HydropowerRelease

This function calculates the additional outflow necessary to satisfy an unmet load (energy requirement) while not causing additional downstream flooding.

Description	Calculates the additional outflow necessary to meet an unmet load, if any exists. The function limits the additional release to ensure that additional downstream flooding does not occur.	
Type	LIST { LIST { SLOT, NUMERIC, OBJECT } }	
Arguments	**Type**	**Meaning**
1	STRING	the name of the computational subbasin
Evaluation	The function does the following: 1. Prioritizes the power reservoirs in the basin according to the relative Load shortage using the equation below. $Shortage = \frac{Load - Energy}{Load}$ If Load is unknown because the Seasonal Load Time method is selected, it is calculated. The calculated shortage then is a value less than one. The reservoirs with the highest values are first, the lowest reservoirs last. 2. Runs the selected Additional Hydropower Release method on the each power reservoir in the subbasin to calculate the proposed additional release required to meet the load within outflow constraints. 3. In order of priority, hypothetically makes the additional release, visits downstream control points until it reaches a tandem reservoir or the end of the subbasin, whichever comes first. If the release causes (additional) flooding at a control point, it reduces the release until flooding is not caused or the release becomes zero. Resulting releases are then hypothetically routed to downstream control points to make adjustments to their available space hydrographs. A control point's available space hydrograph (in units of flow projected into the future based on the routing coefficients on the control point) is calculated as: $\text{Available Space} = \text{Regulation Discharge} - (\text{Inflow} + \text{forecasted Local Inflow})$ Inflow includes the value of the Inflow slot (at the time of the last dispatch) and the additional inflow resulting from the hypothetical additional releases from upstream reservoirs. It also contains the proposed flood control release hydrograph from the last pass of the flood control method. Note: Note, if the "Releases Not Constrained by Flooding" method is selected in the "Hydropower Flooding Exception" category on the control point ((Objects.pdf, Section 9.1.16.2) , the control point is ignored, i.e. flooding is allowed at that control point.	
Evaluation (cont)	4. Once all power reservoirs have been visited (in priority order), the HydropowerRelease function returns to the calling rule. For each reservoir in the subbasin, three triplets may be returned: • Additional Power Release: This is the additional release to meet the load. If the proposed release is positive, but the additional power release was constrained to zero, the triplet {reservoir.Additional Power Release, 0.0, reservoir} will be returned. If the proposed release is zero, the Additional Power Release (of zero) triplet will not be returned. • Outflow: If the new Outflow is the same as the existing Outflow, no Outflow triplet is returned because the value of the Outflow slot will not change as a direct result of this rule. Otherwise the value for outflow is the existing Outflow plus the additional power release. • Load: the Load triplet is only returned if the "Seasonal Load Time" method ((Objects.pdf, Section 16.1.33.7)) is selected on the reservoir. The calling rule is expected to make the assignments of the values to the slots	
Comments	This function is dependent on having executed the predefined function FloodControl() on a computational subbasin using the Operating Level Balancing method. This flood control method operation sets up the network topology and necessary data. HydropowerRelease requires that the reservoirs in the subbasin have already dispatched and have valid values in the Regulation Discharge, Outflow, Energy, and Load slots. Use of this function for USACE-SWD: (USACE_SWD.pdf, Section 3.9.3) .	

Syntax Example:

```
HydropowerRelease("Flood Basin") where "Flood Basin" contains Res1 and Res2.
```

Return Example:

```
{ {"Res1.Additional Hydropower Release", 63.32 "cms", "res1"},
{"Res1.Outflow", 63.32 "cms", "Res1"},
{"Res2.Additional Hydropower Release", 23.20 "cms", "Res2"}
{"Res2.Outflow", 32.02 "cms", "Res2"},
{"Res2.Load", 2.1 "MWH", "Res2"} }
```

Use Examples:

```
FOREACH (LIST triplet IN HydropowerRelease( "Flood Basin" )) DO
( triplet<0> )[] = triplet<1>
ENDFOREACH
```

Hypothetical Simulation Overview

This section presents an overview of a set of functions that allow the user to run a "hypothetical simulation" where:

• a limited number of objects on the workspace are involved

• the simulation has no side-effects, i.e., after simulation the workspace is exactly as before

• the objects involved initially have their current values, except for those values that the user provides as "fixed values" to the hypothetical simulation

simulation

• at least one and possibly more values resulting from the hypothetical simulation are available for use within rulebased simulation

Why would you want to do this? Lets consider the following example. Suppose that you would like to maintain a minimum flow of 100 cfs at some point in the River Y. Many miles upstream from this point you can control the outflow from Reservoir X. One question you might ask is: what is the release from X which will lead to the 100 cfs flow at the point of concern? A related but simpler question is: If I release 200 cfs from reservoir X, what will be the flow at the point of concern?

Even the answer to the second question can't be easily predicted; you might have to take into consideration many hydrologic inflows and flow-dependent physical processes like lags, losses, and diversions through different sections of River Y. you might even require that you know the release over some extended period of time in order to be able to determine the flow in the Y at a single time. At any rate, this is exactly the sort of computation performed by the objects in a RiverWare simulation.

On the other hand, answering the first question requires not only knowing the physical consequences of outflow from X, but a search for the release which has the target consequence. The target consequence cannot just be set and allowed to solve upstream because there are routing algorithms can only solve in the downstream direction.

The following functions can be used to do hypothetical simulations:

• **HypSim** - perform a hypothetical simulation with user specified values and returns user-requested result values. Click (HypSim) for more information on this function.

• **HypLimitSim** - perform hypothetical simulations iteratively to find a value which, when set on a given slot, will lead to another slot achieving but not exceeding a limiting value. Click (HypLimitSim) for more information on this function.

• **HypLimitSimWithStatus** - Same as HypLimitSim but with information on whether a satisfying value was found or not. Click (HypLimitSimWithStatus) for more information on this function.

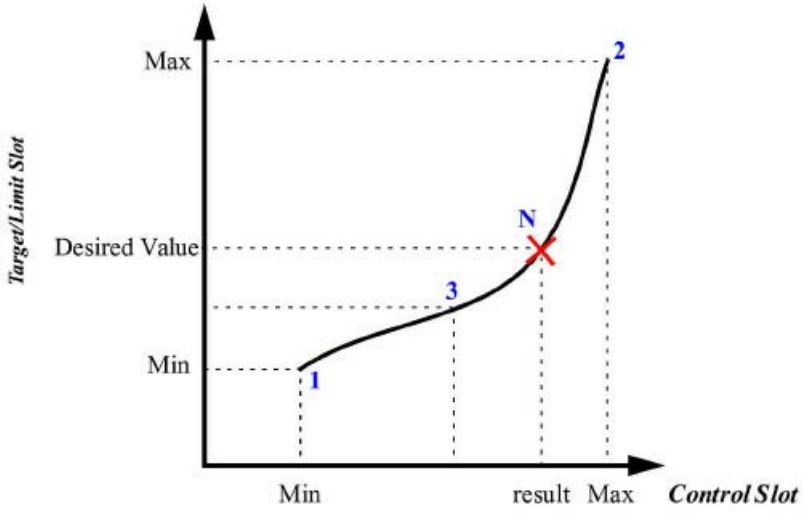
• **HypTargetSim** - perform hypothetical simulations iteratively to find a value which, when set on a given slot, will lead to a desired value on another slot. Click (HypTargetSim) for more information on this function.

• **HypTargetSimWithStatus** - Same as HypTargetSim but with information on whether a satisfying value was found or not. Click (HypTargetSimWithStatus) for more information on this function.

Each hypothetical simulation function proceeds in the following manner. The first time the function is called, all of the objects and links in the specified subbasin are cloned. This copies each object including all of the slots and values. This cloning is necessary as all subsequent computations are performed on the cloned objects, thereby not affecting the real objects on the workspace. Once the objects are cloned, any values specified in the arguments as "fixed values" are set on the objects. Remember, the function also copied all its data at the time it was cloned, so the fixed values are values that are to be set on the object that are not already there.

Then, the hypothetical simulation performs the computations described for that function. For HypSim, the cloned objects will dispatch to simulate the effects of the fixed values. A list containing the values of the specified slots at the specified datetimes will be returned. The calling rule/function can then use these results in its computation.

HypLimitSim, HypLimitSimWithStatus, HypTargetSim, and HypTargetSimWithStatus perform an iterative solution. For each of these functions, you provide min and max values for a control slot and a target or limit of a downstream slot. The computations then boil down to univariate zero-finding, where each evaluation is a hypothetical simulation with different inputs. The solution is found using the bisection method as shown in the following figure; it simulates using the min control slot value (1), then the max control slot value (2), then bisects between the two(3). It continues bisecting until the tolerance or desired accuracy of the limit/target slot is met and a solution is found (N.) If the result of any try is outside of the range of previous tries, then a warning message is issued saying that the function is non-monotonic. There are either multiple solutions or the function is not well behaved. This typically leads to convergence issues.



Following are some notes to consider about hypothetical simulation and particularly the iterative hypothetical simulations:

Following are some notes to consider about hypothetical simulation and particularly the iterative hypothetical simulations:

• These are hypothetical simulations, not rulebased simulations. You can set fixed values in the arguments but they cannot change once the function is executed.

• Hypothetical simulation does not support accounting or optimization functionality.

• Each iteration within hypothetical simulation is setting the control value to a new value and re-simulating the system. Some object methods may have a dependence on previous solutions that will lead to different results. Also, slot convergence can be an issue. If setting a new value on a slot does not propagate down the system because convergence is too loose, then the target/limit slot value may never be achieved.

• Use the "WithStatus" version of the function if you suspect that the function may not work in all cases but you still need a result. If these functions do not find a result, the closest value is returned along with the status. But, make sure to check the status in the calling rule or function, don't just use the result blindly. Use the "without status" version if you reasonably expect there to be a solution. Then, if there is a problem with the computation and no solution is found, the run is stopped and a message is posted.

• For the iterative functions, if there are multiple timesteps involved, the control slot is always set to the same value for all timesteps in the hypothetical simulation range.

• These hypothetical simulation functions are expensive in terms of run time. Following are some approaches to limit slow down from these functions:

- Limit their use as much as possible.

- If you need to call a function multiple times per timestep with the same arguments, create a helper function with no arguments; functions with no arguments are executed once per timestep and the results are cached for later use.

- Only include the relevant objects in the subbasin; cloning objects and copying values is time and memory expensive.

- Only include as many "fixed values" as necessary. If you are only solving for the target slot value at t+2, don't include fixed values from t through t+7. This will lead to unnecessary dispatching of the cloned objects.