRESSION
ATMOSPHERE
  :CanopyEvaporation              CANEVP_MAXIMUM          CANOPY
ATMOSPHERE
  :CanopySnowEvap                 CANEVP_MAXIMUM          CANOPY_SNOW
ATMOSPHERE
  :Seepage                        SEEP_LINEAR             DEPRESSION
SOIL[1]
  :Infiltration                   INF_HMETS               PONDED_WATER
MULTIPLE
         :Overflow                OVERFLOW_RAVEN          SOIL[0]
DELAYED_RUNOFF
   :ProcessGroup #quickflow/interflow group
                :Baseflow         BASE_POWER_LAW          SOIL[0]
SURFACE_WATER
                :Baseflow         BASE_THRESH_POWER       SOIL[0]
SURFACE_WATER
  :EndProcessGroup CALCULATE_WTS 0.5
  :Percolation                    PERC_LINEAR             SOIL[0]
SOIL[1]         # recharge
  :CapillaryRise                  CRISE_HBV               SOIL[1]
SOIL[0]
  :Overflow                       OVERFLOW_RAVEN          SOIL[1]
DELAYED_RUNOFF
  :Percolation                    PERC_LINEAR             SOIL[1]
SOIL[2]         # loss to deep gw
  :CapillaryRise                  CRISE_HBV               SOIL[2]
SOIL[1]
  :ProcessGroup #evaporation/AET group
                :SoilEvaporation  SOILEVAP_ALL            SOIL[0]
ATMOSPHERE
                :SoilEvaporation  SOILEVAP_ROOT           SOIL[0]
ATMOSPHERE
                :SoilEvaporation  SOILEVAP_SEQUEN         SOIL[0]
ATMOSPHERE
  :EndProcessGroup CALCULATE_WTS  0.55556 0.5
  :Flush                          RAVEN_DEFAULT
CONVOLUTION[0] SURFACE_WATER   # 'surface runoff'
  :Convolve                       CONVOL_GAMMA_2
DELAYED_RUNOFF SURFACE_WATER   # 'delayed runoff'
  #
  :Flush                          RAVEN_DEFAULT
PONDED_WATER   GLACIER
      :-->Conditional HRU_TYPE IS GLACIER
  :GlacierMelt                    GMELT_HBV
```

```
GLACIER_ICE    GLACIER
  :GlacierRelease                GRELEASE_HBV_EC
GLACIER        SURFACE_WATER
  #
  :ProcessGroup #baseflow group
                :Baseflow         BASE_POWER_LAW          SOIL[1]
SURFACE_WATER
                :Baseflow         BASE_THRESH_POWER       SOIL[1]
SURFACE_WATER
  :EndProcessGroup CALCULATE_WTS  0.5
  :SnowBalance                   SNOBAL_HBV              MULTIPLE
MULTIPLE
:EndHydrologicProcesses
```

F.14 EnKF implementation

In the .rvi file:

```
:EnsembleMode ENSEMBLE_ENKF 30
:SilentMode
```

The subbasin group `AllSubbasins` must be populated in the .rvh file and there must be `HYDROGRAPH`-type `:ObservationData` in subbasins 321, 604, and 1204 indicated below. In the .rve file:

```
:OutputDirectoryFormat ./outEnKF/mem_*

# EnKF parameters
#-----------------------------------------------------
:EnKFMode ENKF_CLOSED_LOOP
:SolutionRunName Alouette

# the forcing perturbations to apply
#-----------------------------------------------------
:ForcingPerturbation PRECIP   DIST_NORMAL   0.0 1.5 ADDITIVE
:ForcingPerturbation TEMP_AVE DIST_UNIFORM -0.5 0.5 ADDITIVE

# the states to adjust during assimilation
#-----------------------------------------------------
:AssimilatedState SNOW    ALLSubbasins
:AssimilatedState SOIL[0] AllSubbasins

# Observations to assimilate at 3 stream gauges
#-----------------------------------------------------
:AssimilateStreamflow 321
:AssimilateStreamflow 604
:AssimilateStreamflow 1204
:ObservationErrorModel STREAMFLOW DIST_NORMAL 1.0 0.07 MULTIPLICATIVE
```

Bibliography

Ahmed, M.I., Elshorbagy, A., Pietroniro, A., 2020. Toward simple modeling practices in the complex canadian prairie watersheds. Journal of Hydrologic Engineering 25, 04020024. doi:10.1061/(ASCE)HE.1943-5584.0001922.

Allen, R.G., Trezza, R., Tasumi, M., 2006. Analytical integrated functions for daily solar radiation on slopes. Agricultural and Forest Meteorology 139, 55–73.

Annandale, J., Jovanovic, N., Benadé, N., Allen, R., 2002. Software for missing data error analysis of penman-monteith reference evapotranspiration. Irrigation Science 21, 57–67.

Baker, D., Ruschy, D.L., Wall, D., 1990. The albedo decay of prairie snows. J. Appl. Meteor. 29, 179–187.

Barry, D.A., Parlange, J.Y., Li, L., Jeng, D.S., Crappert, M., 2005. Green Ampt approximations. Advances in Water Resources 28, 1003–1009.

Bennett, N.D., Croke, B.F., Guariso, G., Guillaume, J.H., Hamilton, S.H., Jakeman, A.J., Marsili-Libelli, S., Newham, L.T., Norton, J.P., Perrin, C., Pierce, S.A., Robson, B., Seppelt, R., Voinov, A.A., Fath, B.D., Andreassian, V., 2013. Characterising performance of environmental models. Environmental modelling and software 40, 1–20.

Bergstrom, S., 1995. Computer models of watershed hydrology. Water Resources Publications, Highlands Ranch, Colorado. chapter The HBV Model. pp. 443–476.

Beven, K.J., Kirkby, M.J., 1979. A physically based, variable contributing area model of basin hydrology. Hydrol Sci. 24, 43–69.

Bicknell, B., Imhoff, J., Kittle, J., Donigian, A., Johanson, R., 1997. Hydrological Simulation Program–Fortran, User's manual for version 11. Technical Report EPA/600/R-97/080. U.S. Environmental Protection Agency, National Exposure Research Laboratory. Athens, GA. 755pp.

Boughton, W., 2004. The Australian water balance model. Environmental Modelling and Software 19, 943–956. doi:https://doi.org/10.1016/j.envsoft.2003.10.007.

Brown, D.M., Bootsma, A., 1993. Crop Heat Units for Corn and Other Warm Season Crops in Ontario. Technical Report Fact sheet 93-119. Ontario Ministry for Food and Rural Affairs.

Burnash, R., Ferral, R., McGuire, R., 1973. A generalized streamflow simulation system: Conceptual modeling for digital computers. Technical Report. Joint Federal-State River Forecast Center, U.S. National Weather Service and California Department of Water Resources. Sacramento, CA. 204pp.

Chlumsky, R., Mai, J., Craig, J.R., Tolson, B.A., 2024. Advancement of a blended hydrologic model for robust model performance. Journal of Hydrologic Engineering 29, 04024033. doi:10.1061/JHYEFF.HEENG-6246.

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.

U.S. Dept. of Commerce, O.o.T.S., 1956. Snow Hydrology. Washington, D.C.

Craig, J., Brown, G., Chlumsky, R., Jenkinson, W., Jost, G., Lee, K., Mai, J., Serrer, M., Shafii, M., Sgro, N., Snowdon, A., Tolson, B., 2020. Flexible watershed simulation with the raven hydrological modelling framework. Environmental Modelling and Software 129, 104728. doi:https://doi.org/10.1016/j.envsoft.2020.104728.

Dingman, S., 2002. Physical Hydrology. Waveland Press Inc.

Fortin, V., Turcotte, R., 2006. Le modèle hydrologique MOHYSE. Technical Report. Université du Québec à Montréal. Note de cours pour SCA7420, Département des sciences de la terre et de l'atmosphere.

Granger, R.J., Gray, D.M., 1989. Evaporation from natural nonsaturated surfaces. Journal of Hydrology 111, 21–29.

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.

Hamilton, A., Hutchinson, D., Moore, R., 2000. Estimating winter streamflow using conceptual streamflow model. Journal of Cold Regions Engineering 14, 158–175.

Hamon, W., 1961. Estimating potential evapotranspiration. Journal of Hydraulics Division, Proceedings of the ASCE 871, 107–120.

Harder, P., Pomeroy, J., 2013. Estimating precipitation phase using a psychrometric energy balance method. Hyrdological Processes 27, 1901–1914.

Hargreaves, G., Samani, Z., 1982. Estimating potential evapotranspiration. Journal of the Irrigation and Drainage Division, ASCE 108, 225–230.

Hargreaves, G., Samani, Z., 1985. Reference crop evapotranspiration from temperature. Applied Engineering in Agriculture 1, 96–99.

Hedstrom, N.R., Pomeroy, J.W., 1998. Measurements and modelling of snow interception in the boreal forest. Hydrological Processes 12, 1611–1625.

Hock, R., Holmgren, B., 2005. A distributed surface energy-balance model for complex topography and its application to Storglaciären, Sweden. Journal of Glaciology 51, 25–36. doi:doi:10.3189/172756505781829566.

Jansen, K.F., Teuling, A.J., Craig, J.R., Dal Molin, M., Knoben, W.J.M., Parajka, J., Vis, M., Melsen, L.A., 2021. Mimicry of a conceptual hydrological model (hbv): What's in a name? Water Resources Research 57, e2020WR029143. doi:https://doi.org/10.1029/2020WR029143.

Jordan, R.E., 1991. A one-dimensional temperature model for a snow cover: Technical documentation for sntherm.89 .

Kutchment, L., Gelfan, A.N., 1996. The determination of the snowmelt rate and the meltwater outflow from a snowpack for modelling river runoff generation. Journal of Hydrology 179, 23–26.

Kuzmin, P.P., 1972. Melting of snow cover. Technical Report. Israel Progr. Sci. Translation. Jerusalem.

Leavesley, G., Lichty, R., Troutman, B., Saindon, L., 1983. Precipitation-Runoff Modeling System: User's manual. U.S. Geological Survey Water-Resources Investigations 83-4238, 207 p.

Leavesley, G., Stannard, L., 1995. Computer models of watershed hydrology. Water Resources Publications, Highlands Ranch, Colorado. chapter The Precipitation-Runoff Modeling System - PRMS. pp. 281–310.

Linacre, E., 1977. A simple formula for estimating evaporation rates in various climates, using temperature data alone. Agricultural Meteorology 18, 409–424.

Lindström, G., Johansson, B., Persson, M., Gardelin, M., Bergström, S., 1997. Development and test of the distributed hbv-96 hydrological model. Journal of hydrology 201, 272–288.

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, 621–633.

Mai, J., Craig, J.R., Tolson, B.A., 2020. Simultaneously determining global sensitivities of model parameters and model structure. Hydrology and Earth System Sciences 24, 5835–5858.

Mai, J., Craig, J.R., Tolson, B.A., 2022. The pie sharing problem: Unbiased sampling of n+ 1 summative weights. Environmental Modelling & Software 148, 105282.

Makkink, G.F., 1957. Testing the Penman formula by means of lysimeters. J. Inst. of Water Eng. 11, 277–288.

Mandel, J., 2006. Efficient Implementation of the Ensemble Kalman Filter. Report. Univ. of Colorado.

Marks, D., Dozier, J., 1992. Climate and energy exchange at the snow surface in the alpine region of the Sierra Nevada: 2. Snow cover energy balance. Water Resources Research 28, 3043–3054.

Martel, J., Demeester, K., Brissette, F., Poulin, A., Arsenault, R., 2017. HMETS - a simple and efficient hydrology model for teaching hydrological modelling, flow forecasting and climate change impacts to civil engineering students. International Journal of Engineering Education 34, 1307–1316.

Mekonnen, M.A., Wheater, H.S., Ireson, A., Spence, C., Davison, B., Pietroniro, A., 2014. Towards an improved land surface scheme for prairie landscapes. Journal of Hydrology 511, 105–116.

Monteith, J., 1965. The state and movement of water in living organisms. Academic Press Inc., New York. volume 17. chapter Evaporation and environment. pp. 205–234.

Moore, R.J., 2007. The PDM rainfall-runoff model. Hydrology and Earth System Sciences 11, 483–499.

Oudin, L., Hervieu, F., Michel, C., Perrin, C., Andréassian, V., Anctil, F., Loumagne, C., 2005. Which potential evapotranspiration input for a lumped rainfall-runoff model?: Part 2-Towards a simple and efficient potential evapotranspiration model for rainfall-runoff modelling. Journal of Hydrology 303, 290–306.

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., Andréassian, V., 2003. Improvement of a parsimonious model for streamflow simulation. Journal of Hydrology 279, 275–289.

Pomeroy, J.W., Gray, D.M., Brown, T., Hedstrom, N.R., Quinton, W.L., Granger, R., Carey, S., 2007. The cold regions hydrological model: A platform for basing process representation and model structure on physical evidence. Hydrological Processes , 2650–2667.

Prata, A., 1996. A new long-wave formula for estimating downward clear-sky radiation at the surface. Quarterly Journal of the Royal Meteorological Society 122, 1127–1151.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 1992. Numerical Recipes in C. Second ed., Cambridge University Press, Cambridge, USA.

Priestley, C., Taylor, R., 1972. On the assessment of surface heat flux and evaporation using large-scale parameters. Monthly Weather Review , 81–92.

Quick, M., 1995. Computer models of watershed hydrology. Water Resources Publications, Highlands Ranch, Colorado. chapter The UBC Watershed Model. pp. 233–280.

Quick, M., 2003. UBC Watershed Model Documentation. Technical Report. University of British Columbia.

Riley, J.P., Israelsen, E., Eggleston, K., 1972. Some approaches to snowmelt prediction. Actes du Colloque de Banff sur le role de la niege et de la glace en hydrologie, AISH Pub 2, 956–971.

Roy, T., Gupta, H.V., Serrat-Capdevila, A., Valdes, J.B., 2017. Using satellite-based evapotranspiration estimates to improve the structure of a simple conceptual rainfall–runoff model. Hydrology and Earth System Sciences 21, 879–896.

Rutter, A., Kershaw, K., Robins, P., Morton, A., 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.

Seibert, J., . HBV light version 2 user's manual. Department of Earth Sciences, Uppsala University.

Soil Conservation Service, 1986. Urban Hydrology for Small Watersheds, 2nd ed. Technical Report Tech. Release No. 55 (NTIS PB87-101580). U.S. Department of Agriculture. Washington, D.C.

Sicart, J.E., Pomeroy, J., Essery, R., Bewley, D., 2006. Incoming longwave radiation to melting snow: observations, sensitivity and estimation in northern environments. Hydrological processes 20, 3697–3708.

Turc, L., 1961. Evaluation de besoins en eau d'irrigation, ET potentielle. Ann. Agron. 12, 13–49.

U.S. Army Corps of Engineers, 1998. Engineering and Design: Runoff from Snowmelt. Technical Report. Washington, D.C.

Valiantzas, J.D., 2006. Simplified versions for the penman evaporation equation using routine weather data. Journal of Hydrology 331, 690–702.

Wagener, T., Boyle, D.P., Lees, M.J., Wheater, H.S., Gupta, H.V., Sorooshian, S., 2001. A framework for development and application of hydrological models. Hydrology and Earth Systems Science 5, 13–26.

Wang, Y.H., Broxton, P., Fang, Y., Behrangi, A., Barlage, M., Zeng, X., Niu, G.Y., 2019. A wet-bulb temperature-based rain-snow partitioning scheme improves snowpack prediction over the drier western united states. Geophysical Research Letters 46, 13825–13835.

Williams, J., 1969. Flood routing with variable travel time or variable storage coefficients. Trans. ASAE 12, 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, 2717–2728.

Yassin, F., Razavi, S., Elshamy, M., Davison, B., Sapriza-Azuri, G., Wheater, H., 2019. Representation and improved parameterization of reservoir operation in hydrological and land-surface models. Hydrology and Earth System Sciences 23, 3735–3764.

Yin, X., 1997. Optical air mass: Daily integration and its applications. Meteorology and Atmospheric Physics 63, 227–233. Doi:10.1007/BF01027387.

To do...

- ☐ 1 (p. 46): Add PERC ASPEN routine / parameter
- ☐ 2 (p. 81): Add unit hydrograph intercomparison figure
- ☐ 3 (p. 83): find RobertsonEtAl1995 reference
- ☐ 4 (p. 84): Muskingum citations
- ☐ 5 (p. 101): Sub-daily temperature orographic and lapsing temp ranges not yet described
- ☐ 6 (p. 151): Forcing estimator code development section
- ☐ 7 (p. 172): Create a table for 'Required Parameters for Hydrologic Processes Options'