Multi-Scale Routing Setup

Author: Will Burke

Date: Fall 2019

Notes

* Download and use RHESSysPreprocessing R package for setup. Package is included along with RHESSys via RHESSys git (<https://github.com/RHESSys/RHESSys/tree/Multiscale/RHESSysPreprocessing>)
* Begin setup normally with a template and associated input maps
* Setup/modify/write your rules file. The “rules file” is needed for multiscale routing, and it defines the setups of patch families and patches within them. A rule ID will at minimum have the information on the number of patches in the patch family (*subpatch\_count*), and the percent coverage of each of the constituent patches (*pct\_family\_area*). Additional information, to overwrite any information included in the template can be included specifically for each patch, and basics such as the cover type should probably be included with any use of multiscale routing. See term definitions and example rules file.
* In the worldfile template, add a patch level state variable called “asp\_rule” and have it point to the input map or value being used to define your patch family rules. In this way either a single patch family rule, or many different patch family rules, varying spatially can be used. Example: “asp\_rule mode rule\_map” or “asp\_rule value 1.0”
* When running the RHESSysPreprocess function, follow existing instructions, and add the “asprules” argument, with it set to your rules file, e.g. asprules = "rules/p301.rules"

Running RHESSys

* Run multiscale routing by using the RHESSys github and specifically the “Multiscale” branch (this will create RHESSys with a version=>7.1.1). When running RHESSys from the command line, include the multiscale routing argument: “-msr” which has no arguments to it and will on its own trigger multiscale routing to run. If desired, the verbose flag being used to debug multiscale routing is negative six, i.e. “-v -6”.

Output

* In addition to existing patchIDs, familyIDs will indicate the overarching families that individual patches belong to.
* Rz\_transfer, unsat\_transfer, and sat\_transfer are new outputs at the patch level that capture the amount being laterally transferred to other patches within the patch family (positive values of gains, negative for loses).

Definitions:

*ID* = rule identification number

*subpatch\_count =* the number of patches in the patch family, must occur in the rules doc above the patch level

*\_patch* = delineates where patch (family) level state variables begin

*pct\_family\_area* = portion of the patch family area made up by each constituent patch

*\_canopy\_strata* = delineates where canopy strata level variables begin