RiverWare

Technical Documentation Version 7.3

Optimization

Center for Advanced Decision Support for
Water and Environmental Systems (CADSWES)
UNIVERSITY OF COLORADO **BOULDER**

Optimization
Table of Contents

Optimization

This document describes RiverWare's Optimization solver. First it provides an overview of Optimization and how it works in RiverWare. Then it gives describes details about the preemptive linear goal programming approach used by RiverWare, how Optimization interacts with Simulation and Rulebased Simulation, and how variables and constraints (policy) are formulated for RiverWare Optimization.This is followed by a section describing the methods and slots on each object that apply for Optimization. Finally, there are sections on creating, analyzing and debugging a RiverWare Optimization model.

1. Introduction and Overview

RiverWare's Optimization solver provides a different solution approach. Instead of solving a model on a timestep-by-timestep basis, Optimization provides a single global solution over all objects and all timesteps. Typically Optimization is used to determine the "best" solution in terms of a stated objective, such as maximizing the value of hydropower over the run period, while satisfying higher-priority policy constraints.

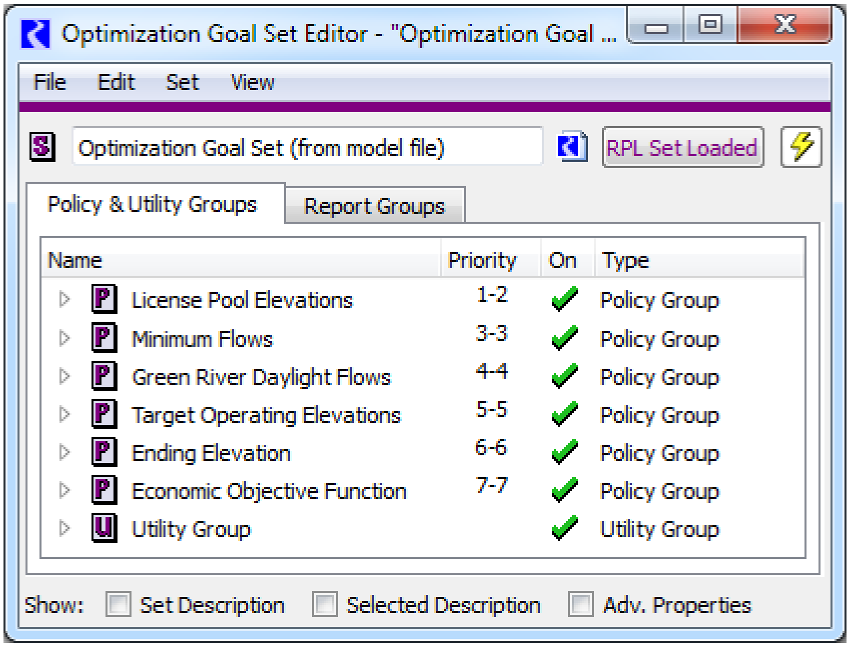
1.1 What is Optimization?

1.1.1 System-wide Approach

Simulation and Rulebased Simulation solve each object individually, one timestep at a time. Optimization, on the other hand, provides a global solution across all objects and across all timesteps. This allows the solution to trade off objectives both spatially and over time. For example, the model can "look ahead" to a peak price period and send water from upstream reservoirs to a downstream reservoir with high generating capacity early in the run so that it can generate at maximum capacity during peak prices. At the same time, it will assure that the upstream reservoirs do not release so much water that they will violate their elevation targets by the end of the run based on inflow forecasts.

Hydropower objectives are a common focus of Optimization models. RiverWare contains some built-in functionality that assists in formulating common hydropower objectives; however there is nothing that limits RiverWare Optimization objectives to be related to hydropower, nor is hydropower required to be included in an Optimization model. Optimization models could just as easily optimize a water management objective unrelated to hydropower.

1.1.2 Preemptive Linear Goal Programming



Name	Priority	On	Type
License Pool Elevations	1-2	✓	Policy Group
Minimum Flows	3-3	✓	Policy Group
Green River Daylight Flows	4-4	✓	Policy Group
Target Operating Elevations	5-5	✓	Policy Group
Ending Elevation	6-6	✓	Policy Group
Economic Objective Function	7-7	✓	Policy Group
Utility Group		✓	Utility Group

The Optimization solution in RiverWare is a preemptive linear goal program. The user expresses the policy of the system in a set of prioritized goals. These goals contain either constraints or objectives. A linear program (LP) is solved at each priority level, beginning at the highest priority. If the goal contains a set of soft constraints, the objective function for the LP is to maximize the satisfaction of the constraints. The objective value (level of satisfaction) will not be reduced when the LPs are solved for lower priority goals. In other words, once a constraint has been satisfied at a high priority, it will not be violated to meet a lower priority constraint or objective. In this way, as the solution progresses through the priorities, it shrinks the solution space or removes degrees of freedom from the solution. Typically the final priority is a maximize or minimize objective, such as maximizing the total value of hydropower over the run period. The objective will be maximized/minimized within the reduced solution space after maximizing the satisfaction of all higher priority constraints.

RiverWare provides three solution approaches in the case that not all soft constraints within a goal cannot be satisfied due to hydrologic conditions and/or higher priority constraints. The first approach, Summation, minimizes the total deviation from the constraint values. The second approach, Maximin, minimizes the single largest deviation (technically it maximizes the lowest satisfaction). The third approach, Repeated Maximin, minimizes the single largest deviation. Then it freezes that value and minimizes the next largest deviation, and so on until it has either minimized all deviations or satisfied all remaining constraints. In practice, the Repeated Minimax approach is most-often used.

1.1.3 Optimization Variables

An optimization problem consists of a set of equations that include linear combinations of optimization variables. In RiverWare, some Optimization variables are directly related to slots. For example, Outflow and Storage on a reservoir object are common variables. Each of these slots at each timestep corresponds directly to an individual variable in the Optimization problem. In some cases slots correspond to a linear combination of variables. For example, Power is typically defined by a piecewise linear function of Turbine Release. Each individual variable, or “piece,” is multiplied by the slope of that segment of the power curve. The sum of the pieces is equal to the total Turbine Release. Some variables have no relation to a slot in RiverWare. For example, the satisfaction level of each Repeated Maximin iteration is added to the Optimization problem as a variable. RiverWare decides automatically which variables to add to the Optimization problem based on what is referenced in the Optimization Goal Set.

1.1.4 Physical Constraints

RiverWare automatically adds all relevant physical constraints to the Optimization problem. For example, the mass balance constraint for a reservoir is

$$\mathrm{Storage}_t = \mathrm{Storage}_{t-1} + \mathrm{Inflow}_t - \mathrm{Outflow}_t + \sum \mathrm{Gains}_t - \sum \mathrm{Losses}_t$$

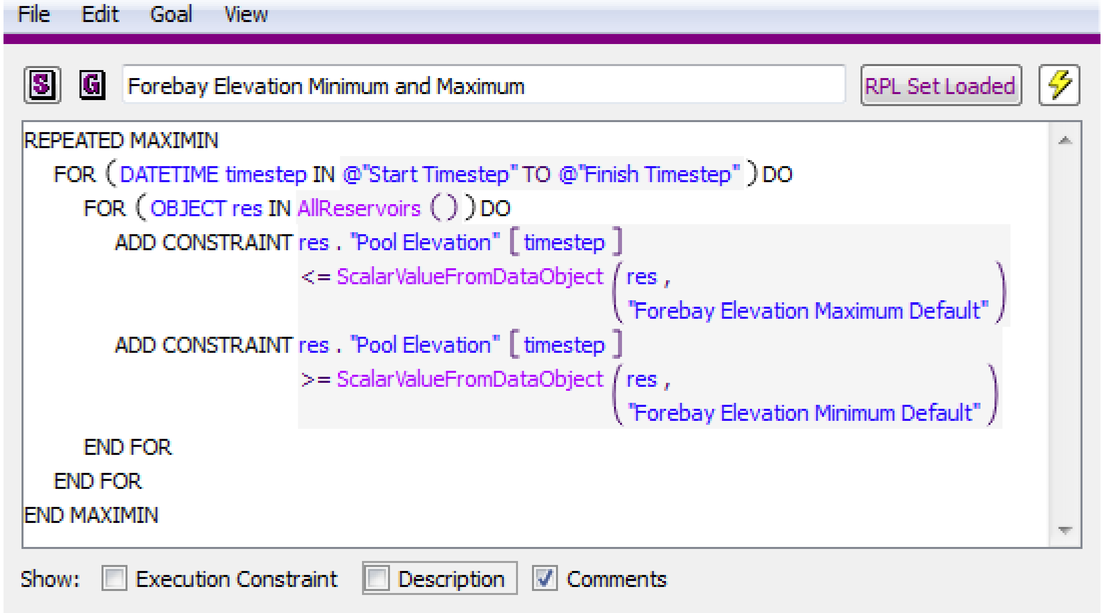
This is the defining constraint for the Storage, Inflow and Outflow variables at time *t*. RiverWare determines which individual variables to include in Gains and Losses depending on the method selection on the reservoir. For example, Evaporation could be an additional loss variable in the mass balance constraint. For a reach, the defining physical constraints are determined by the selected routing method.

RiverWare only adds physical constraints as necessary based on the policy in the Optimization Goal Set. In this way it minimizes the size of the Optimization problem in order to achieve maximum computational efficiency. Physical constraints are always treated as hard constraints.

1.1.5 Prioritized Policy

The policy of a river basin is expressed by the user in an Optimization Goal Set. Like rulesets in Rule-based Simulation, the Goal Set is formulated in RiverWare Policy Language (RPL). Also similarly to a ruleset, the goals in a Goal Set are expressed in priority order. For example, license minimum and maximum pool elevation constraints for reservoirs might be the highest priority goal. These are goals that would essentially never be violated. Lower priority goals might include target operating elevation range constraints. These would represent elevations that should be maintained under normal operating conditions but might be exceeded in extreme scenarios such as flood control operations. Goals with a Minimize or Maximize objective tend to be at the lowest priority, after all policy constraints have been satisfied to the greatest extent possible.

A user has the flexibility to formulate any type of constraint or objective they desire as long as the expressions are linear combinations of variables. Constraints can include average values over multiple timesteps or summations over multiple slots on multiple objects. Similar constraints for multiple objects can be grouped into a single constraint statement. For example, a single maximum elevation constraint can be added for all reservoirs in a model by creating a loop over a list of reservoirs and referencing the maximum elevations from a data object slot corresponding to each reservoir.



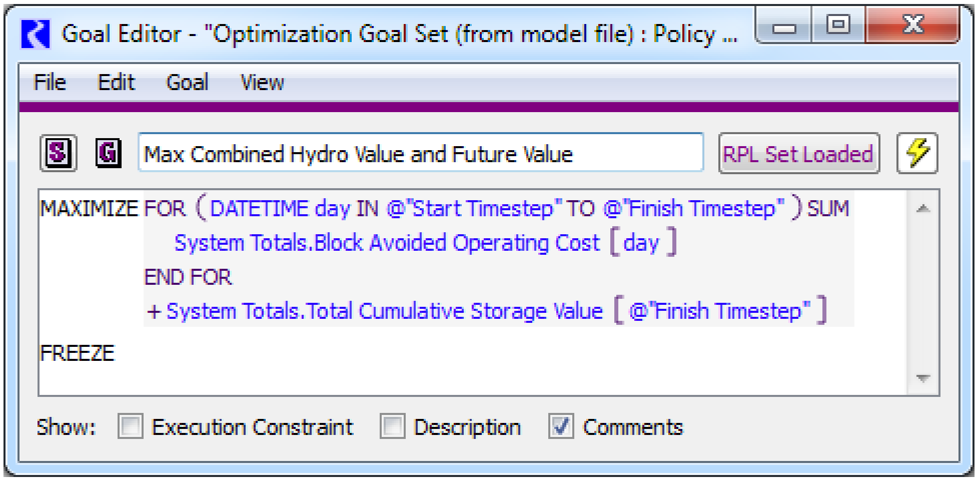
Forebay Elevation Minimum and Maximum

```
REPEATED MAXIMIN
    FOR ( DATETIME timestep IN @"Start Timestep" TO @"Finish Timestep" ) DO
        FOR ( OBJECT res IN AllReservoirs ( ) ) DO
            ADD CONSTRAINT res . "Pool Elevation" [ timestep ]
                           <= ScalarValueFromDataObject ( res ,
                                                          "Forebay Elevation Maximum Default" )
            ADD CONSTRAINT res . "Pool Elevation" [ timestep ]
                           >= ScalarValueFromDataObject ( res ,
                                                          "Forebay Elevation Minimum Default" )
        END FOR
    END FOR
END MAXIMIN
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

Conditional statements can be added for the inclusion of constraints in the form of If-Then statements. For example, seasonal constraints might apply only if the run period is within specified dates, or Special Operations constraints might only apply if data are present in a certain slot.

Objectives allow for a trade-off between present value and future value. An objective might include the combined value of the hydropower generated during the run and the value of energy in storage (future potential value) at the end of the run.


Goal Editor - "Optimization Goal Set (from model file) : Policy ...
File
Edit
Goal
View
S
G
RPL Set Loaded

Max Combined Hydro Value and Future Value

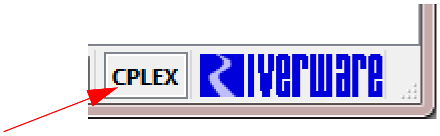
```
MAXIMIZE FOR ( DATETIME day IN @"Start Timestep" TO @"Finish Timestep" ) SUM
              System Totals.Block Avoided Operating Cost [ day ]
         END FOR
         + System Totals.Total Cumulative Storage Value [ @"Finish Timestep" ]
FREEZE
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

1.2 Why Consider Using Optimization?

Optimization is valuable when trying to determine a “best” solution as opposed to just evaluating a solution. Also, because the Optimization solution is global in space and time, it is useful for planning current operations in manner that leaves the system in a good position to meet upcoming conditions. Complex systems can require a great deal of trial and error to approach a “best” solution, and it might not be obvious how changes to one part of the system will affect other parts of the system or operations in the future. Optimization provides an efficient means to find the best operations for near term objectives without jeopardizing the ability to meet constraints in the future.

1.3 CPLEX License



RiverWare Optimization uses a third-party solver, IBM ILOG CPLEX, which is bundled with RiverWare. To access the solver, your RiverWare license must specifically allow you access to the solver library. Please contact riverware-support@colorado.edu for more information. When your license allows you access to the solver, a CPLEX icon is displayed in the lower right corner of the workspace as shown in the screenshot. You can find additional information on your license in the **Help ➥ About RiverWare...** dialog. Then click on **Show System Info...** and **More About CPLEX**.

2. Preemptive Linear Goal Programming: The Mathematics

This section provides technical details about the mathematics in the Preemptive Linear Goal Programming solution used by RiverWare Optimization. This includes details about how constraints and objectives are applied and how they get formulated internally in the RiverWare Optimization solution. It also describes the use of priorities to handle conflicting objectives.

2.1 Preemptive Linear Goal Programming Overview

This section provides a conceptual overview to the Preemptive Linear Goal Programming solution used by RiverWare Optimization. It begins with brief, general descriptions of linear programming, goal programming and preemptive goal programming. These are followed by some simple examples to illustrate the use of preemptive goal programming to handle conflicting objectives and conflicting constraints.

2.1.1 Linear Programming

The RiverWare Optimization solver uses linear programming at its core. Linear programming is a common optimization approach where linear objective function is maximized or minimized subject to a set of linear constraints.

Maximize $z = c_1x_1 + c_2x_2 + \dots + c_{n-1}x_{n-1} + c_nx_n$

Subject to:

$$a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n-1}x_{n-1} + a_{1,n}x_n = b_1$$

$$a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n-1}x_{n-1} + a_{2,n}x_n \le b_2$$

$$a_{3,1}x_1 + a_{3,2}x_2 + \ldots + a_{3,n-1}x_{n-1} + a_{3,n}x_n \ge b_3$$

$$\ldots$$

$$a_{m,1}x_1 + a_{m,2}x_2 + \ldots + a_{m,n-1}x_{n-1} + a_{m,n}x_n = a_m$$

where each *x* is a variable in the problem. Each *c* is a numeric coefficient in the objective function. Each *a* is a numeric coefficient in a constraint. Each *b* is the numeric right-hand-side value of each constraint. Each constraint can be either an “equal-to,” “less-than-or-equal-to,” or “greater-than-or-equal-to” constraint. The objective function and the left-hand-side of every constraint must be a linear combination of one or more variables. Further general information about linear programming, including approaches for solving a linear program, are covered in any text on the basics of linear programming optimization and thus are not covered here.

In RiverWare, an optimization model ultimately gets formulated as a linear program. The details about that formulation are covered in following sections.

2.1.2 Goal Programming

One limitation of standard linear programming is that all constraints are “hard” constraints. If there is not a solution that can simultaneously satisfy all constraints, then the problem is infeasible. Also, there is only a single objective function.

Often in water resources contexts, a system will have multiple objectives that need to be traded off against each other. For example, there might be an objective to maximize the value of hydropower generation, but that must also be traded off with environmental objectives and an objective to carry sufficient reserve capacity. If environmental objectives are expressed as constraints (e.g. minimum and maximum flows), cases can result in which the problem becomes infeasible. For example, low inflow inputs can produce a case in which it is not possible to simultaneously satisfy minimum reservoir level constraints and minimum outflow requirements.

One approach to handle the case of conflicting objectives or conflicting constraints is goal programming. In traditional goal programming, a penalty function is applied to any deviation from a constraint value, and the penalties are minimized as part of the objective function. Typically penalty functions are convex. Most often the penalty function is linear, piecewise linear or quadratic because of the available solution mechanisms. For example, some objectives have a term that minimizes the sum of squared violations. In order to combine terms with non-commensurate units in a single objective, the deviations are typically scaled. Additionally the penalty functions and multiple objectives in the objective function might be weighted to express the relative importance of some policies over others. One challenge of this approach is that it can be difficult to interpret the meaning of the weights applied. Also the solution can be sensitive the weights that are used, and it can be difficult to anticipate the effects of modifying weights without significant trial and error.

Rather than combining weighted penalty terms and weighted objectives in a single objective function, RiverWare Optimization uses an alternative form of goal programming referred to as preemptive goal programming, which is described in the following section.

2.1.3 Preemptive Goal Programming

RiverWare Optimization uses a form of goal programming referred to as preemptive goal programming. Constraints and objectives are expressed at distinct priorities. At each priority an optimization problem is solved either to maximize or minimize the stated objective at that priority. If the priority contains constraints, the objective is to maximize the satisfaction of the constraints. (In RiverWare, this “derived” objective to maximize constraint satisfaction is automatically formulated based on the constraints that the user specifies.) Before moving on to the next priority, the objective value is “frozen.” This is the equivalent of adding a constraint to the problem that sets the objective function expression equal to the objective value. The concept of “freezing” is explained later with an example and is covered in more detail **HERE (Section 2.3)**. This freezing guarantees that lower priority constraints and objectives will not degrade higher priority constraints and objectives. After the higher priority solutions, there are typically many alternative optima, multiple feasible solutions that would equally maximize the objective function. Any solution for a lower priority must come from the set of alternative optima from the preceding solution. As the solution proceeds through the priorities, the solution space is reduced by the constraints added at each priority. Another way to say this is that the set of alternative optima is reduced, or degrees of freedom are removed from the problem.

The following, simple example illustrates the basic concepts of preemptive goal programming, including prioritized objectives, the shrinking solution space, alternative optima and freezing of variables and constraints.

Assume we have two variables, *a* and *b*. The problem has only two dimensions. The variables are constrained by the following (hard constraints):

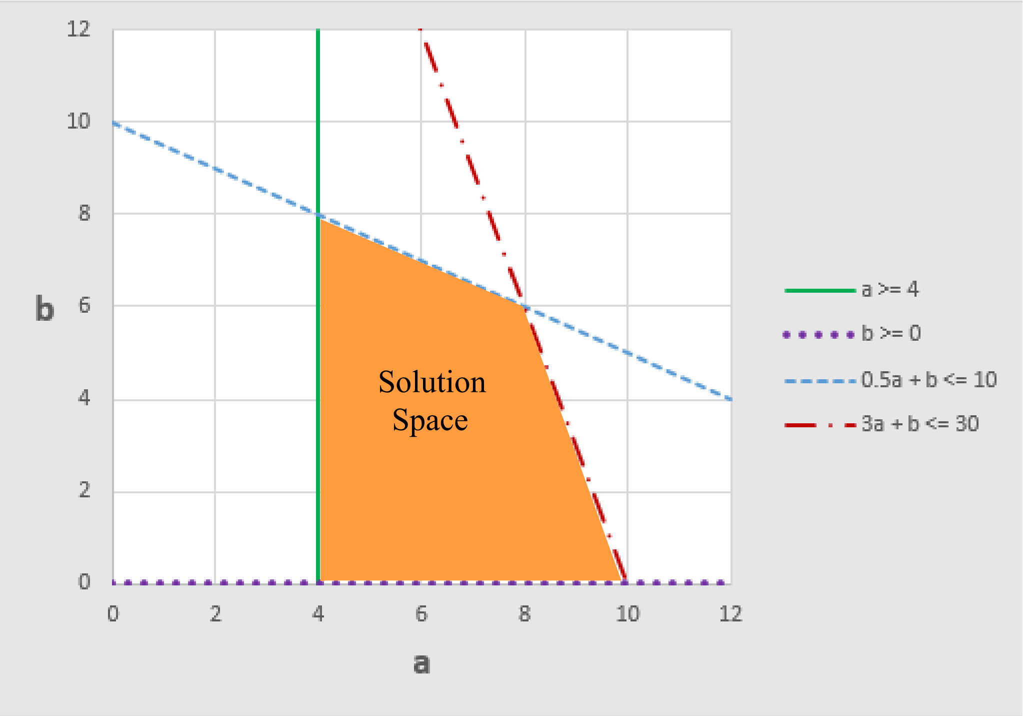
$a \geq 4$

$b \geq 0$

$0.5a + b \leq 10$

$3a + b \leq 30$

These constraints limit the solution space as illustrated in the following plot. In other words they have removed degrees of freedom from the solution.



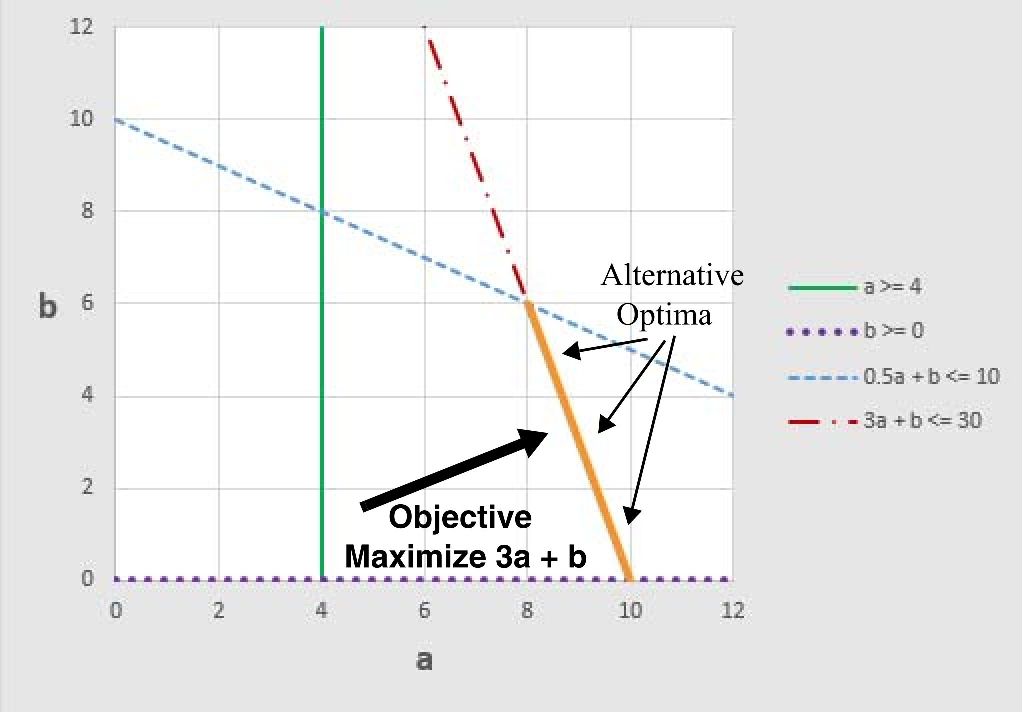
Any feasible solution must be within the orange region (including the edges that bound the region).

The problem has two prioritized objectives.

Objective 1 (highest priority): Maximize $z_1 = 3a + b$

Objective 2: (lower priority): Maximize $z_2 = b$

First, we will solve for the highest priority objective, Maximize $z_1 = 3a + b$.



The figure above shows the direction that the objective is pushing the solution. In this case, there are many possible solutions that will equally maximize the objective. Another way to state this is that there are alternative optima. Any point that is on the line $3a + b = 30$ with a between 8 and 10 (equivalently, b between 0 and 6) satisfies all constraints and equally maximizes the objective. The alternative optima are illustrated by the orange line segment in the figure above. Given the current set of constraints and objectives we have included in the problem (we have not yet added Objective 2), there is no preference for one point over another from this set of alternative optima. If we increase a, then we must decrease b, but in all cases the objective value, z_1, is 30. The simplex method for solving linear programs tends to generate "extreme point" solutions. In this case it would generate a solution at one of the end points of the line segment.

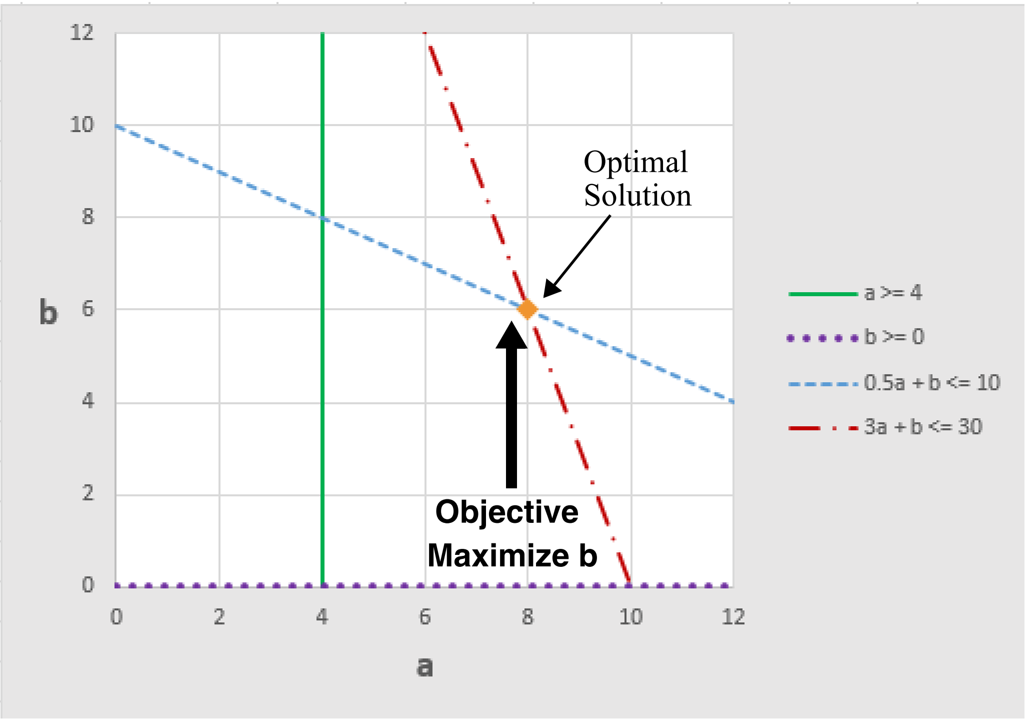
Now as we move on to Objective 2, we will not degrade the objective value of the higher priority Objective 1. In order to guarantee that Objective 2 does not degrade Objective 1, we will only use the solutions from the set of alternative optima from Objective 1. This is done by "freezing" the objective value with an "equal-to" constraint:

$$3a + b = 30$$

Note that we have frozen the constraint $3a + b \leq 30$ by changing it to an "equal to" constraint.

The new constraint reduces the solution space to the orange line segment in the figure above (i.e. remove degrees of freedom).

Now we apply Objective 2: Maximize $z_2 = b$. This is illustrated in the next figure. Once again the large arrow shows the direction that the objective is driving the solution.



In this case, the objective evaluates to a single optimal solution with the following values for all variables:

$a = 8$

$b = 6$

$z_1 = 30$

$z_2 = 6$

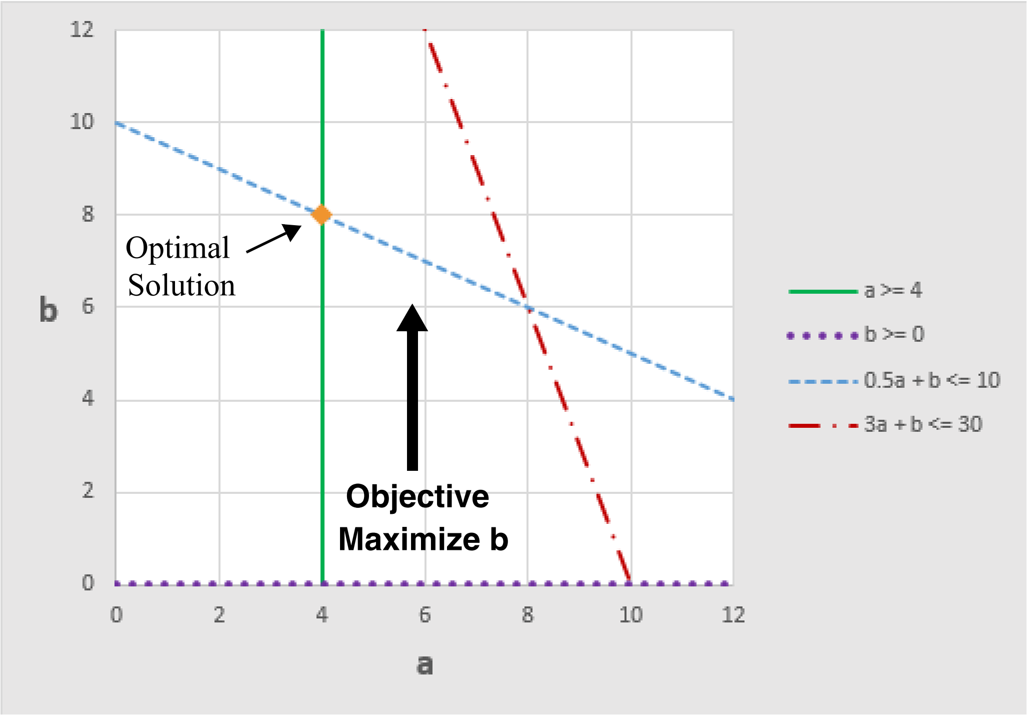
All degrees of freedom have been removed from the solution. There are no longer any alternative optima, only a single optimal solution.

Now let’s assume we have the same problem, but we reverse the priorities of the two objectives.

Objective 1 (highest priority): Maximize $z_1 = b$

Objective 2: (lower priority): Maximize $z_2 = 3a + b$

In this case, Objective 1 would remove all degrees of freedom, evaluating to a single optimal solution as illustrated in the figure below.



The objective value gets frozen as.

$z_1 = 8$

The constraint $0.5a + b \leq 10$ also gets frozen, changing it to an “equal to” constraint.

$0.5a + b = 10$

In addition the constraint $a \geq 4$ gets frozen.

$a = 4$

There are no alternative optima, so there are no degrees of freedom left for Objective 2 to affect the solution. The final solution is determined after Objective 1.

$a = 4$

$b = 8$

$z_1 = 8$

$z_2 = 20$

2.2 Soft Constraints and Derived Objectives

In RiverWare, operational policy constraints created by the user are typically added to the optimization problem as prioritized soft constraints. (Constraints defining the physical characteristics of the system are automatically added to the optimization problem by RiverWare as hard constraints. See **HERE (Section 4.2.3)**.) RiverWare automatically converts the soft constraints at each priority into a "derived objective" to maximize the satisfaction of the soft constraints (minimize violations). This is done to prevent an infeasible problem in the case that it is not possible to satisfy all constraints. If it is not possible to fully satisfy all constraints at a given priority, the solution will get as close as possible, rather than return an infeasible solution. The next section provides a simple example with two variables to illustrate the conceptual use of prioritized soft constraints. This is followed by a simple example from a water resources management context. Sections 2.2.2 and 2.2.3 proved technical details about how soft constraints are defined and how derived objectives are formulated respectively.

2.2.1 Prevent Infeasibilities

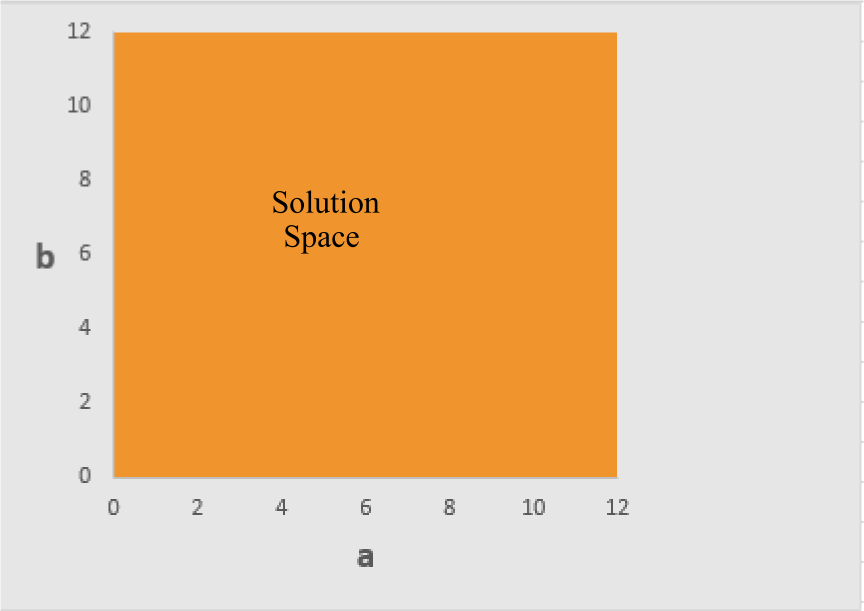
2.2.1.1 A Two-dimensional Example

To illustrate the conceptual use of prioritized soft constraints to prevent infeasibilities, assume an optimization problem with two variables, *a* and *b*. The variables have the following formal bounds.

$0 \le a \le 12$

$0 \le b \le 12$

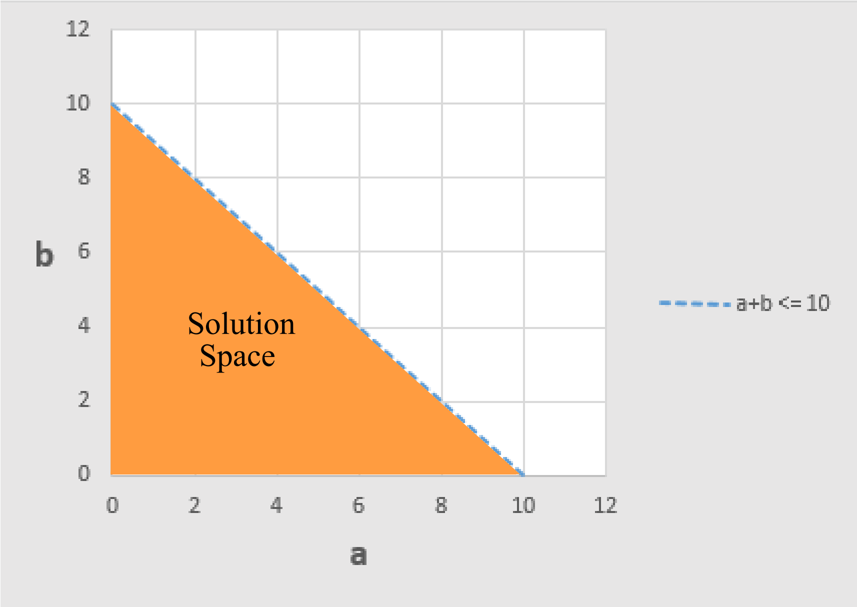
The solution space given those bounds is illustrated in the following figure.



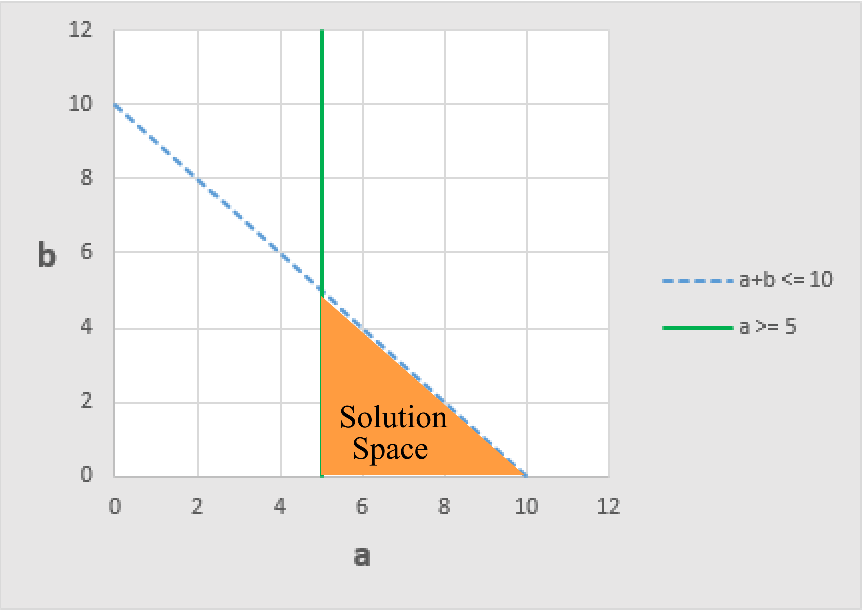
The problem has the following soft constraints and objectives in order from highest priority to lowest priority.

1. Soft Constraint: $a + b \le 10$
2. Soft Constraint: $a \ge 5$
3. Soft Constraint: $b \ge 6$
4. Objective: Maximize $z_1 = a + b$
5. Objective: Maximize $z_2 = b$

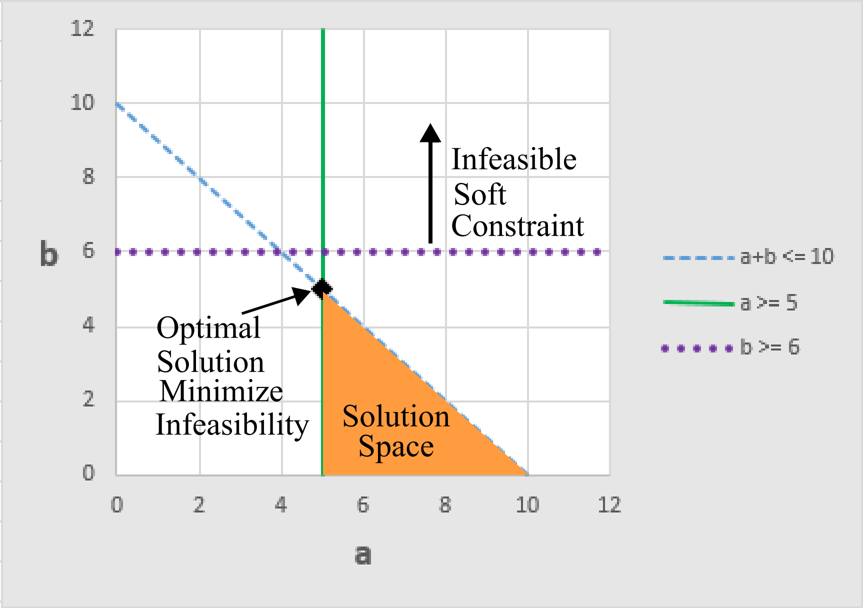
At priority 1, adding the constraint $a + b \le 10$ reduces the solution space as illustrated in the figure below.



At priority 2, the constraint $a \ge 5$ further reduces the solution space as illustrated in the next figure.



Now when we get to priority 3, we can see that the constraint $b \geq 6$ cannot be satisfied without violating one of the two higher priority constraints. If the constraints were all hard constraints, the problem would be infeasible, and there would be no solution. Using soft constraints, the solution gets as close as possible to satisfying the constraint. It does this in a way that minimizes the violation of the priority 3 constraint in a manner that does not compromise the satisfaction of the two higher priority constraints. This is illustrated in the next figure.



In this case, priority 3 removes all degrees of freedom from the solution. All of the constraints are frozen, and there is only one optimal solution.

$$a + b = 10$$

$$a = 5$$

$$b = 5$$

With no degrees of freedom remaining, the two objectives can do nothing to affect the solution.

2.2.1.2 Goal Programming with Many Dimensions

In the two dimensional examples above, we saw that a soft constraint may have one of several possible effects on the solution space:

- Cut away a portion of the 2-d solution space, leaving a smaller 2-d area as the new solution space
- Limit the solution to a 1-d line segment
- Pick a 0-d solution, a single point, which is a vertex

A soft constraint could also have no effect because it does not reduce space of feasible solutions.

If we extend the goal programming solution to a problem with three variables (three dimensions), the initial solution space is represented by a 3-d polyhedron. A soft constraint may have one of several possible outcomes:

- Cut away a portion of the 3-d polyhedron, leaving a smaller 3-d polyhedron as the new solution space
- Limit the solution to a face of the polyhedron, which is a 2-d polyhedron
- Limit the solution to a 1-d line segment on the polyhedron
- Pick a 0-d solution, a single point, which is a vertex of the polyhedron
- No effect because it does not reduce the polyhedron of feasible solutions

Goal programming in higher dimensions is similar. In *n* dimensions, if a constraint has an effect, it may result in a smaller polyhedron with *n* or fewer dimensions.

Goal programs used in a water resource management context typically have many dimensions. Each physical component (e.g. Reservoir Storage, Reservoir Outflow, Reach Outflow) at each timestep is an individual variable in the problem. Additional decision variables that do not represent physical components of the system may also be included at each timestep (e.g. Energy Value). This results in a solution space represented by a polyhedron in many dimensions. (Note that RiverWare only adds the variables to the problem that are necessary based on the specified policy. See **HERE (Section 4.2.2)** and **HERE (Section 4.2.3)** for details. In addition, any quantities that are known inputs are included in the problem as numeric values, not as variables. For example, the Inflow to the upstream reservoir in a basin might be provided as an input. In this case, the Inflow would not be a variable. The numeric inputs would get incorporated into problem as appropriate, in mass balance constraints for example.)

2.2.1.3 A Water Resource Management Example

To illustrate how goal programming applies in a water management context, consider the following, simple example.

A reservoir is modeled for a single timestep with a length of one day. The initial storage of the reservoir is 50,000 acre-ft, and the inflow over the timestep is an input at 2,000 acre-ft/day. (Assume a single inflow, a single outflow and no other gains or losses.) The reservoir has the following operational policy in priority order, highest priority to lowest.

1. Minimum Storage Constraint: $\text{Storage}_t \geq 45{,}000 \text{ acre-ft}$
2. Minimum Outflow Constraint: $\text{Outflow}_t \geq 10{,}000 \text{ acre-ft/day}$
3. Maximize Storage Objective: Maximize Storage_t

The priority 1 constraint can be fully satisfied and guarantees that the Storage will not go below 45,000 acre-ft when solving for the later priorities.

The priority 2 constraint cannot be fully satisfied without drafting the reservoir to 42,000 acre-ft, but priority 1 already guaranteed that the reservoir would not go below 45,000 acre-ft. So priority 2 will get as close as possible to meeting the Minimum Outflow by setting the Outflow to 7,000 acre-ft/day. This will freeze both constraints as follows:

$$\text{Storage}_t = 45{,}000 \text{ acre-ft}$$

$$\text{Outflow}_t = 7{,}000 \text{ acre-ft/day}$$

In this case there are no degrees of freedom remaining for the priority 3 objective, so it will do nothing to the solution. If the constraints at priorities 1 and 2 were added as hard constraints, the problem would be infeasible, and there would be no solution. Using soft constraints provides a solution that gets as close to the constraints as possible based on their priorities.

Now assume that the inflow over the timestep is 7,000 acre-ft/day. Both constraints could be fully satisfied, and the objective at priority 3 would result in the following solution.

$$\text{Storage}_t = 47{,}000 \text{ acre-ft}$$

$$\text{Outflow}_t = 10{,}000 \text{ acre-ft/day}$$

In practice, optimization problems for a water management context are far more complex, spanning many timesteps, multiple objects (physical features of the basin) and many operational policy constraints and objectives. The initial solution space is a polyhedron of many dimensions. In those contexts, the general conceptual approach is the same as for this example: at each priority get as close to satisfying the constraints as is possible (maximize satisfaction) without degrading any of the higher priority constraints or objectives. However there are multiple possible approaches to “maximize satisfaction.” The use of soft constraints and derived objectives to maximize soft constraint satisfaction in RiverWare’s optimization solver are described in more technical detail in the following sections.

2.2.2 Satisfaction Variables in Soft Constraints

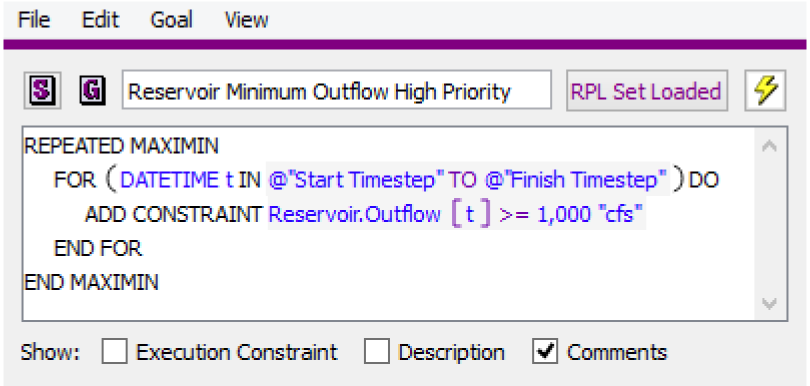
This section describes how satisfaction variables are introduced to formulate soft constraints. The following section describes how those satisfaction variables are included in a derived objective to maximize the soft constraint satisfaction (minimize violations).

In RiverWare, soft constraint violations, v_i, are scaled from 0 to 1 where 1 corresponds to the maximum violation from a higher priority constraint or bound. For performance reasons, the RiverWare Optimization solution actually uses a formulation based on satisfaction rather than violations.

$$s_i = 1 - v_i$$

It then maximizes satisfaction, where 1 corresponds to fully satisfied, and 0 corresponds to doing no better than the previous bound.

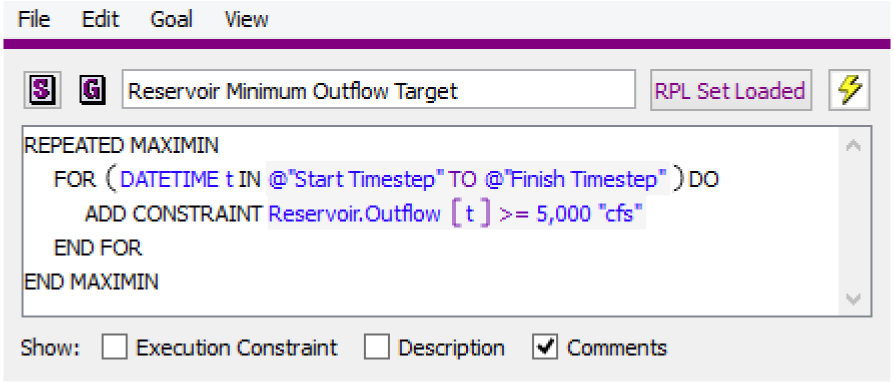
As an example, assume a reservoir has the following high priority minimum flow constraint.



Reservoir Minimum Outflow High Priority

```
REPEATED MAXIMIN
   FOR ( DATETIME t IN @"Start Timestep" TO @"Finish Timestep" ) DO
      ADD CONSTRAINT Reservoir.Outflow [ t ] >= 1,000 "cfs"
   END FOR
END MAXIMIN
```

(Technically this would translate to multiple constraints, one for each timestep.) Then assume that at a lower priority a more restrictive target minimum outflow is added.



Reservoir Minimum Outflow Target

```
REPEATED MAXIMIN
   FOR ( DATETIME t IN @"Start Timestep" TO @"Finish Timestep" ) DO
      ADD CONSTRAINT Reservoir.Outflow [ t ] >= 5,000 "cfs"
   END FOR
END MAXIMIN
```

Now assume that the high priority minimum flow constraint of 1,000 cfs was fully satisfied on all time steps, but on one time step it was only possible to get an outflow 4,000 cfs due to some other high priority constraints and low inflows. The lower priority target minimum outflow would not be fully satisfied. The satisfaction, s_i, for that particular constraint would be 0.75. 4,000 cfs is 75% of the way from the old lower bound of 1,000 cfs to the new constraint value of 5,000 cfs.

For a greater-than-or-equal-to constraint containing only a single variable, this can be expressed as:

$$s_i \le \frac{\mathrm{VariableValue} - \mathrm{OldLowerBound}}{\mathrm{NewConstraintValue} - \mathrm{OldLowerBound}}$$

$$0 \le s_i \le 1$$

In the above example,

$$s_i = \frac{4000\mathrm{cfs} - 1000\mathrm{cfs}}{5000\mathrm{cfs} - 1000\mathrm{cfs}} = 0.75$$

Now assume that the high priority constraint setting a minimum of 1,000 cfs did not exist, and thus the constraint using 5,000 cfs was the first constraint with this left-had-side formulation. In this case, the *OldLowerBound* would be taken from the Lower Bound set on the variable (slot configuration, see **HERE (Section 4.2.4)**). Assume that the slot Lower Bound was 0 cfs. Then the satisfaction would be

$$s_s = \frac{4000\mathrm{cfs} - 0\mathrm{cfs}}{5000\mathrm{cfs} - 0\mathrm{cfs}} = 0.8$$

More generally, a soft constraint is defined as

$$a_1x_1 + a_2x_2 + \ldots + a_{n-1}x_{n-1} + a_nx_n \le s_i(b - \mathrm{OldUpperBound}) + \mathrm{OldUpperBound}$$

or

$$a_1x_1 + a_2x_2 + \ldots + a_{n-1}x_{n-1} + a_nx_n \ge s_i(b - \mathrm{OldLowerBound}) + \mathrm{OldLowerBound}$$

$$0 \le s_i \le 1$$

where each x_n is an optimization variable in the constraint. Each a_n is a numeric coefficient on the corresponding variable, *b* is the right-hand-side numeric constraint value, and *OldBound* is the previous limit for the right-hand-side either from a higher priority constraint of the same form (same left-hand-side) or based on the slot bounds if there is not a higher priority constraint with the same LHS. Note that internally RiverWare always rearranges all constraints to have this same “cannonical” form with the sum of variables and their coefficients on the LHS and all constant numeric values combined in a single RHS value.

For every priority that contains soft constraints, RiverWare will add the constraints in this form, using satisfaction variables, to Optimization problem. It will then solve the Optimization problem at that priority with a derived objective to maximize the satisfaction. The derived objective that RiverWare automatically creates to maximize the value of the satisfaction variable(s) is described in the following section.

2.2.3 Derived Objectives

For all priorities that contain soft constraints, RiverWare automatically generates an objective to maximize the satisfaction of the soft constraints and then solves the Optimization problem at that priority to maximize the derived objective with the new soft constraints added. The manner in which satisfaction is maximized depends on which type of derived objective is selected. Each of the derived objectives are described in the following sections. Details about how to select which derived objective to use are given in the section on the RPL Optimization Goal Set, **HERE (Section 4.1.3.2)**.

There are three general types of derived objectives in RiverWare: Summation, Single Maximin and Repeated Maximin. In addition Summation with Reward Table (Reward Function) is described here as a fourth type of derived objective, though technically it is implemented by adding a “With Reward Table” statement within a Summation derived objective (see **HERE (Section 4.1.3.5)**).

2.2.3.1 Summation

The Summation derived objective includes a separate satisfaction variable in each constraint added to the Optimization problem at the current priority *p*. The objective is then to maximize the sum of all of the satisfaction variables added at that priority.

$$\text{Maximize} \sum_{\text{all } i} s_{p,\,i}$$

where there is one satisfaction variable, $s_{p,i}$, for each constraint *i* added at the current priority *p*.

After maximizing the derived objective, the objective value is frozen so that the objective (satisfaction) is not degraded by lower priority objectives. Conceptually the following constraint is added to the Optimization problem

$$\sum_{\text{all } i} s_{p,\,i} = z_p$$

were z_p is the value from maximizing the derived objective. Technically this constraint does not get included, but the equivalent of adding this constraint is carried out by freezing select constraints and variables. See **HERE (Section 2.3)** for details about freezing constraints and variables.

The potential downside of the Summation approach is that it can tend to place all of the deviation on one or just a few constraints. For example, if a minimum flow constraint cannot be met for every timestep, it might minimize the total violation by satisfying the constraint at all but one timestep and putting a very large violation on that one timestep. Typically, this is not the preferred solution in a water resources context.

2.2.3.2 Single Maximin

The Single Maximin derived objective applies the same satisfaction variable, s_p, for all constraints added at priority *p*. The objective is then to maximize the value of that single satisfaction variable.

$$\text{Maximize } s_p$$

Instead of minimizing the total violation, this approach will minimize the single largest violation (formally it maximizes the minimum satisfaction). This prevents a single, very large violation by spreading the violation out over multiple variables or timesteps.

Conceptually, the following constraint is added to the Optimization problem to freeze the satisfaction.

$$s_p = z_p$$

where z_p is the value from maximizing the derived objective. Technically this constraint does not get included, but the equivalent of adding this constraint is carried out by freezing select constraints and variables. The solution will freeze all constraints that are limiting the satisfaction if s_p is less than 1. These are the constraints with the largest violation. See **HERE (Section 2.3)** for details about freezing constraints and variables.

The potential downside of the Single Maximin approach is that it does nothing to improve the satisfaction of the remaining constraints that are not frozen. There might be other constraints introduced at priority *p* that could have a higher satisfaction that the limiting constraint(s), but this approach does nothing to enforce that higher satisfaction when moving on to solve at lower priorities. It only guarantees that they will have the same satisfaction as the least satisfied constraint. This issue is addressed by the next type of derived objective, Repeated Maximin.

2.2.3.3 Repeated Maximin

The Repeated Maximin approach begins with the single maximin. A single satisfaction variable, $s_{p,1}$, is applied for the first iteration for all constraints added at the current priority *p*, and it solves the Optimization problem with an objective to maximize the satisfaction variable (minimize the largest violation(s)). It then conceptually freezes the satisfaction variable (satisfaction of the constraint(s) with the largest violation). It also freezes all constraints that are limiting the satisfaction if $s_{p,1}$ is less than 1. These are the constraints with the largest violation. (Click **HERE (Section 2.3)** for details about freezing constraints.) It then adds a new satisfaction variable, $s_{p,2}$, for all remaining constraints and solves to minimize the next largest violation (technically maximize the next smallest satisfaction). It repeats this process until either all violations have been minimized (all constraints added at the current priority *p* have been frozen) or all remaining constraints can be fully satisfied ($s_{p,i}$) equals 1.

```
While $s_{p,i-1} < 1$
    Maximize $s_{p,i}$
    $s_{p,i} \geq s_{p,i-1}$    for $i > 1$
```

where $s_{p,i}$ is a single satisfaction variable applied to all constraints added at the current priority p that have not yet been frozen prior to the *ith* Repeated Maximin iteration.

If it is possible to fully satisfy all constraints added at the current priority p (i.e. $s_{p,1} = 1$), there will only be a single iteration. In this case there may not be any constraints frozen, dependent on whether the goal forced any constraints to their limits.

When it is not possible to fully satisfy all constraints at a given priority, the Repeated Maximin approach tends to spread out deviations as evenly as possible, which is often the desired solution in a water resources context. We recommend Repeated Maximin as the default approach for most soft constraints. It is relatively easy to change to one of the other types of derived objectives later if this is desirable.

The potential downside of the Repeated Maximin approach is that in some cases it can require numerous iterations when it is not possible to satisfy all constraints. Depending on the size of the model this can cause a significant increase run time. The Summation approach always requires only a single solution at each priority, so it tends to be faster, but it does not, in general, provide the same solution quality as Repeated Maximin. In some cases, a reasonable balance between solution quality and run time can be provided by applying the Summation with Reward Table approach as described in the next section.

2.2.3.4 Summation with Reward Table

Summation with Reward Table is a special case of the Summation derived objective. Rather than summing all satisfaction variables at a given priority equally, this approach applies a reward function to each of the satisfaction variables. Typically this approach is used to penalize larger violations more than smaller violations (greater reward for higher satisfaction). For example, a common approach is to minimize the sum of squared violations. This tends to "smooth out" large violations by spreading them out over multiple time steps rather than concentrating large violations on a few time steps. Applying the appropriate reward function can often produce a result that compares to the Repeated Maximin solution in terms of solution quality but has the benefit of only requiring a single solution at the given priority and can therefore be much faster.

Note that technically the Summation with Reward Table approach is implemented by adding a With Reward Table statement within a Summation derived objective. For details on implementing Summation with Reward Table, click **HERE (Section 4.1.3.5)**.

A "reward table" is used to apply a piecewise function to the satisfaction variables within a Summation objective. Instead of using the standard Summation derived objective for maximizing satisfaction, the derived objective becomes

$$\text{Maximize} \sum_{\text{all i}} R_{p,\,i}$$

where *R* is the reward function for the satisfaction variables.

$$R_{p,\,i} = f(S_{p,\,i})$$

The reward table slot has two columns. The first column contains satisfaction values. The second column contains the corresponding reward values. The values in both columns must be between 0 and 1. The reward table must be concave. (Reward must be a concave function of Satisfaction.)

Details of the general formulation of the Summation with Reward Table derived objective are given below, followed by an example that minimizes the sum of squared violations.

The following constraint is added that defines the satisfaction $S_{p,i}$ of constraint *i* introduced at priority *p* as the sum of all piecewise segments.

$$S_{p,\,i} = \sum_{\text{all j}} x_{p,\,i,\,j}$$

where $x_{p,i,j}$ is the length (Δx) of each segment *j*. The number of segments is defined by the Reward Table (# segments = # rows - 1). The maximum length of each segment is constrained by the satisfaction values $s_{p,i,j}$ entered in the first column of the Reward Table.

$$0 \le x_{p,\,i,\,j} \le s_{p,\,i,\,j} - s_{p,\,i,\,j-1}$$

The first row in the table is indexed as row 0. The slope for each segment $m_{p,i,j}$ is calculated as

$$m_{p,\,i,\,j} = \frac{r_{p,\,i,\,j} - r_{p,\,i,\,j-1}}{s_{p,\,i,\,j} - s_{p,\,i,\,j-1}}$$

where each $r_{p,i,j}$ is the value in the Reward column of the Reward Table corresponding to the satisfaction $s_{p,i,j}$.

The objective maximizes the total reward R_p, which is defined as follows.

$$\text{Maximize} \qquad R_p = \sum_{\text{all } i} \sum_{\text{all } j} m_{p,i,j} x_{p,i,j}$$

An example is provided below for a reward table that corresponds to minimizing the sum of squared violations

Example Reward Table: Minimize the Sum of Squared Violations

Conceptually the desired derived objective is to minimize the sum of squared violations of all constraints introduced at the current priority *p*.

$$\text{Minimize} \sum_{\text{all } i} v_{p,i}^{2}$$

where $v_{p,i}$ is the violation of constraint *i* introduced at the current priority *p*. Violations are scaled from 0 to 1, where 1 corresponds to the maximum violation based on a higher priority constraint or variable bound. Thus an equivalent objective to minimize the sum of squared violations is

$$\text{Maximize} \sum_{\text{all } i} (1 - v_{p,i}^{2})$$

For performance reasons, the RiverWare Optimization solution maximizes satisfaction variables rather than minimizing violations. (Click **HERE (Section 2.2.2)** for details about how the satisfaction variables are defined.).

$$S_{p,i} = 1 - v_{p,i}$$

The maximize objective then becomes

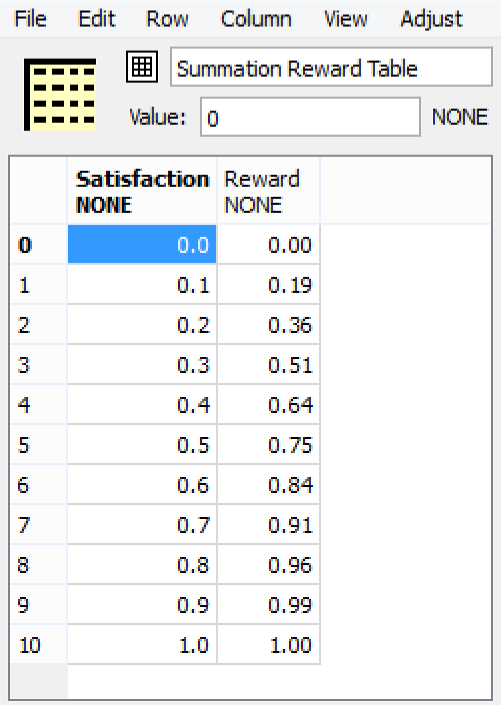
$$\text{Maximize} \sum_{\text{all i}} (1 - (1 - S_{p,\,i})^2)$$

This can be approximated by a piecewise linear function as described in the following table, which linearizes the function in 10 discreet segments.

Reward Table to Minimize Squared Violations

v_i	v_i^2	s_i	$1 - v_i^2$
1.0	1.00	0.0	0.00
0.9	0.81	0.1	0.19
0.8	0.64	0.2	0.36
0.7	0.49	0.3	0.51
0.6	0.36	0.4	0.64
0.5	0.25	0.5	0.75
0.4	0.16	0.6	0.84
0.3	0.09	0.7	0.91
0.2	0.04	0.8	0.96
0.1	0.01	0.9	0.99
0.0	0.00	1.0	1.00

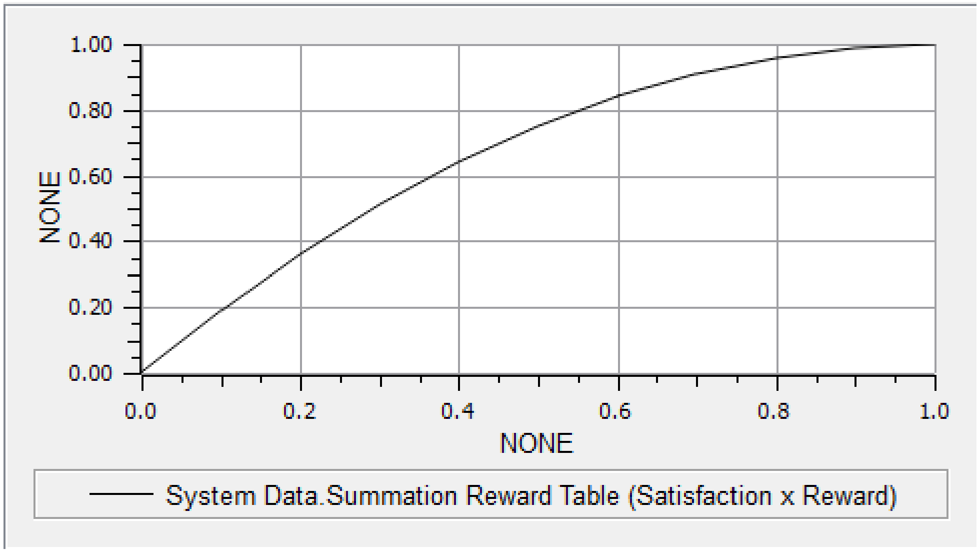
The reward table entered in RiverWare is taken from the last two columns as shown below and in the plot that follows.



Summation Reward Table

Value: 0 NONE

	Satisfaction NONE	Reward NONE
0	0.0	0.00
1	0.1	0.19
2	0.2	0.36
3	0.3	0.51
4	0.4	0.64
5	0.5	0.75
6	0.6	0.84
7	0.7	0.91
8	0.8	0.96
9	0.9	0.99
10	1.0	1.00



More generally, any concave table may be used. For example, a table can be created for minimizing cubed violations. Tables with more rows may degrade performance (increase run time) and numerical stability. There is a tendency for optimal solutions to be at the discrete values in the table. This may be more noticeable when tables with fewer rows (larger segments) are used. For example, if the reward table only had three rows corresponding to satisfaction values of 0, 0.5 and 1, it might be common to see the solution at a satisfaction of either 0.5 or 1.0. Some experimentation may be required to determine the best table size for a given goal.

2.3 Freezing Constraints and Variables

After solving at each priority, the RiverWare Optimization solution guarantees that later priorities will not degrade the objective from the current priority by locking in or “freezing” portions of the solution. Conceptual examples of this are illustrated **HERE (Section 2.1.3)**. For numeric stability reasons, RiverWare does not technically freeze the objective function itself but rather freezes constraints and variables. These are constraints that are forced to their limits by the objective and variables that are forced to their upper or lower bounds. Freezing the appropriate constraints and variables guarantees that the objective will not be degraded by lower priorities. Violations of soft constraints always results in freezing that removes degrees of freedom from the solution. If there are no violations, freezing usually reduces the size of the feasible polyhedron without removing degrees of freedom.This section provides more details about the freezing of constraints and variables.

Information about frozen constraints and objectives is helpful in Optimization solution analysis. Constraints that get frozen at the priority that they are introduced, which are typically soft constraints that cannot be fully satisfied, are constraints that are driving the solution at that priority. Constraints that were introduced at an earlier priority and get frozen at a lower priority are limiting the objective function at the lower priority. In the case of unsatisfied soft constraints at a given priority, the frozen constraints that were introduced at an earlier priority are preventing the constraints from being fully satisfied. Details about frozen constraint information in the Optimization Solution Analysis Tool are given **HERE (Section 7.1)**.

Freezing always occurs automatically after each iteration of a Repeated Maximin derived objective. This is necessary given the nature of the solution. A second Repeated Maximin iteration at a given priority only makes sense when constraints and variables are frozen after the first iteration. Otherwise it would simply be re-solving the same problem over and over. For all other objective types (Maximize, Minimize, Summation, Single Maximin), an explicit Freeze statement must be added to the goal after the objective statement. See **HERE (Section 4.1.3.4)** for details about how to apply a Freeze statement to goals. A common mistake is forgetting to add the Freeze statement after these objectives. If the Freeze statement is omitted, the solution will look the same as if the goal was never included in the optimization policy. The reason for this design is to retain the ability to have “test” objectives that provide information for a subsequent goal without using it to freeze the solution. This approach is not common, but when needed, it is very useful.

2.3.1 Freezing Constraints

Assume that at priority 1 a reservoir has a Minimum Storage constraint for every timestep.

$$\text{Reservoir.Storage}[t] \geq 10,000 \text{ acre-ft}$$

At priority 2, the reservoir has a Minimum Outflow constraint.

$$\text{Reservoir.Outflow}[t] \geq 5{,}000 \text{ cfs}$$

Assume that, given the initial Storage and Inflow over the first three timesteps (t1, t2, t3), it is not possible to meet both the Minimum Outflow and the Minimum Storage. The Minimum Storage is at a higher priority, so the solution would satisfy those constraints, and it would get the Outflow as close to the Minimum of 5,000 cfs as possible. In doing so, it would take the Storage down to the minimum by the third timestep. Using as much water as is available from the reservoir is the means by which it would get as close to the Minimum Outflow as possible. In other words, the Minimum Outflow constraint at priority 2 forces the Minimum Storage from priority 1 to be “tight” on t3, and the Minimum Storage constraint on t3 is limiting the objective for priority 2.

For all priorities below priority 2, we already know that the Storage at t3 must be at the Minimum Storage in order to prevent degrading the priority 2 objective to maximize the satisfaction of the Minimum Outflow constraint. So RiverWare “freezes” the Minimum Storage constraint on t3 by changing it to an “equal-to” constraint.

$$\text{Reservoir.Storage[t3]} = 10{,}000 \text{ acre-ft}$$

This would remove a degree of freedom from the solution for all subsequent priorities. The Storage on t1 and t2 is still constrained to be greater-than-or-equal-to 10,000 acre-ft, but at this point they have not been forced to be equal to 10,000 acre-ft, so they are not frozen here.

Technically the constraint would be frozen in its canonical form including the satisfaction variable; see **HERE (Section 2.2.2)**. Assume that the initial lower bound on Reservoir Storage before priority 1 was 0 acre-ft, then this would look like the following:

$$1.0 \times \text{Reservoir.Storage[t3]} = s_1(10{,}000 \text{ acre-ft} - 0 \text{ acre-ft}) + 0 \text{ acre-ft}$$

where 1.0 is the coefficient on the Reservoir Storage variable, s_1 is the satisfaction variable from priority 1, which would have been frozen at a value of 1.0 when solving at priority 1, and 0 acre-ft was the previous bound on Reservoir Storage.

(Note that technically all evaluation is done internally in RiverWare Optimization units, which are typically scaled SI units, not in user display units as shown here.)

The Minimum Outflow constraints would also get frozen at priority 2 but with a satisfaction variable value less than 1. Assume that the previous bound on Outflow was 0 cfs, and the priority 2 solution was only able to get the outflow to 4,000 cfs on all three timesteps. The satisfaction would be 0.8 (80% from the old bound of 0 cfs to the new constraint value of 5,000 cfs). The Minimum Outflow constraint would be frozen as:

$$1.0 \times \text{Reservoir.Outflow[t1]} = s_2(5{,}000 \text{ cfs} - 0 \text{ cfs}) + 0 \text{ cfs}$$

where s_2 is the satisfaction variable for priority 2 and will be frozen at a value of 0.8. The Minimum Outflow constraints at t2 and t3 would be frozen in a similar manner.

RiverWare determines which constraints to freeze by evaluating the dual price for each constraint. The dual price represents the unit improvement in the objective function that would result from a unit change in the constraint right-hand-side value while holding all else constant. If a constraint’s dual price is greater than 0, it is an indication that the constraint is limiting the objective (i.e. the constraint is binding) so that constraint gets frozen. For example, assume that reducing the Minimum Storage limit on t3 from 10,000 acre-ft to 9,999 acre-ft would allow an Outflow of 4,000.2 cfs on all three timesteps. This would improve the objective value from 80% to 80.004%. The dual price of the Minimum Storage constraint on t3 would be 0.004 %/acre-ft. This is greater than zero, so the Minimum Storage constraint on t3 gets frozen. If the Minimum Storage value on t1 or t2 were relaxed, the objective would still be limited by the Minimum Storage on t3. The the objective value would not be improved by changing the t1 or t2 constraint value. Thus the dual price for these constraints is zero, and the constraints do not get frozen. Internally this evaluation is always carried out in RiverWare Optimization units, and the dual price must actually be greater than a freezing tolerance of 10^{-6} in RiverWare Optimization units in order to freeze the constraint.

When looking at the Optimization Solution Analysis Tool, **HERE (Section 7.1)**, for priority 2, you would see the Minimum Outflow constraints as constraints from the current priority that were frozen. These constraints were driving the solution at priority 2. You would also see the Minimum Storage constraint from t3 as a constraint that was introduced earlier that was frozen at priority 2, indicating that it limited the satisfaction of the priority 2 Minimum Outflow constraints.

2.3.2 Freezing Variables

Variables that are binding on the solution get frozen as well. This occurs when variables are forced to their upper or lower bound by the objective. (For variables that correspond to slots, the upper and lower bounds are set in their slot configuration by the user. See **HERE (Section 4.2.4)** for information about setting the required bounds on variables. For variables that are added to the problem automatically by RiverWare and do not correspond to slots, RiverWare sets the bounds automatically. One example of the latter is the satisfaction variable for a soft constraint.)

For example, Regulated Spill typically has a lower bound of zero. Assume that there are no policy constraints that require Regulated Spill to be greater-than-or-equal-to a non-zero value. An objective to maximize Power generation will often drive Regulated Spill to zero in order to use as much water as possible for generation. The variable will get frozen at its lower bound of zero.

$$\text{Reservoir.Regulated Spill}[t] = 0 \text{ cfs}$$

Another common case in which variables are frozen is the freezing of satisfaction variables for soft constraints. Satisfaction variables automatically have upper and lower bounds of 0 and 1. For example, assume that the goal at priority *i* contains a Repeated Maximin derived objective. RiverWare will add a new satisfaction variable, s_i, to the problem.

$$0 \le s_i \le 1$$

If the solution is able to fully satisfy all of the constraints in goal *i*, then it will freeze the satisfaction constraint at its upper bound of 1.

$$s_i = 1$$

Similar to the use of dual prices for freezing constraints, reduced costs are used in the freezing of variables. RiverWare freezes all variables with a reduced cost greater than zero. The reduced cost represents the unit decrease in the objective value for a unit change in the bound on the variable (unit increase for a lower bound or unit decrease for an upper bound). In the Regulated Spill example, increasing the lower bound on Regulated Spill to be greater than zero would decrease the amount of power that could be generated. Some water would have to be used for spill rather than generation. Thus increasing the lower bound would reduce the objective value, so Regulated Spill would have a non-zero reduced cost. If the upper bound on the satisfaction variable in the second example were less than one, it would reduce the objective to maximize satisfaction, so the reduced cost on the satisfaction variable is non-zero.

In this case that the satisfaction variable is frozen at a value of 1, it is as if the objective is frozen directly. This is not the case in general, however. The satisfaction variable only gets frozen if it is at its bound. (Typically this is when the satisfaction variable is at its upper bound of 1, but it can also occur if a satisfaction variable on an individual constraint in a Summation derived objective is at its lower bound of zero.) If the satisfaction is somewhere between 0 and 1, the satisfaction will not get frozen directly. Freezing all constraints with a non-zero dual price and all variables with a non-zero reduced cost has the equivalent effect of freezing the objective value but performs much better in terms of numeric stability.

Typically information about frozen variables is not as useful as information about frozen constraints when trying to understand the Optimization solution. Often this is due to the fact that when a variable gets frozen at a bound, this is an “obvious” limit that the user considers “intuitively.” Thus it does not tend to provide the user with more information than they already had. In some cases, however, identifying unexpected frozen variables can assist in debugging an Optimization model, particularly if incorrect upper or lower bounds were set on a variable unintentionally.

2.4 Shrinking Constraints

The concept of “shrinking constraints” can be important for understanding some optimization solution analysis information provided by RiverWare and for understanding how the solution behaves in terms of adding new constraints to the optimization problem.

In water resource management policies, it is common to have constraints at different priorities that have the same form (same left-hand-side) and differ only in the right-hand-side constraint value. For example, assume that a reservoir Storage variable has a formal upper bound of 10,000 acre-ft. In the policy constraints, there is a high priority Maximum Storage constraint to not exceed 9,000 acre-ft. At a lower priority there is a Target Operating Range constraint that applies a more restrictive max Storage of 8,000 acre-ft. Typically it is desirable to be below this elevation, but it might not always be possible due to high inflows. Then at a lower priority there is a Target Operating Storage Point constraint that constrains the Storage equal to 7,000 acre-ft, if possible. Note that internally all “equal-to” constraints are converted to an equivalent pair of constraints.

Reservoir.Storage[t] ≥ 7,000 acre-ft

Reservoir.Stroage[t] ≤ 7,000 acre-ft

Together, in addition to the upper bound, there would be three constraints with the same left-hand-side (LHS), in order from highest priority (1) to lowest (3):

Priority 1: $\text{Reservoir.Storage}[t] \le 9{,}000$ acre-ft

Priority 2: $\text{Reservoir.Storage}[t] \le 8{,}000$ acre-ft

Priority 3: $\text{Reservoir.Storage}[t] \le 7{,}000$ acre-ft

(Note that in practice there would typically be constraints on other variables at intervening priorities between the constraints with the same LHS. For simplicity here were are only looking at the constraints with common LHS.)

Rather than add a new constraint to the optimization problem each time, RiverWare minimizes the size of the optimization problem by “shrinking” the lower priority constraints into the higher priority constraint of the same form.

The priority 1 constraint, when written as a soft constraint with a satisfaction variable would have the following form:

$$\text{Reservoir.Storage}[t] \le s_1(9{,}000\text{ acre-ft} - 10{,}000\text{ acre-ft}) + 10{,}000\text{ acre-ft}$$

Assuming that the priority 1 constraint is fully satisfied, the priority 2 constraint would conceptually have the following form:

$$\text{Reservoir.Storage}[t] \le s_2(8{,}000\text{ acre-ft} - 9{,}000\text{ acre-ft}) + 9{,}000\text{ acre-ft}$$

However, instead of adding a new constraint, RiverWare notices that the new constraint has the same LHS and only differs in the right-hand-side value, so it shrinks the priority 2 constraint into the priority 1 constraint in the following form.

$$\text{Reservoir.Storage}[t] \le s_2(8{,}0000\text{ acre-ft} - 9{,}000\text{ acre-ft}) + s_1(9{,}000\text{ acre-ft} - 10{,}000\text{ acre-ft}) + 10{,}000\text{ acre-ft}$$

The less-than-or-equal-to portion of the priority 3 constraint would then shrink into this constraint in the same manner.

$$\begin{aligned}&\text{Reservoir.Storage}[t]\\ &\le s_3(7{,}000\text{ acre-ft} - 8{,}000\text{ acre-ft}) + s_2(8{,}000\text{ acre-ft} - 9{,}000\text{ acre-ft}) + s_1(9{,}000\text{ acre-ft} - 10{,}000\text{ acre-ft}) + 10{,}000\text{ acre-ft}\end{aligned}$$

Note that all evaluation is done internally in RiverWare Optimization units, which are typically scaled SI units, not in user display units as shown here.

More generally, constraints with shrinking have the following form:

$$\begin{aligned}&a_1x_1 + a_2x_2 + \ldots + a_{m-1}x_{m-1} + a_mx_m\\ &\le s_n(b_n - b_{n-1}) + s_{n-1}(b_{n-1} - b_{n-2}) + \ldots + s_2(b_2 - b_1) + s_1(b_1 - \text{UpperBound}) + \text{UpperBound}\end{aligned}$$

where each x_m is a variable in the LHS of each of the constraints in the shrinking sequence, each a_m is the corresponding coefficient on the variable, each s_n is the satisfaction variable associated with the constraint at the *nth* priority that contains a constraint with this common LHS, and each b_n is the right-hand-side constraint value at the *nth* priority that contains a constraint with this LHS. *UpperBound* is the formal upper bound on the LHS. It is calculated by evaluating the LHS with each variable with a positive coefficient replaced by the variable’s upper bound and each variable with a negative coefficient replaced by the variable’s lower bound. For variables that correspond directly to a slot, the variable bounds are taken directly from the Upper and Lower Bounds set in the slot configuration (see **HERE (Section 4.2.4)**).

The form is the same for a “greater-than-or-equal-to constraint” except that the *LowerBound* is used. The *LowerBound* is calculated by evaluating the LHS with each variable with a positive coefficient replaced by the variable’s lower bound and each variable with a negative coefficient replaced by the variable’s upper bound.

$$a_1x_1 + a_2x_2 + \ldots + a_{m-1}x_{m-1} + a_mx_m \\ \geq s_n(b_n - b_{n-1}) + s_{n-1}(b_{n-1} - b_{n-2}) + \ldots + s_2(b_2 - b_1) + s_1(b_1 - \mathrm{LowerBound}) + \mathrm{LowerBound}$$

In the example above, if the priority 2 constraint were not fully satisfied, then that constraint would be frozen (see **HERE (Section 2.3.1)** for details about freezing constraints). In that case, the priority 3 constraint (or any subsequent constraints with the same LHS) would never get added to problem. Because the LHS value was already “locked in” at priority 2, no lower priority constraints can change its value. Thus there is no need to add any new constraints with the same LHS to the problem. In other words, RiverWare identifies if a new constraint would shrink to a constraint that is already frozen. If that is the case, then the new constraint does not get added to the problem. This can be important to understand when analyzing the solution. No solution analysis information will be provided for a constraint that would shrink to a constraint that was already frozen because the new constraint never gets added to the problem. If all of the constraints at one priority would shrink to constraints that are already frozen, then no new constraints get added to the problem at that priority, and thus there is no need to solve the problem at that priority. As a result, that priority will not show up in the standard diagnostic output with an objective value (percent satisfaction), nor will that goal appear in the Optimization Solution Analysis Tool, **HERE (Section 7.1)**.

If some constraints from a given priority would shrink to frozen constraints, but others do not (i.e. there is still a solution at that priority, but some constraints introduced at that priority are omitted because they would not change the solution), the reporting of the satisfaction for the soft constraint derived objective depends on the type of derived objective that is selected. If the derived objective is Single Maximin or Repeated Maximin, then the satisfaction only reflects the satisfaction of the constraints that were formally added to the problem. It does not reflect the satisfaction of the constraints that were omitted (and would possibly have been violated). A tilde (~) is added to the reported satisfaction value to indicate that it is greater than the actual satisfaction from the user’s perspective because it does not include the omitted constraints. For a Summation derived objective, the reported average satisfaction is calculated as if the omitted constraints had been added to the problem. Thus it reflects the satisfaction from the user’s perspective, and no special notation is required.

If a frozen constraint shrinks to a higher priority constraint, the Optimization Solution Analysis Tool, **HERE (Section 7.1)**, lists which constraint it shrinks to as part of the frozen constraint information. Also, if a constraint is frozen, the solution analysis tool will report that all of its “shrink to” constraints were frozen as well. Without understanding how constraints shrink to earlier constraints, this can be confusing because it appears that a higher priority constraint was frozen even though the solution was not at the limit set by the constraint. Really it is the lowest priority constraint that shrank to that higher priority constraint that was forced to its limit and was frozen.

3. Combining Optimization with Simulation

A complete RiverWare Optimization run makes use of a combination of the Simulation, Optimization and Rulebased Simulation. The motivation for this approach along with the details of each of the components are described in the following sections.

3.1 Standard Controller Sequence

A standard RiverWare Optimization run uses the Simulation - Optimization - Rulebased Simulation controller approach as follows: a Simulation run is made to clear any output values, check data, and propagate input values, then the Optimization is performed, and finally, a Post-optimization Rulebased Simulation is run to set the optimal values on the objects. Every run should be made as follows:

- Switch the run controller to **Simulation**, and click Start. Let the model run to completion.
- Switch the run controller to **Optimization**, load a Goal Set (if not already loaded), and click Start. Let the model run to completion.
- Switch the run controller to **Rulebased Simulation**, load a Ruleset (if not already loaded), and click Start. Let the model run to completion.

The use of the three controllers is described in more detail in the following sections.

3.1.1 Simulation

The user starts with a model set up to run with the Simulation controller. The Simulation run serves five purposes described below.

- **Clear Output Values** - This guarantees that leftover output data from a previous solution does not inadvertently get used in the current run.
- **Check for Required Inputs** - Many Simulation methods check for required input data at the start of the Simulation run.
- **Pre-processing of Data** - If the model contains initialization rules or expression slots that evaluate at the beginning of the run to pre-process data, these get evaluated during the simulation.
- **Propagate Input Values** - Input values on linked slots get propagated across the links in the Simulation run so that they are properly registered as values for the corresponding variables.
- **Simulate where Possible** - The Simulation run can also be used to simulate all aspects of the basin for which the user does not want to optimize. For example, the user might know that one reservoir may only release minimum flows during the run. Then the Outflow for that reservoir can be set as input to the minimum flow value for all timesteps, and the Simulation run will propagate the effect of this. Then in the optimization run, this reservoir is not considered. (Note that a minimum flow requirement could alternatively be set by a high priority constraint in the Optimization Goal Set.)

In most cases, after the Simulation run most slots of interest will still display NaN.

3.1.2 Optimization

The solution of the linear goal program takes place during the Optimization run. During this run RiverWare takes the combination of the prioritized Optimization Goal Set and the model data (topology, physical characteristic data, time series data) and uses them to formulate a series of linear programs which it sends to CPLEX. CPLEX solves each linear program and returns the optimal values of the variables to RiverWare. Details of the preemptive linear goal programming solution are described **HERE (Section 2)**.

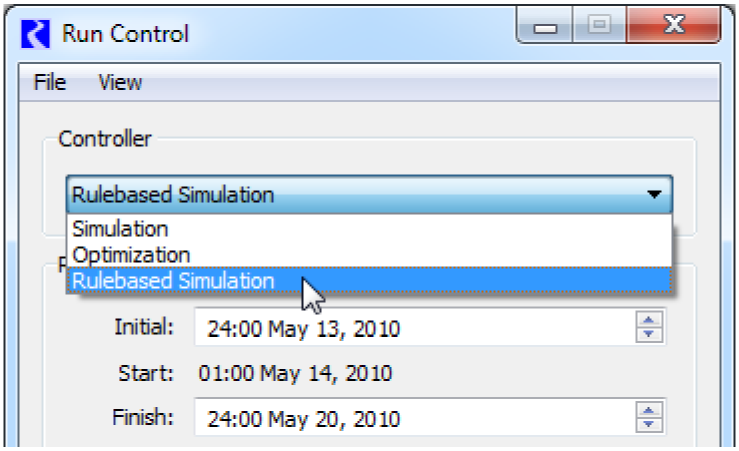
At the end of the Optimization run, RiverWare stores the final optimal value for every variable. At this point, no values have been set in slots on the workspace. That is to say that all slots which displayed NaN after the Simulation run will still display NaN after the Optimization run. Values will not be set in slots until the Post-optimization Rulebased Simulation, which is described in the following section.

3.1.3 Post-optimization Rulebased Simulation

After the run with the Optimization controller is completed, the Optimization solution is stored internally in RiverWare. Values from the solution are not yet returned to the workspace. All slots that displayed NaN at the end of the Simulation run will continue to display NaN at the end of the Optimization run. The solution is returned to the workspace by running a Post-optimization Rulebased Simulation. This is a special use of RiverWare's Rulebased Simulation (RBS) controller. The Ruleset for a Post-optimization RBS does not typically reflect the operating policy of the basin as it would for a standard RBS. The operating policy is expressed in the Optimization Goal Set (though there is technically nothing to prevent formulating rules in the Post-optimization ruleset to override results from the Optimization solution with additional policies). Rather the rules in a Post-optimization ruleset set select values on the simulation objects, typically reservoir outflows, based on the values from the Optimization solution. With these values set, the simulation can solve fully for all remaining variables.

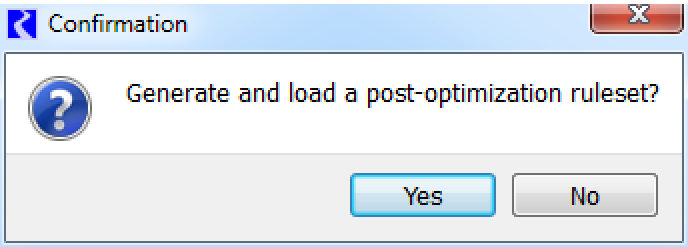
A second purpose of the Post-optimization RBS is to remove approximation error introduced by linear approximations in the Optimization solution. For example, Power must be linearized in the Optimization solution, introducing approximation error into the solution for Power. In the Post-optimization RBS, the reservoirs solve for Power using the full non-linear simulation method given the Outflow specified by the Optimization solution. This removes the approximation error introduced into Power from the linearization employed for Optimization.

After the Optimization run is complete, switch the Controller to **Rulebased Simulation**.



Controller: Rulebased Simulation
Initial: 24:00 May 13, 2010
Start: 01:00 May 14, 2010
Finish: 24:00 May 20, 2010

If no Ruleset is loaded when the controller is changed from Optimization to Rulebased Simulation, a message will appear asking if the user wants RiverWare to generate and load a ruleset.Either click Yes to use this automatically generated Ruleset for the Post-optimization RBS, or create your own custom set. These two options are described in the following sections **HERE (Section 3.2)**.



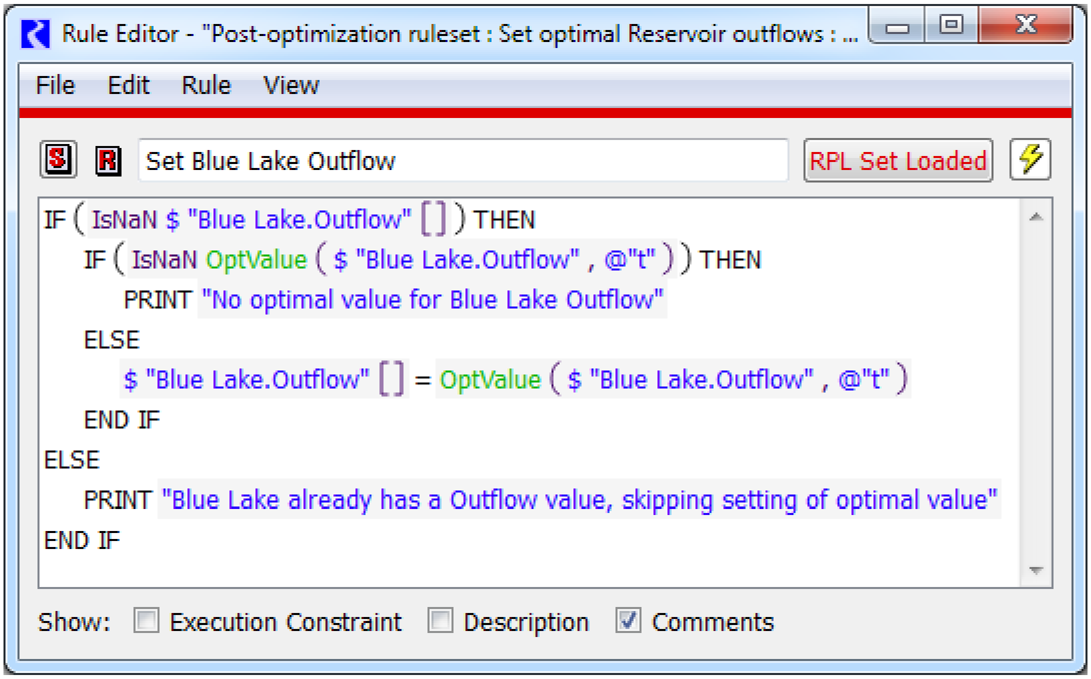
Generate and load a post-optimization ruleset?

3.2 Post-optimization Ruleset

The final step of a complete Optimization run in RiverWare is a Post-optimization Rulebased Simulation (see **HERE (Section 3.1.3)**). The ruleset used for this special case of RBS can either be generated automatically by RiverWare, or a custom set can be developed by the user. These two options are described in the following sections.

3.2.1 Automatically Generated Ruleset

The automatically generated ruleset contains two policy groups. The first group (higher priority) is called "Set optimal Reservoir outflows." The rules in this group set Reservoir.Outflow to the value from the Optimization solution by calling the OptValue() function. There is one rule for each reservoir, and they are ordered topologically. Specifically, the low to high priority order of rules corresponds to an upstream to downstream ordering of the associated reservoirs. Thus when using the default agenda order (...3,2,1), this causes the reservoirs to solve upstream to downstream. Note also that if two objects are on different tributaries or in unconnected basins, then there is no constraint on the relative order of their associated rules. Reservoirs in a "No Optimization" subbasin are excluded from the automatically generated ruleset. An example of an automatically generated Post-optimization Rule is shown below.


Rule Editor - "Post-optimization ruleset : Set optimal Reservoir outflows : ...
File
Edit
Rule
View
S
R
RPL Set Loaded

Set Blue Lake Outflow

```
IF ( IsNaN $ "Blue Lake.Outflow" [] ) THEN
    IF ( IsNaN OptValue ( $ "Blue Lake.Outflow" , @"t" ) ) THEN
        PRINT "No optimal value for Blue Lake Outflow"
    ELSE
        $ "Blue Lake.Outflow" [] = OptValue ( $ "Blue Lake.Outflow" , @"t" )
    END IF
ELSE
    PRINT "Blue Lake already has a Outflow value, skipping setting of optimal value"
END IF
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

The second policy group (lower priority) is called “Set optimal user defined variables values.” This policy group will only contain rules if the user has specified any user-defined variables (see **HERE (Section 4.2.5)**. The form of the rules is the same as those for reservoir outflows, but they set values on user defined variable custom slots. These user defined variable slots have no effect on the simulation when using the automatically generated Post-optimization Ruleset. Their values are set only for user reference.

If no ruleset is loaded when the user switches from the Optimization controller to the Rulebased Simulation controller after an Optimization run, then the user will be prompted regarding whether they would like RiverWare to generate an automatic Post-optimization Ruleset.

3.2.2 Custom Post-optimization Ruleset

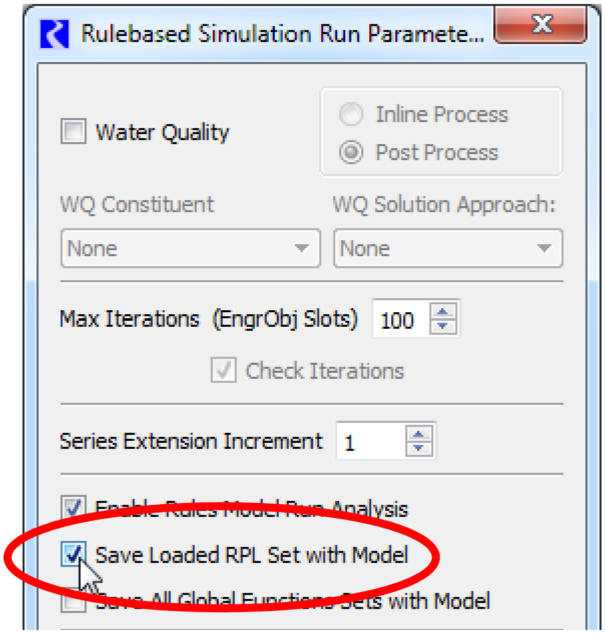
In some cases, setting only reservoir outflows might not be sufficient for a Post-opt RBS. When Outflow is set on a Reservoir in RBS, the object will solve by putting as much of the Outflow through the turbines as is possible based on physical parameters. It will only use Spill if the specified Outflow is greater than the Turbine Capacity or if the Pool Elevation is above the crest of an Unregulated Spillway. This might be sufficient in a system where it is always desirable to prevent Spill, when possible. In some systems, however, a specified Spill amount might be required by an environmental policy. The Optimization Goal Set would likely contain one or more goals corresponding to this policy. In order to get the specified Spill into the Post-optimization RBS, a rule must set Turbine Release and Spill individually, rather than setting total Outflow (or alternatively set Outflow and Spill and let the model calculate Turbine Release). Other uses for a custom Ruleset would be:

- if other water uses, such as diversions, need to be specified in the RBS based on the Optimization

solution.
- to refine the Optimization solution further in the Post-optimization RBS, possibly to make small adjustments to the flows once outputs have been calculated with the approximation errors removed.
- to add post-processing calculations to the Post-optimization rules.

A useful approach for creating a custom Post-optimization Ruleset is to begin with the automatically generated set. Then revise the rules as needed using the basic structure of the automatically generated set.

Once a custom Post-optimization Ruleset has been created, it can optionally be saved with the model file. Then the next time the model is opened, the Ruleset will automatically be opened and loaded. To save the Ruleset with the model file it must first be loaded. Then on the **Run Control** dialog select **View ➥ Rulebased Simulation Run Parameters....** In the resulting dialog, check the box for **Save Loaded RPL Set with Model**. Close the dialog, and save the model with the Ruleset now included.



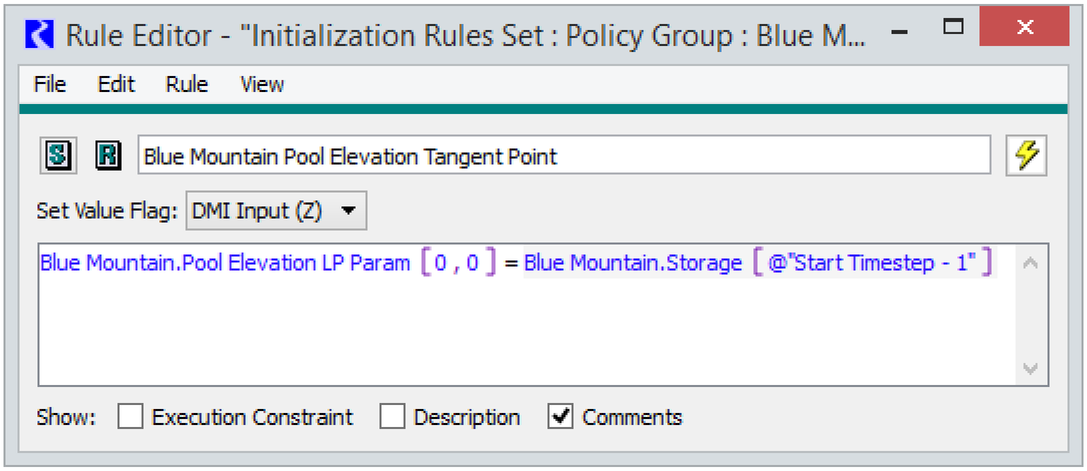
☐ Water Quality
○ Inline Process
● Post Process
WQ Constituent: None
WQ Solution Approach: None
Max Iterations (EngrObj Slots) 100
☑ Check Iterations
Series Extension Increment 1
☑ Enable Rules Model Run Analysis
☑ Save Loaded RPL Set with Model
☐ Save All Global Functions Sets with Model

3.3 Initialization Rules

Initialization Rules can be used for pre-processing of data for Optimization runs. For general information on initialization rules, click **HERE (RPLUserInterface.pdf, Section 4)**.

One use of Initialization Rules that is particular to Optimization is setting approximation points. Approximation points allow the user to control how non-linear processes are linearized. For general information on linearization click **HERE (Section 4.3)**. For guidelines on setting approximation points, click **HERE (Section 6.6)**.

In some cases, it might make sense to set the approximation points based on the specific run conditions. For example, for a reservoir that has a large seasonal variation in Pool Elevation, it might not be possible to set Tangent and Line approximation points in the Pool Elevation LP Param table or an Operating Head point in the Power LP Param table (when using the Independent Linearizations method in the Optimization Power category) that will result in sufficiently small approximation error across all conditions. In these cases, initialization rules can be used to set the approximation points such that the approximation error will be small for the expected operating range of the given run but would be large if operating in a different range in another season. For example, an initialization rule could set the Pool Elevation LP Param Tangent point equal to the initial Storage for the run. An example initialization rule is shown below.



Rule Editor - "Initialization Rules Set : Policy Group : Blue M...
File Edit Rule View
S R

Blue Mountain Pool Elevation Tangent Point

Set Value Flag: DMI Input (Z)

```
Blue Mountain.Pool Elevation LP Param [ 0 , 0 ] = Blue Mountain.Storage [ @"Start Timestep - 1" ]
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

A second rule could set the two Line points to the Tangent point plus or minus some expected delta. In this manner, the approximations will be updated automatically when the model runs, rather than needing to adjust the values manually at the start of each run.

Note: Initialization rules execute at the beginning of the Simulation controller run AND at the beginning of the Post-optimization Rulebased Simulation. They do not execute at the beginning of the Optimization controller run. Also note that initialization rules execute *before* pre-run expression slots. These execution times should be kept in mind when formulating the logic of initialization rules used for Optimization and which slot values they can reference.

3.4 Limitations of Optimization

The Optimization solution is based on a linear programming solution, and as such, has some inherent limitations as described below.

3.4.1 CPLEX License Required

RiverWare Optimization uses a third-party solver, IBM ILOG CPLEX, which is bundled with RiverWare. To access the solver, your RiverWare license must specifically allow you access to the solver library. Please contact riverware-support@colorado.edu for more information about obtaining a RiverWare license with access to CPLEX.

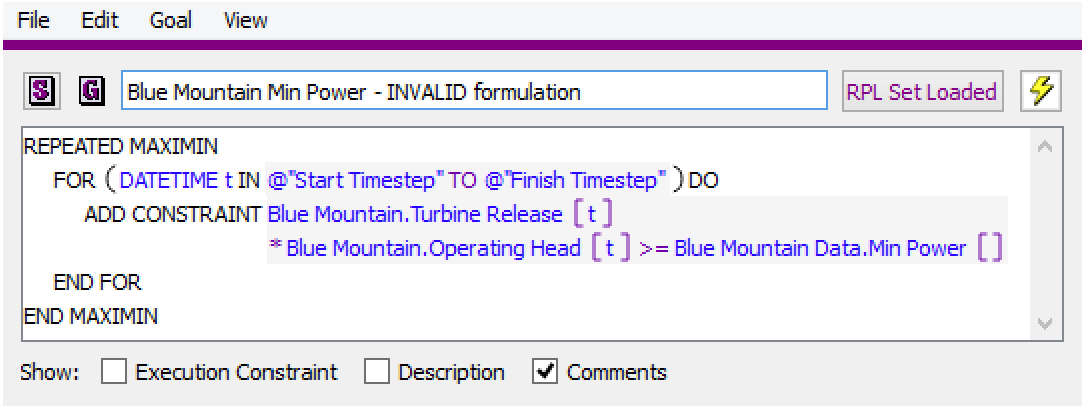
3.4.2 Linear Relationships

All relationships must be linear or piecewise linear. This means that all means that all relationships that are non-linear must be converted to linear or piecewise linear approximations. RiverWare carries out this linear approximation automatically based on approximation points specified by the user. (Click **HERE (Section 4.3)** for details about linearization.) Some common non-linear physical processes that must be linearized include:

- **Power** - a function of Operating Head and Turbine Release
- **Turbine Capacity -** a function of Operating Head
- **Tailwater** - a function of Outflow and (optionally) Tailwater Base Value (often downstream Pool Elevation)
- **Operating Head** - a function of Pool Elevation and Tailwater
- **Unregulated Spill** - a function of Pool Elevation

3.4.3 Linear Constraint Expressions

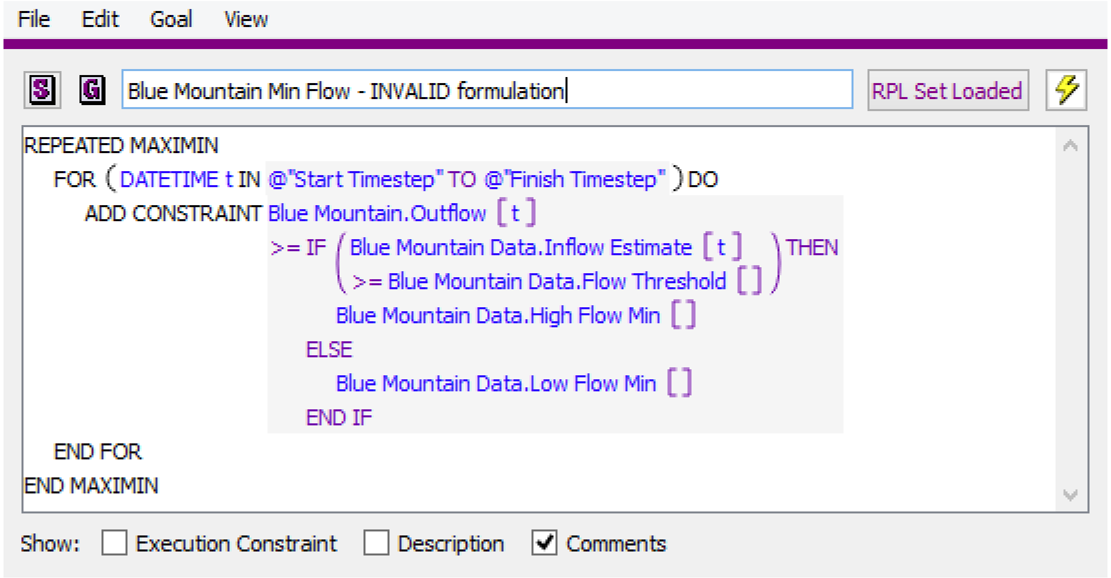
A corollary to the requirement that all relationship must be linear is that constraint statements can only contain linear expressions. This means that a constraint cannot contain a non-linear combination of variables. Also, to prevent the potential formulation of nonlinear expressions of variables in a constraint, RiverWare does not allow the use of IF/THEN expressions within a constraint statement (even if the IF/THEN expression does not contain a reference to a variable), nor the use of any RPL predefined functions within a constraint statement. Shown below are examples of constraint formulations that would not be valid.



Blue Mountain Min Power - INVALID formulation

```
REPEATED MAXIMIN
  FOR ( DATETIME t IN @"Start Timestep" TO @"Finish Timestep" ) DO
    ADD CONSTRAINT Blue Mountain.Turbine Release [ t ]
                   * Blue Mountain.Operating Head [ t ] >= Blue Mountain Data.Min Power [ ]
  END FOR
END MAXIMIN
```

This constraint contains a nonlinear combination of two variables, Turbine Release and Operating Head, and would cause the Optimization run to abort.

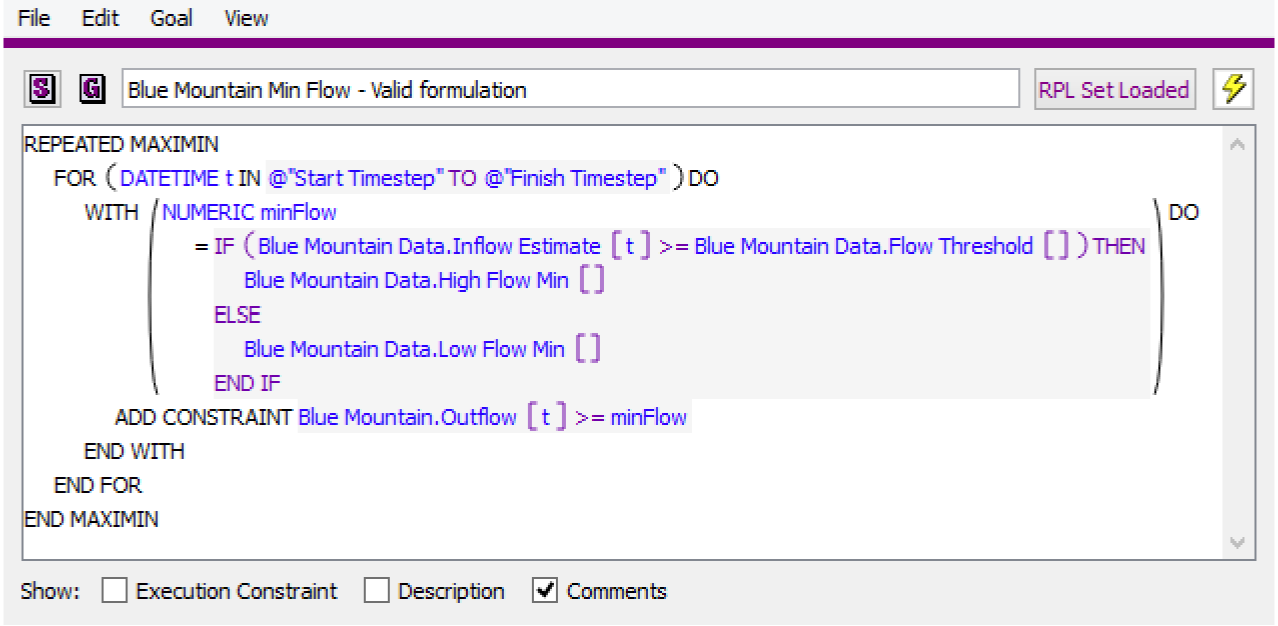


Blue Mountain Min Flow - INVALID formulation

```
REPEATED MAXIMIN
   FOR ( DATETIME t IN @"Start Timestep" TO @"Finish Timestep" ) DO
      ADD CONSTRAINT Blue Mountain.Outflow [ t ]
                     >= IF ( Blue Mountain Data.Inflow Estimate [ t ]
                             >= Blue Mountain Data.Flow Threshold [ ] ) THEN
                           Blue Mountain Data.High Flow Min [ ]
                        ELSE
                           Blue Mountain Data.Low Flow Min [ ]
                        END IF
   END FOR
END MAXIMIN
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

This constraint contains and IF/THEN expression within the constraint statement and would cause the run to abort. This type of logic can be achieved by putting the constraint statement within a WITH statement that carries out the IF/THEN logic. For example the following would be a valid goal.

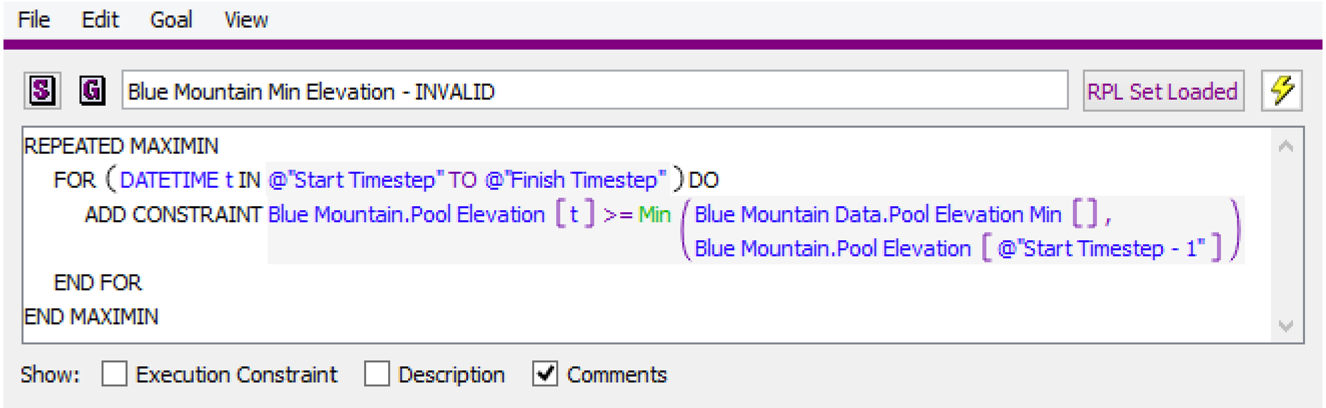


Blue Mountain Min Flow - Valid formulation

```
REPEATED MAXIMIN
   FOR ( DATETIME t IN @"Start Timestep" TO @"Finish Timestep" ) DO
      WITH ( NUMERIC minFlow
                = IF ( Blue Mountain Data.Inflow Estimate [ t ] >= Blue Mountain Data.Flow Threshold [ ] ) THEN
                     Blue Mountain Data.High Flow Min [ ]
                  ELSE
                     Blue Mountain Data.Low Flow Min [ ]
                  END IF
           ) DO
         ADD CONSTRAINT Blue Mountain.Outflow [ t ] >= minFlow
      END WITH
   END FOR
END MAXIMIN
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

This is an example of a valid use of IF/THEN logic in an Optimization goal outside of a constraint statement. (Note that in this type of use, the expression in the WITH statement cannot contain a reference to an Optimization variable. Otherwise the expression would return an invalid value, which would cause the goal to terminate without writing any constraints.)



Blue Mountain Min Elevation - INVALID

```
REPEATED MAXIMIN
   FOR ( DATETIME t IN @"Start Timestep" TO @"Finish Timestep" ) DO
      ADD CONSTRAINT Blue Mountain.Pool Elevation [ t ] >= Min ( Blue Mountain Data.Pool Elevation Min [ ] ,
                                                                 Blue Mountain.Pool Elevation [ @"Start Timestep - 1" ] )
   END FOR
END MAXIMIN
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

This goal contains a RPL predefined function, **Min**, within the constraint statement and would cause the run to abort. User-defined functions can be used within constraint statements as long as the functions do not contain nonlinear functions of variables.

3.4.4 Approximation Error

A side effect of the requirement for linear relationships is approximation error. All variables that are linearized will have approximation error in the Optimization solution. Typically the largest and most significant approximation error shows up in the linearization of Power. In the Post-optimization Rule-based Simulation, the rules typically set flow values based on the values for those flow variables from the Optimization solution. Then the objects dispatch and calculate Power and other quantities based on the nonlinear Simulation methods. Therefore the calculation of Power in the Post-optimization RBS, once the approximation error from linearization is removed, will differ from the Power value in the Optimization solution. The Power approximation error tends to be most noticeable when the Optimization policy contains constraints that generation must equal some value (load). In these cases, the Optimization solution will often say that the constraint is 100% satisfied, but the Power calculated in the Post-optimization RBS will not match the constraint value exactly.

3.4.5 Solution Time

The computational requirements of the preemptive linear goal programming solution limit the size of the Optimization problem that can be solved in a “reasonable” amount of time. For large models (many objects and/or many goals) keeping the run length reasonable typically means limiting the number of timesteps. The definition of “reasonable” solution time is dependent on the use(s) of the model, and the user will need to determine what is an acceptable solution time. When determining the upper limit on the number of timesteps, one important point to consider is that the Optimization solution time for a given model does not increase linearly with the number of timesteps, as it generally does for RBS. The relationship between solution time and number of timesteps is typically somewhere between quadratic and cubic. The total solution time is a combination of a number factors, including but not necessarily limited to:

- Number of objects (roughly correlates to the number of variables)

- Number of timesteps
- Number of constraints
- Number of goals
- Difficulty of the solution (Objectives that force a lot constraints to be frozen or that have soft constraint violations can take longer to solve.)
- Hardware - number of available processors, processor speed, memory

3.4.6 Perfect Foresight

The ability of the Optimization solution to “look ahead” in both time and space (global solution) is one of the benefits of Optimization. Optimization can be used to make sure you do not compromise you ability to satisfy constraints in the future when optimizing for today. However, if not managed appropriately, perfect foresight can be a potential weakness. Because the solution does not inherently put a higher value on one time period over another, in some cases it can exploit the solution to maximize an objective farther into the future, which has a result of degrading the objective in the near term. A common example would be if the value of Energy is somewhat higher later in the run. The solution might save as much water as possible until this higher value period and then generate at max capacity over the high value period. This might be considered a “good” solution, but in some cases there is more uncertainty about input data farther out in the forecast horizon, and it is not desirable for the solution to “over-optimize” these future periods at the expense of near term operations. This potential consequence of Optimization’s “prefect foresight” is important to keep in mind when considering the appropriate run horizon for an Optimization model.

3.4.7 Limited Objects and Method Selections

Not all object types are supported by Optimization. Additionally, only a subset of Simulation user methods can be used in conjunction with Optimization. For a complete listing of the objects and methods that are available for Optimization, refer to the Objects section **HERE (Section 5)**. When creating a model that will be used for Optimization, it is important to make Simulation method selections that are compatible with Optimization.

3.4.8 Debugging

The Optimization solution is generally more difficult to debug than Rulebased Simulation. Because it is a global solution, there is typically not a single cause for a bad output or “odd” result. It can be the combination of multiple constraints and input data at multiple locations and multiple timesteps. Often the solution can be exploited to improve some objective function if it is not properly constrained, resulting in unrealistic or undesirable operations. In these cases it can be difficult to determine what exactly is contributing to the undesirable result. Click **HERE (Section 7)** for descriptions of tools that can assist in debugging the Optimization solution.

4. Variables, Constraints and Objectives

4.1 RPL Optimization Goal Set

The operating policy of a basin is expressed in a RPL Optimization Goal Set. This section describes the components of the Optimization Goal Set. An Optimization Goal Set uses much of the same syntax as a Rulebased Simulation ruleset **HERE (RPLUserInterface.pdf, Section 1)**. However, there some key differences between the formulation of rules and Optimization goals as described in the following sections.

4.1.1 Prioritized Policy

In Optimization, each aspect of the basin operating policy is expressed in a goal, and each goal is given a unique priority. In this way, goals for Optimization are analogous to rules for Rulebased Simulation. The contents of each goal are either constraints or objectives. Details about the contents of goals are described in the next two sections **HERE (Section 4.1.2)** and **HERE (Section 4.1.3)**. This section focuses on the top level structure of the goal set.

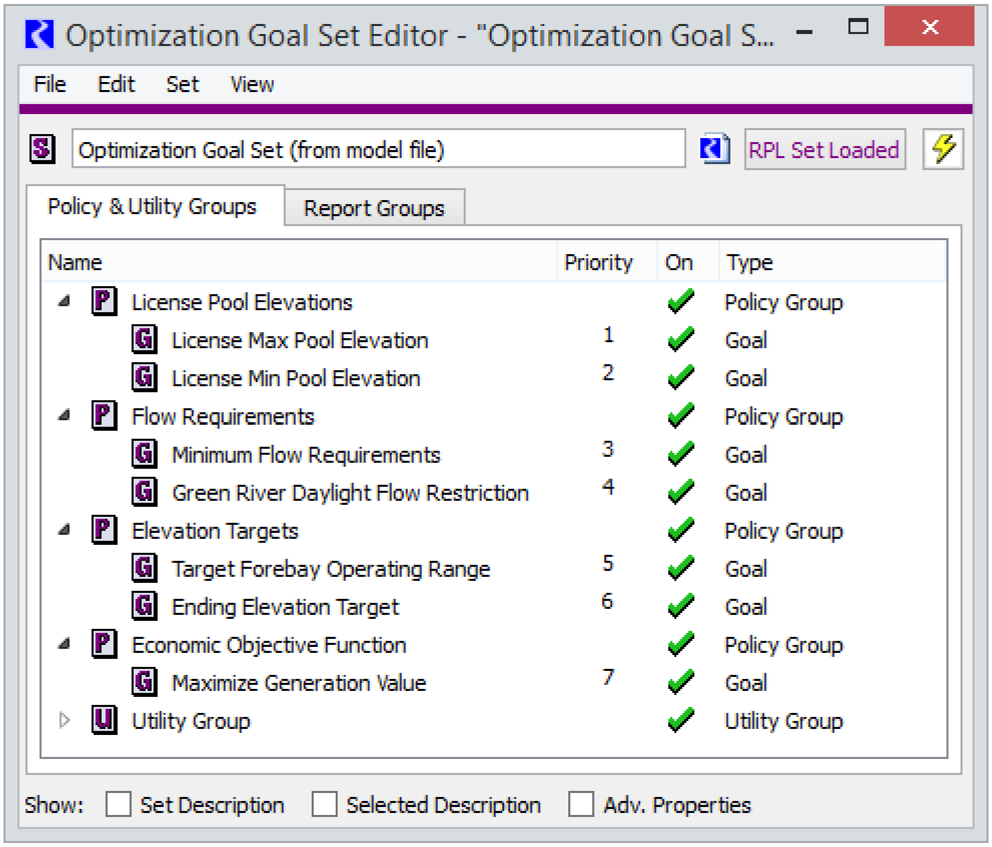
RBS rules execute from lowest priority to highest priority. Optimization goals on the other hand, execute from highest priority to lowest priority (lowest number to highest number). Each priority that contains an objective, either a Maximize or Minimize objective or a Soft Constraint Set derived objective, will trigger a solution of the Optimization problem. The problem at each priority will include the constraints from all higher priority goals, along with their maximized satisfaction, and any new constraints added at the current priority. Thus at each priority, degrees of freedom are removed from the problem reducing the solution space. If the goal contains only hard constraints, or if the logic in the goal results in no new constraints getting added to the Optimization problem, then there will not be a new solution of the Optimization problem at that priority.

RBS rules are normally ordered to execute in an upstream to downstream order, object-by-object. Generally there is no need to order Optimization goals in an upstream to downstream order. The priorities of the goals should correspond to their “true” priorities in operations. The highest priority goals should correspond to the most important operating constraints, those that should essentially never be violated.

A single goal can contain constraints for multiple objects, but it is generally recommended that a single goal correspond to a single operating policy. For example, the highest priority goal might contain the license maximum Pool Elevation constraints for all reservoirs at all timesteps. The second goal might contain the license minimum Pool Elevation constraints for all reservoirs at all timesteps. This is generally better practice than placing the minimum and maximum elevation constraints in the same goal or placing the minimum elevation constraint in the same goal (same priority) as a minimum flow constraint. When multiple policies are combined in a single goal, it is not always obvious how the policies will trade off with one another if it is not possible to satisfy all constraints at that priority.

Medium priority goals generally contain constraints that you expect to get satisfied most of the time but that may not be possible to satisfy under some operating conditions, for example under extreme high or low flows. Lower priority goals typically contain target operating constraints, targets that you would like to meet under ideal conditions but are expected to be frequently violated due to higher priority constraints and input data (e.g. inflows). For example, you might have a low priority goal to keep flows equal across all reservoirs on each timestep, but most of the time you cannot keep flows exactly equal due to other requirements on the system. Typically the lowest priority contains an objective function to optimize with the remaining degrees of freedom given all higher priority constraints. For example, the objective might maximize the total value of hydropower generation during the run within the operational limits specified by higher priority constraints.

A simple example of an Optimization Goal Set is shown below.



Optimization Goal Set (from model file)

Name	Priority	On	Type
License Pool Elevations		✓	Policy Group
License Max Pool Elevation	1	✓	Goal
License Min Pool Elevation	2	✓	Goal
Flow Requirements		✓	Policy Group
Minimum Flow Requirements	3	✓	Goal
Green River Daylight Flow Restriction	4	✓	Goal
Elevation Targets		✓	Policy Group
Target Forebay Operating Range	5	✓	Goal
Ending Elevation Target	6	✓	Goal
Economic Objective Function		✓	Policy Group
Maximize Generation Value	7	✓	Goal
Utility Group		✓	Utility Group

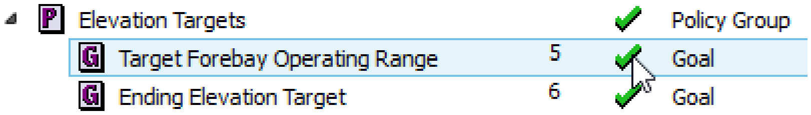
In this goal set, the License Max and Min Pool Elevations are at the highest priorities. You would therefore expect that these elevation limits are almost never violated. The Minimum Flow Requirement is at the next priority, so you would also expect it to be satisfied most of the time. However, if there were a case with low flows and it was not possible to meet both the minimum elevation requirement and the minimum flow requirement, then the solution would meet the minimum elevation and would violate the minimum flow requirement. It would get the flow as close to the minimum as possible, and in doing so, it would take the reservoir(s) to the minimum elevation.

The Target Forebay Operating Range goal is at a lower priority. This suggests that ideally you would like to keep the forebay(s) (Pool Elevation) within some specified range, but you are not going to violate either of the flow requirements in goals 3 and 4 to do so.

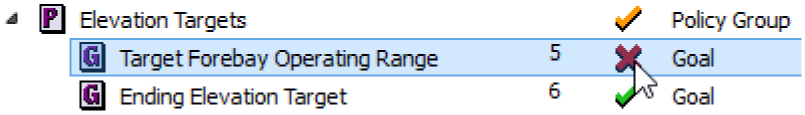
The final goal is maximizing the value of generation. Normally an objective such as this would use as much water as possible to produce as much generation as possible. The Ending Elevation Target goal at priority 6 is likely in place to prevent the maximum value goal from draining all reservoirs to the bottom of their operating range. The solution will only maximize the economic value of generation within the limits specified by the higher priority goals. It can only use the degrees of freedom that remain after solving to meet all of the higher priority constraints. The solution would go outside of the target forebay operating range (priority 5) if necessary to meet a flow requirement (priority 3 or 4), but it would not go outside of the target operating range just to increase the generation value (priority 7).

Note that, in practice, most Optimization Goal Sets contain many more goals than the example shown here in order to model all of the operational requirements the system. Also note that all of the goals correspond to operational policies. There are no goals added to express the physical constraints that define the system (e.g. mass balance and routing). All physical constraints get included in the Optimization problem automatically by RiverWare (see **HERE (Section 4.2.3)**). There is no need for the user to formulate the physical constraints.

Like rulesets, Optimization Goal Sets can be organized into policy groups and utility groups, as seen in the example above. The policy groups contain goals and the utility groups contain functions used within the goals. As with rules, the policy groups are for organizational purposes only. The solution evaluates each goal individually in priority order. It does not solve by policy group. Priorities of individual goals and policy groups can be shifted by clicking and dragging the goal/group name to the new priority location. Individual goals and policy groups can be activated and deactivated by clicking on the green check mark or red ✖ next to the goal or policy group name.



Clicking on the green check mark will change it to a red ✖.



Clicking on the red ✖ will re-activate the goal or policy group and change it back to a green check mark.

4.1.2 Goal “Types”

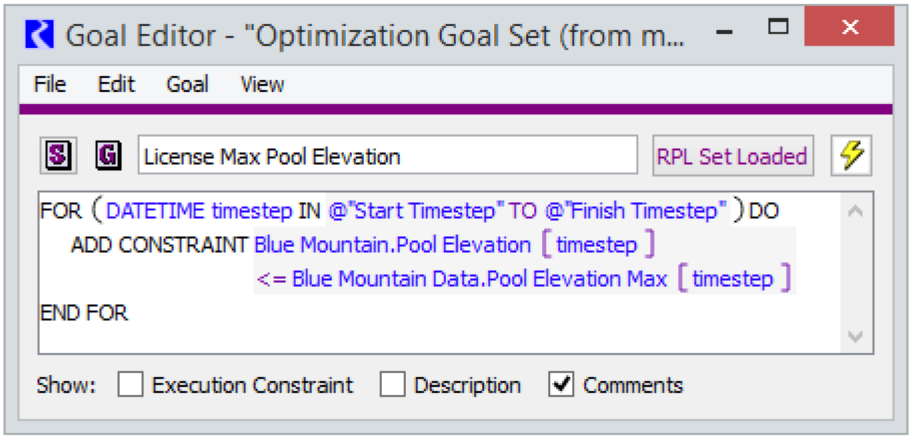
Technically there is only a single type of goal that is added to a goal set. However, goals can be thought of as falling in one of three general categories: hard constraints, soft constraints or an objective. Like rules, Optimization goals consist of statements. The “type” of each goal is determined by the type of statements that are added to it. The types of statements that can be added to a goal are described **HERE (Section 4.1.3)**. The different goal types are described in the following sections.

4.1.2.1 Hard Constraints

A “hard constraint” goal includes Constraint statements (see **HERE (Section 4.1.3.3)**) that are not within a Soft Constraint Set derived objective (see **HERE (Section 4.1.3.2)**). A goal with only hard constraints does not trigger a solution of the Optimization problem. It only adds new constraints to the problem. The constraints will not have an effect on the solution until the next priority that includes either a Soft Constraint Set derived objective or a Maximize or Minimize objective.

Hard constraints should be used with caution. If any hard constraint cannot be fully satisfied at the next priority with a solution, then an infeasible solution (i.e. no solution) will be returned. These infeasibilities can often be difficult to debug. It can be tempting to consider the highest priority operating constraints as “hard” constraints that are never violated, for example, a license Pool Elevation limit. However it is better practice to formulate these highest priority policies as soft constraints at the highest priorities in the goal set. Once the soft constraints are satisfied at a high priority, they will effectively be treated as hard constraints when solving at all remaining priorities. In the rare case that the high priority constraints cannot be fully satisfied, the soft constraint approach will still return a solution rather than an infeasibiliy. The solution returned will provide the user with more information about where, when and why the violation occurred than if the model simply returns an infeasible solution.

As an example, the following would technically be a valid hard constraint formulation:



License Max Pool Elevation

```
FOR ( DATETIME timestep IN @"Start Timestep" TO @"Finish Timestep" ) DO
   ADD CONSTRAINT Blue Mountain.Pool Elevation [ timestep ]
                  <= Blue Mountain Data.Pool Elevation Max [ timestep ]
END FOR
```

However, it is recommend instead that the contents of goal such as this with operational policy be placed within a Soft Constraint Set derived objective statement.

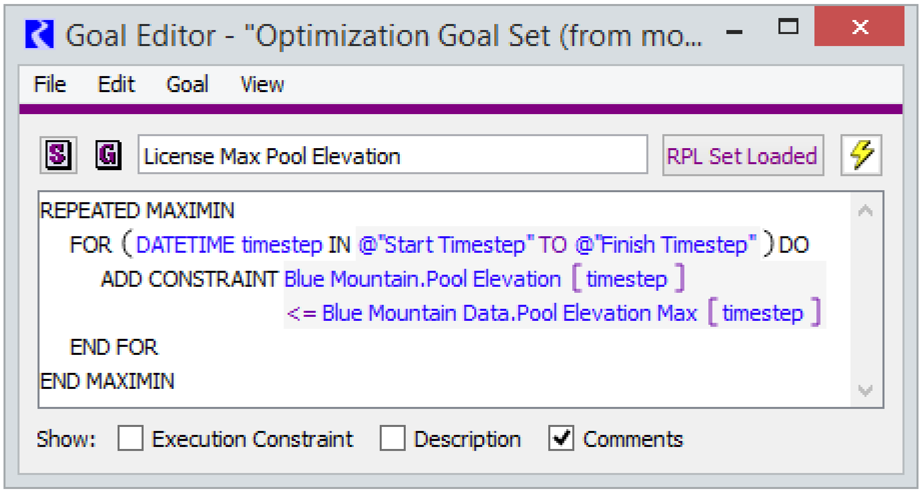
One common use of hard constraints is to formulate the defining constraints for user-defined variables (see **HERE (Section 4.2.5)**). In fact, defining constraints are typically the only recommended use of hard constraints. Again, these defining, hard constraints should be formulated in a manner that they are guaranteed to always be feasible. They are typically placed at the highest priorities in the goal set.

4.1.2.2 Soft Constraints

The large majority of goals in an Optimization Goal Set are typically “soft constraint” goals. These goals have a Soft Constraint Set derived objective statement (**HERE (Section 4.1.3.2)**) at the top level within the goal, and within the Soft Constraint Set statement are one or more Constraint statements. The Soft Constraint Set triggers a solution of the optimization problem as long as the goal actually adds new constraints to the problem. (Some goals may contain logic to only add constraints under certain conditions, for example seasonal constraints. If the logic within the goal results in no new constraints being added, then there is no need for a solution after the goal is processed because there is no change to the Optimization problem.) RiverWare automatically converts the goal into a derived objective to maximize the satisfaction of the constraints in the goal given the objective value (satisfaction level) of all higher priority goals. RiverWare provides four different approaches for maximizing the satisfaction of soft constraints, which are described **HERE (Section 2.2.3)**. Details for implementing soft constraints are described **HERE (Section 4.1.3.2)**.

Soft constraint goals should be used to formulate all operational constraints (e.g. Pool Elevation limits, minimum Outflow requirements, minimum Power generation requirements, etc.). They are also often used for lower priority “target” operations and “solution quality” constraints, constraints that prevent the Optimization solution from exploiting remaining degrees of freedom to produce operations that would never be carried out in practice. These lower priority target constraints are usually not expected to be satisfied under all conditions.

An example of a maximum Pool Elevation policy formulated as a soft constraint goal using the Repeated Maximin derived objective is shown below.



License Max Pool Elevation

```
REPEATED MAXIMIN
  FOR ( DATETIME timestep IN @"Start Timestep" TO @"Finish Timestep" ) DO
    ADD CONSTRAINT Blue Mountain.Pool Elevation [ timestep ]
                   <= Blue Mountain Data.Pool Elevation Max [ timestep ]
  END FOR
END MAXIMIN
```

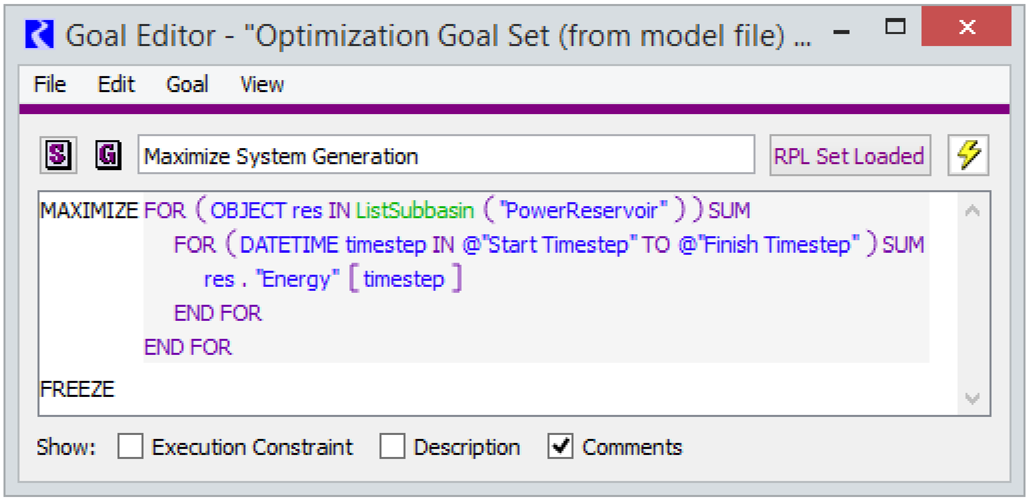
Show: ☐ Execution Constraint ☐ Description ☑ Comments

4.1.2.3 Objectives

The final type of goal is an objective. This type of goal is used to maximize or minimize a specified quantity. Technically it maximizes a linear expression of variables. For example, the objective might maximize the total value of hydropower generation over the entire run period.

Typically objectives are at or near the lowest priorities in the goal set. They maximize/minimize the objective given any remaining degrees of freedom after all higher priority constraints. The solution from an objective goal will often use up most remaining degrees of freedom in the Optimization problem. Details for implementing an objective are described **HERE (Section 4.1.3.1)**.

An example of an objective to maximize the total system hydropower generation over the entire run is shown below.



Maximize System Generation

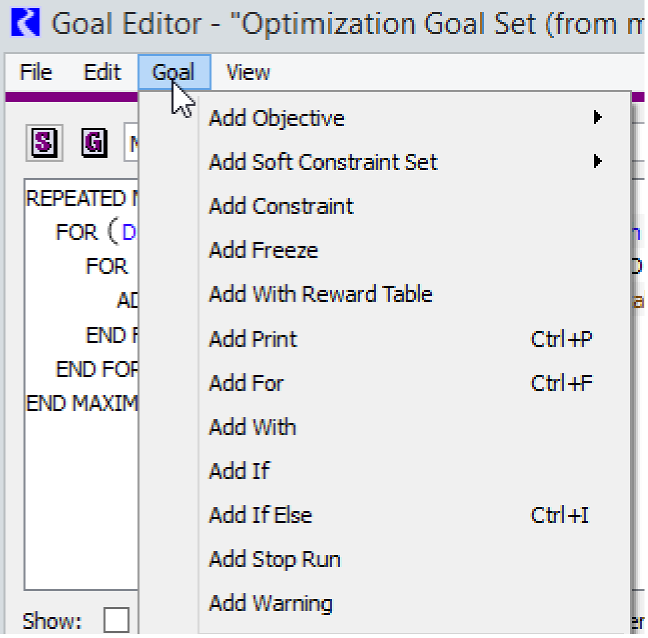
```
MAXIMIZE FOR ( OBJECT res IN ListSubbasin ( "PowerReservoir" ) ) SUM
           FOR ( DATETIME timestep IN @"Start Timestep" TO @"Finish Timestep" ) SUM
              res . "Energy" [ timestep ]
           END FOR
        END FOR
FREEZE
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

This goal loops over all power reservoirs in the model and over all timesteps and creates an expression that is the sum of the Energy variable (slot) for those reservoirs and timesteps. The objective maximizes that sum.

4.1.3 Statements in Goal Sets

This section describes the pieces of an Optimization Goal. At the top level, an Optimization Goal consists of statements. These statements are available from the **Goal** menu in the Goal Editor dialog. The list of available statements is slightly different than the list available for rules **HERE (RPLUserInterface.pdf, Section 2.3)**. A description of each is given below.



4.1.3.1 Objective

An Objective statement can be a minimize or maximize objective and has the form:

MAXIMIZE <expr>

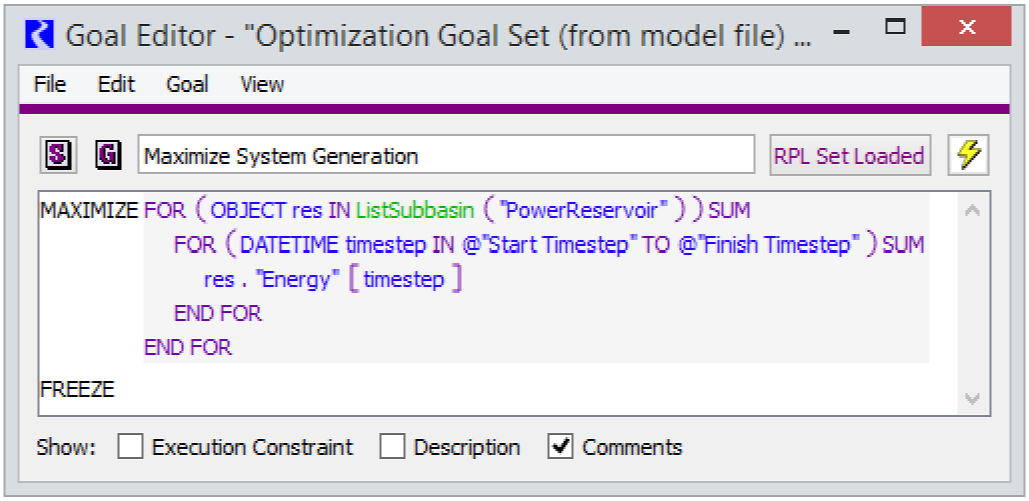
or

MINIMIZE <expr>

The user then defines the expression to minimize or maximize. The expression within the objective statement may contain multiple terms and summations over multiple slots and/or timesteps as long as it is a linear combination of variables.

Note: A Freeze statement must be added at the end of an Objective statement in order to freeze the objective value. Otherwise the objective value from the goal will not necessarily be maintained at lower priorities.

An example of an objective to maximize the total system hydropower generation over the entire run is shown below.



Maximize System Generation

```
MAXIMIZE FOR ( OBJECT res IN ListSubbasin ( "PowerReservoir" ) ) SUM
           FOR ( DATETIME timestep IN @"Start Timestep" TO @"Finish Timestep" ) SUM
              res . "Energy" [ timestep ]
           END FOR
         END FOR
FREEZE
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

This goal loops over all power reservoirs in the model and over all timesteps and creates an expression that is the sum of the Energy variable (slot) for those reservoirs and timesteps. The objective maximizes that sum.

4.1.3.2 Soft Constraint Set

When a Soft Constraint Set is added, the Optimization solution will attempt to satisfy all of the constraints in the set completely. If it is not possible to fully satisfy all constraints introduced at the current priority due to physical conditions and/or higher priority goals, the solution will attempt to satisfy the constraints as fully as possible based on the one of the three available solution methods (derived objectives) described below. The user selects the solution method for each Soft Constraint Set. The satisfaction level of the constraint set is frozen once the Optimization problem has been solved at that priority (i.e. once a constraint has been satisfied at a high priority, it will not be violated to meet a lower priority goal or objective). Only one Soft Constraint Set should be added to a goal, but a single Soft Constraint Set can contain multiple individual Constraint statements.

For the mathematical details about how constraint satisfaction is defined, click **HERE (Section 2.2.2)**.

Soft Constraint Set Solution Approaches:

- **Summation**: This approach will minimize the total deviation from the constraint values within the Soft Constraint Set (formally it maximizes the total satisfaction).

$$\text{Maximize}\sum_{\text{all i}} S_{p,\,i}$$

where there is one satisfaction variable $S_{p,i}$ for each constraint i added at the current priority p.

The potential downside of this approach is that it can tend to place all of the deviation on one or just a few variables. For example, if a minimum flow constraint cannot be met for every timestep, it might minimize the total violation by satisfying the constraint at all but one timestep and putting a very large violation on that one timestep. Typically, this is not the preferred solution in a water resources context. The concentration of large violations when using a Summation Soft Constraint Set can be prevented by applying a With Reward Table statement within the Summation. Click **HERE (Section 4.1.3.5)** for more details about using the With Reward Table statement. Additional details about the Summation derived objective are given **HERE (Section 2.2.3.1)**.

Note: A Freeze statement must be added at the end of a Summation derived objective statement in order to freeze the objective value (satisfaction). Otherwise the satisfaction of the soft constraints from the Summation goal will not necessarily be maintained at lower priorities.

- **Single Maximin**: Instead of minimizing the total violation, this approach will minimize the single largest violation (formally it maximizes the minimum satisfaction). This prevents a single, very large violation by spreading the violation out over multiple variables or timesteps.

$$\text{Maximize } S_p$$

where S_p is a single satisfaction variable applied to all constraints added at the current priority p.

The potential downside of this approach is that it does nothing to improve the satisfaction of the remaining constraints that are not contributing to the largest violation. This issue is addressed by the next type of derived objective, Repeated Maximin. Additional details about the Single Maximin derived objective are given **HERE (Section 2.2.3.2)**.

Note: A Freeze statement must be added at the end of a Single Maximin derived objective statement in order to freeze the objective value (satisfaction). Otherwise the satisfaction of the soft constraints from the Single Maximin goal will not necessarily be maintained at lower priorities.

- **Repeated Maximin**: This approach begins with the single maximin. It then freezes the satisfaction of the largest violation and re-solves problem by minimizing the next largest violation. It repeats this process until either all violations have been minimized or all remaining constraints are fully satisfied.

$$\textbf{While } S_{p,i-1} < 1$$

$$\text{Maximize } S_{p,\,i}$$

$$S_{p,\,i} \geq S_{p,\,i-1} \qquad \text{for } i > 1$$

where $S_{p,i}$ is a single satisfaction variable applied to all constraints added at the current priority p that have not yet been frozen prior to the *ith* Repeated Maximin iteration.

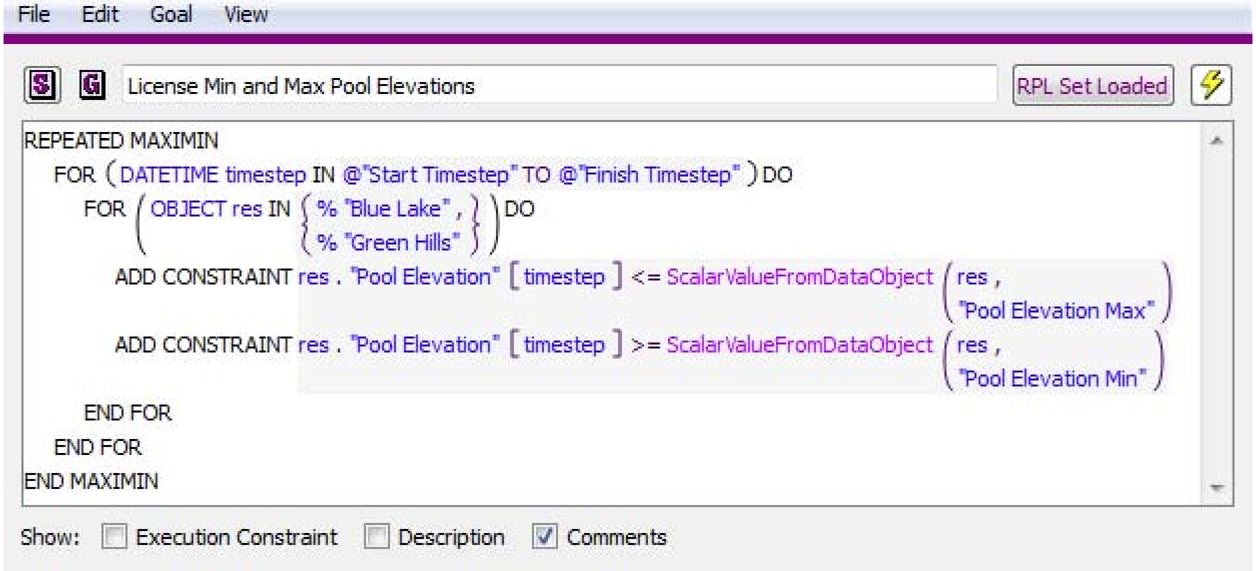
The Repeated Maximin approach tends to spread out deviations as evenly as possible, which is often the desired solution in a water resources context. The potential downside of the Repeated Maximin approach is that in some cases it can require numerous iterations when it is not possible to satisfy all constraints. Depending on the size of the model this can cause a significant increase run time. The Summation approach always requires only a single solution at the given priority, so it tends to be faster, but it does not, in general, provide the same solution quality as Repeated Maximin. In some cases, a reasonable balance between solution quality and run time can be provided by applying a With Reward Table statement within a Summation. Click **HERE (Section 4.1.3.5)** for more details on Summation With Reward Table. Additional details about the Repeated Maximin derived objective are given **HERE (Section 2.2.3.3)**.

4.1.3.3 Constraint

Constraints are formulated as either <=, >= or == expressions. Constraint statements can be added within a Soft Constraint Set, a For statement, an If statement or a With statement (or within a nested combination of these statement types). If a Constraint statement is added that is not within a Soft Constraint Set, it will be treated as a hard constraint. Note that it is not necessary to add physical constraints in Optimization Goals. Physical constraints are added automatically by RiverWare as needed based on the model topology and user-selected methods; see **HERE (Section 4.2.3)**.

The logical operators are added from the RPL Palette. The user can formulate a constraint with variables and/or scalar values on either side of the logical operator as long as all expressions within the constraint are linear combinations of variables and maintain dimensional consistency. RiverWare will automatically convert all constraints to a canonical form with variables (and coefficients, where applicable) on the left-hand-side and scalar values on the right-hand side when the Optimization problem is processed internally.

The following figure shows a goal containing basic minimum and maximum Pool Elevation constraints that are being added as Repeated Maximin soft constraints for two reservoirs for all timesteps in the run.



License Min and Max Pool Elevations

```
REPEATED MAXIMIN
   FOR ( DATETIME timestep IN @"Start Timestep" TO @"Finish Timestep" ) DO
      FOR ( OBJECT res IN { % "Blue Lake" ,
                            % "Green Hills" } ) DO
         ADD CONSTRAINT res . "Pool Elevation" [ timestep ] <= ScalarValueFromDataObject ( res ,
                                                                                          "Pool Elevation Max" )
         ADD CONSTRAINT res . "Pool Elevation" [ timestep ] >= ScalarValueFromDataObject ( res ,
                                                                                          "Pool Elevation Min" )
      END FOR
   END FOR
END MAXIMIN
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

The minimum and maximum Pool Elevation values are being retrieved from slots on a data object corresponding to each reservoir by a user-defined function called ScalarValueFromDataObject.

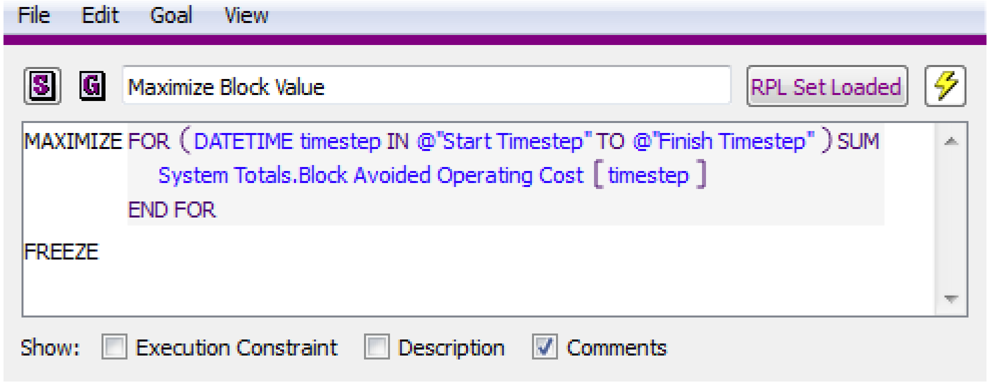
Note that If and With expressions cannot be added within a Constraint statement. Also RPL pre-defined functions cannot be used within Constraint statements; however user-defined functions can be used within a Constraint statement as long as they result in linear constraint expressions. RPL pre-defined functions can be used in the outer level Optimization Goal statements that contain Constraint expressions, for example within the boolean expression of an If statement that contains a Constraint statement.

4.1.3.4 Freeze

A Freeze statement locks in the value of an objective after solving a minimize or maximize objective. A Freeze statement should be added after an objective statement when it is desired to actually preserve the solution resulting from solving the objective function (i.e. for standard optimization objectives). In some cases it might be necessary to maximize or minimize an objective to calculate values for ‘temporary’ use in the solution before solving the ‘true’ objective at a later priority. In this case, it is not desired to freeze value of the objective function for the final solution, and a Freeze statement would not be added.

The figure below shows a Freeze statement added after a typical objective to maximize the value of hydropower over the entire run. Freeze statements must also be added after Summation and Single Maximin Soft Constraint sets in order to lock in the derived objective value (preserve the constraint satisfaction). It is not necessary to add a Freeze statement after a Repeated Maximin Soft Constraint Set.

The satisfaction variables for Repeated Maximin are frozen automatically..

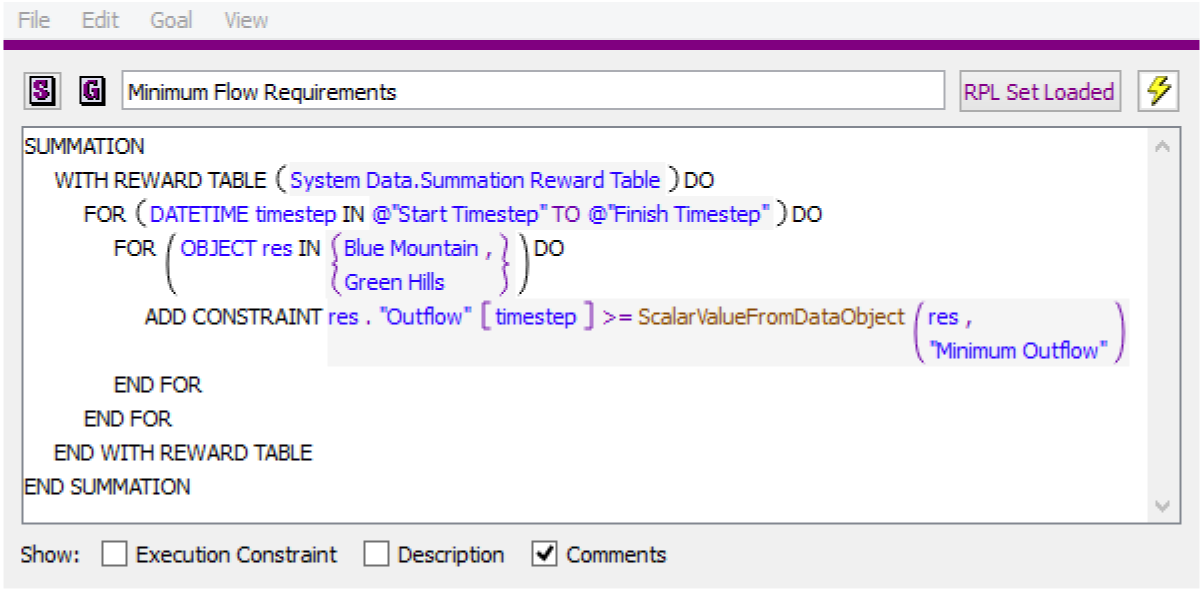


Maximize Block Value

```
MAXIMIZE FOR ( DATETIME timestep IN @"Start Timestep" TO @"Finish Timestep" ) SUM
             System Totals.Block Avoided Operating Cost [ timestep ]
         END FOR
FREEZE
```

4.1.3.5 With Reward Table

A With Reward Table statement allows a reward function to be applied to the satisfaction variables within a Summation objective for a soft constraint set. The With Reward Table statement can only be used within a Summation soft constraint set. Using the With Reward Table statement in any other context will result in an error.



Minimum Flow Requirements

```
SUMMATION
   WITH REWARD TABLE ( System Data.Summation Reward Table ) DO
      FOR ( DATETIME timestep IN @"Start Timestep" TO @"Finish Timestep" ) DO
         FOR ( OBJECT res IN { Blue Mountain ,  } ) DO
                             { Green Hills      }
            ADD CONSTRAINT res . "Outflow" [ timestep ] >= ScalarValueFromDataObject ( res ,
                                                                                    "Minimum Outflow" )
         END FOR
      END FOR
   END WITH REWARD TABLE
END SUMMATION
```

The reward table allows a piecewise function to be applied to the satisfaction variables within Summation objective. Instead of using the standard Summation derived objective for maximizing satisfaction, the derived objective becomes

where *R* is the reward function for the satisfaction variables.

$$\text{Maximize} \sum_{\text{all i}} R_i$$

$$R_i = f(S_i)$$

Typically this approach is used to penalize larger violations more than smaller violations. For example, a common approach is to minimize the sum of squared violations. This tends to “smooth out” large violations by spreading them out over multiple time steps rather than concentrating large violations on a few time steps. Applying the appropriate reward function within a Summation soft constraint set can often produce a result that compares to the Repeated Maximin solution in terms of solution quality but has the benefit of only requiring a single solution at the given priority and can therefore be much faster.

The piecewise reward function must be described in a user created table slot. The table slot is given as an argument to the With Reward Table statement.

WITH REWARD TABLE <slot expr>

The reward table slot must have two columns with units of NONE. The first column contains satisfaction values. The second column contains the corresponding reward values. The values in both columns must be between 0 and 1. If the user does not include rows for a satisfaction of 0 and 1, RiverWare will add these rows internally when processing the piecewise reward function with values of 0 and 1 used for the reward. The reward table must be concave. (Reward must be a concave function of Satisfaction.)

An example is provided below for a reward table that corresponds to minimizing the sum of squared violations

Example Reward Table: Minimize the Sum of Squared Violations

Conceptually the desired derived objective is to minimize the sum of squared violations.

$$\text{Minimize} \sum_{\text{all i}} v_i^2$$

Violations are scaled from 0 to 1, where 1 corresponds to the maximum violation based on a higher priority constraint or bound. Thus an equivalent objective to minimize the sum of squared violations is

$$\text{Maximize} \sum_{\text{all i}} (1 - v_i^2)$$

For performance reasons, the RiverWare Optimization solution maximizes satisfaction variables rather than minimizing violations. (Click **HERE (Soft Constraint Set)** for details about how the satisfaction variables are defined.).

$$S_i = 1 - v_i$$

The maximize objective then becomes

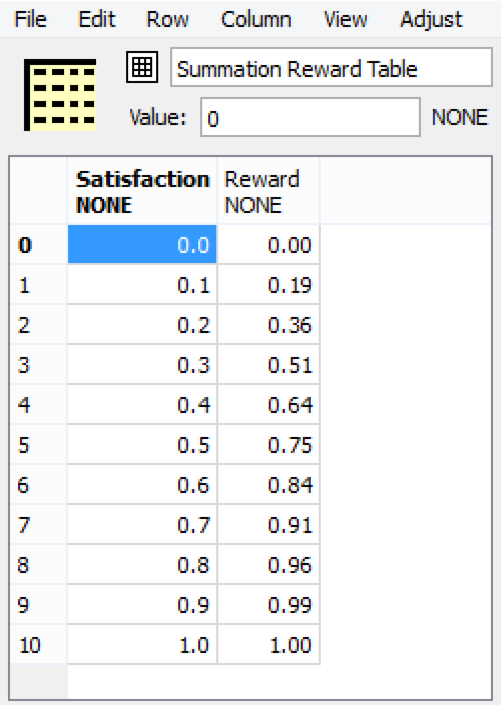
$$\text{Maximize} \sum_{\text{all i}} (1 - (1 - S_i)^2)$$

This can be described by a piecewise linear function in the following table which linearizes the function in 10 discreet segments.

Reward Table to Minimize Squared Violations

v_i	v_i^2	s_i	$1 - v_i^2$
1.0	1.00	0.0	0.00
0.9	0.81	0.1	0.19
0.8	0.64	0.2	0.36
0.7	0.49	0.3	0.51
0.6	0.36	0.4	0.64
0.5	0.25	0.5	0.75
0.4	0.16	0.6	0.84
0.3	0.09	0.7	0.91
0.2	0.04	0.8	0.96
0.1	0.01	0.9	0.99
0.0	0.00	1.0	1.00

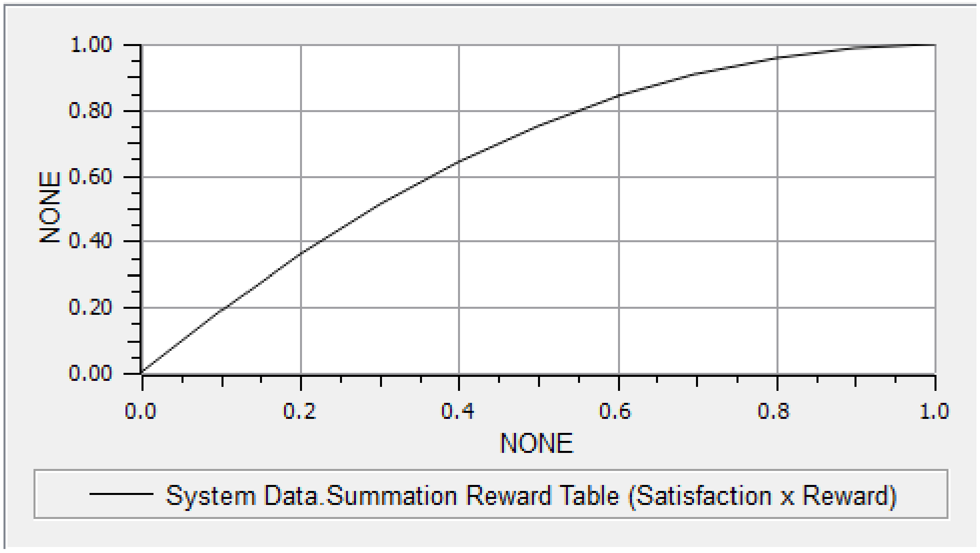
The table entered in RiverWare is taken from the last two columns as shown below and in the plot that follows.



Summation Reward Table

Value: 0 NONE

	Satisfaction NONE	Reward NONE
0	0.0	0.00
1	0.1	0.19
2	0.2	0.36
3	0.3	0.51
4	0.4	0.64
5	0.5	0.75
6	0.6	0.84
7	0.7	0.91
8	0.8	0.96
9	0.9	0.99
10	1.0	1.00



More generally, any concave table may be used. For example, a table can be created for minimizing cubed violations. Tables with more rows may degrade performance (increase run time) and numerical stability. There is a tendency for optimal solutions to be at the discrete values in the table. This may be more noticeable when tables with fewer rows (larger segments) are used. For example, if the reward table only had three rows corresponding to satisfaction values of 0, 0.5 and 1, it might be common to see the solution at a satisfaction of either 0.5 or 1.0. Some experimentation may be required to determine the best table size for a given goal.

It is permissible to use the same reward table for many goals, and this is probably the easiest way to get started. If the results suggest that it may be beneficial to use different tables for some goals, then this can easily be changed. It is also possible to use different reward tables within the same Summation, if desired, by including multiple With Reward Table statements within a single Summation Soft Constraint Set, each referencing a different table slot.

4.1.3.6 Print

A Print statement evaluates its expression and formats the result into a message. The blue message is displayed in the Diagnostics Output window when the Print Statements diagnostics group is enabled. The basic print statement is:

```
PRINT <expr>
```

where the **<expr>** is any expression or concatenated expressions which can be fully evaluated and represented as a string. A Print statement can be useful within an If statement to notify the user that system is within a specified condition. For more information on the Print statement click **HERE (Diagnostics.pdf, Section 3.3)**.

4.1.3.7 For

Iterative loops can be very useful for adding similar constraints for multiple objects or over multiple timesteps. These iterative loops are added through a For statement. An index variable is assigned to a new value for each iteration of the loop. Inside the loop is one or more Constraint statements which should use the index variable to write a different constraint for each iteration of the loop. The default For loop is:

```
FOR (NUMERIC index IN <list expr>) DO
    ADD CONSTRAINT <boolean expr>
END FOR
```

where the number of loops is determined by the number of elements in the **<list expr>**, the **NUMERIC** label indicates the expression data type of the elements in the **<list expr>**, and the **index** is the variable name which will take on the value of each element for use inside the loop. All of these parts of the For statement may be modified. It is also possible to nest a For statement inside another For statement (by replacing the **ADD CONSTRAINT** statement with another For statement) allowing the loops to iterate over two or more variables.

Because Optimization provides a global solution in time, there is no valid notion of “Current Timestep;” therefore, an important use of For loops is to write constraints over all timesteps. Such a For loop would have the form:

```
FOR (DATETIME index IN @”Start Timestep” TO @”Finish Timestep”) DO
    ADD CONSTRAINT Object.Slot[index] == <numeric expr>
END FOR
```

The time range for the constraint can be modified by changing the expressions in the list expression.

4.1.3.8 With

A value can be set on a local variable using a With statement. A With statement evaluates an expression and sets the result to a local variable with a given name and type. The statements contained within the With statement may then reference the variable. The default With statement is:

```
WITH (NUMERIC val = <numeric expr>) DO
    ADD CONSTRAINT <boolean expr>
END WITH
```

4.1.3.9 If

The addition of a constraint can be made conditional on a boolean expression using an If statement. For example, a constraint might only be applied if the timestep is within a specified season or if a valid value is present within a specified slot on a data object. Without an Else, the statement will do nothing if the boolean expression evaluates to false.

```
IF (<boolean expr>) THEN
    <statement>
END IF
```

One important difference in the use of an If statement in an Optimization Goal compared to RBS rules is that an If statement in Optimization cannot contain expressions with Optimization variables. For example,

```
IF (Reservoir.Pool Elevation[@”Start Timestep”] >= DataObject.Max Elevation[]) THEN
```

would cause the Optimization run to abort with an error message because Pool Elevation is a variable in the Optimization problem and thus cannot be known when the If statement is evaluated (unless Pool Elevation at the Start Timestep happened to be specified as an input).

Also, If expressions cannot be added within a Constraint statement; however, Constraint statements can be added within an If statement.

4.1.3.10 If Else

This is similar to the If statement, but an alternative statement will be executed if the boolean expression evaluates to false.

```
IF (<boolean expr>) THEN
    <statement>
ELSE
    <statement>
END IF
```

4.1.3.11 Stop Run

Abort the run and post the provided expression as a diagnostic error message.

```
STOP RUN <expr>
```

A Stop Run might be used if it is known that a certain condition (that can be evaluated prior to the execution of the current goal) will produce an undesirable result, and thus a full Optimization run is not desired. A Stop Run would typically be contained within an If statement.

4.1.3.12 Warning

This posts a brown warning to the diagnostic output window with a brief message followed by the value of the expression. A Warning statement does not stop the run (as does a Stop Run statement), and in contrast to a Print statement, it is shown regardless of diagnostics settings. A Warning message would typically be contained within an If statement.

4.1.4 Timestep References in Optimization

Optimization provides a global solution in time and space (i.e. it solves all timesteps and all objects simultaneously); therefore there is not a valid notion of “Current Timestep” in Optimization. As such, DateTime expressions such as **@”Current Timestep”**, **@”t”** and empty brackets **[]** (representing current controller timestep on series slots) are not valid in an Optimization Goal Set. Use of these DateTime values will cause the Optimization run to abort. In order to evaluate an expression or apply a constraint over all timesteps (or a specified range of timesteps) a For loop is used where the index variable is a DateTime set to the values of a list of timesteps in the run. For example, adding a constraint on a reservoir’s Pool Elevation for every timestep could have the following form:

```
FOR (DATETIME index IN @”Start Timestep” TO @”Finish Timestep”) DO
    ADD CONSTRAINT Reservoir.Pool Elevation[index] <= 2,350 “ft”
END FOR
```

4.2 Optimization Variables

An optimization problem technically consists of a set of equations that include linear combinations of optimization variables. The following sections describe how those variables are defined.

4.2.1 Slots and Variables

In RiverWare, some Optimization variables are directly related to slots. Other variables are related to slots by a linearized expression. Still other variables have no relation to slots, and some slots have no relation to variables. RiverWare decides automatically which variables to add to the Optimization problem based on what is referenced in the Optimization Goal Set.

All slots that are related to Optimization variables, either directly or by linearized expressions, are series slots. These could be mult-slots or agg series slots, which are forms of series slots. In the case of agg series slots, each column corresponds to a separate variable. Note that while it is common to refer to a slot and a variable synonymously, technically each individual variable within the Optimization problem, is associated with a single slot *at a single timestep*.

4.2.1.1 Slots as CPLEX Variables

Some slots are related directly to variables. For example, Outflow and Storage on a reservoir object are common variables. Each of these slots at each timestep corresponds directly to an individual variable in the Optimization problem. Some additional examples of slots that are directly related to variables are provided below for various objects.

Canal: Flow1

Confluence: Outflow

Reach: Outflow

Reservoir: Outflow, Spill, Storage, Release, Diversion, Cumulative Storage Value

Power Reservoir: Turbine Release, Pumped Storage Outflow

Slope Power Reservoir: Qc1, Level Storage, Pool Elevation

4.2.1.2 Linearized Slots

In some cases slots correspond to a linear combination of variables. One example is Power. The manner in which Power is linearized depends on the method selected in the Optimization Power category. With the commonly used Independent Linearizations method, Power is defined by a piecewise linear function of Turbine Release. Each individual variable, or “piece,” is multiplied by the slope of that segment of the power curve. The sum of the pieces is equal to the total Turbine Release. Examples of slots that correspond to linearized expressions of variables are provided below.

Canal: Delta Elevation, Elevation 1, Elevation2 , Flow 2

Confluence: Inflow 1, Inflow 2

Reach: Inflow

Reservoir: Canal Flow, Energy In Storage, Inflow, Pool Elevation, Spill Cost, Spilled Energy, Spilled Power

Power Reservoir: Best Turbine Flow, Energy, Future Value Used Energy, Hydro Capacity, Operating

Head, Power, Pumped Storage Inflow, Spill Capacities, Tailwater Base Value, Tailwater Elevation, Turbine Capacity

Pumped Storage Reservoir: Pump Capacity, Pump Energy, Pump Power, Pump Q Capacity

Slope Power Reservoir: Backwater Elevation, Inflow 2, Wedge

4.2.1.3 Variables Unrelated to Slots

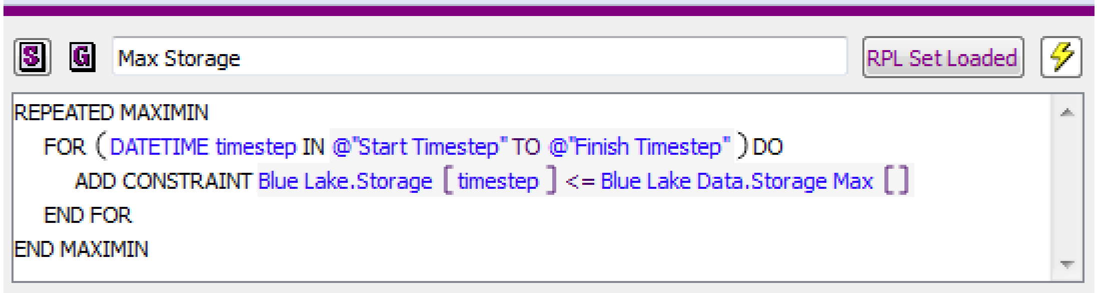
Some variables have no relation to a slot in RiverWare. For example, the satisfaction level of each Repeated Maximin iteration or the satisfaction of each constraint in a Summation derived objective is added to the Optimization problem as a variable. Click **HERE (Section 2.2.2)** for details about how satisfaction variables are defined.

4.2.1.4 Slots Unused by Optimization

Some slots apply to Simulation only and do not get used by the Optimization solution. For example, the Total Inflows and Inflow Sum slots on a Reservoir object do not get used by the Optimization solution. The components that make up Total Inflow and Inflow Sum in Simulation are expressed as individual variables and are incorporated into the mass balance constraint for the Reservoir, but there is not a separate Total Inflows or Inflow Sum variable that is defined. (Total Inflow and Inflow Sum will get calculated in the final results from the Post-optimization Rulebased Simulation.)

4.2.2 Constraints Define Variables

In an optimization run, the controller executes the user’s policy which consists of a series of constraints and objectives. RPL expressions add constraints to the current optimization problem, cause the problem to be solved with a given objective, or freeze some aspects of the current problem solution. When the policy refers to the value of a slot at a given timestep, RiverWare automatically adds constraints to the problem which define the variable associated with the slot value at that timestep. As an example, assume the following goal is at priority 1 in the Goal Set.



```
REPEATED MAXIMIN
   FOR ( DATETIME timestep IN @"Start Timestep" TO @"Finish Timestep" ) DO
      ADD CONSTRAINT Blue Lake.Storage [ timestep ] <= Blue Lake Data.Storage Max [ ]
   END FOR
END MAXIMIN
```

RiverWare will add the constraint shown in the goal to Optimization problem for every timestep (as a soft constraint), replacing the scalar slot Blue Lake Data.Storage Max with the actual value in the slot. Also, because this constraint references the Blue Lake Storage for every timestep, it will add the defining constraint for Blue Lake Storage, which is the mass balance constraint, for every timestep.

$$\text{Blue Lake.Storage}_t = \text{Blue Lake.Storage}_{t-1} + \text{TimestepLength} \times (\text{Blue Lake.Inflow}_t - \text{Blue Lake.Outflow}_t)$$

Notice that the mass balance constraint has also defined the variables for Inflow and Outflow at every timestep. Other variables could be included in the mass balance constraint as well, depending on the selected user methods on the reservoir (e.g. Hydrologic Inflow, Evaporation). RiverWare will also automatically add the upper and lower bound constraints for each of these three variables at each timestep based on the upper and lower bounds in the Slot Configuration for each; see **HERE (Section 4.2.4)**. If Inflow and Outflow are linked to slots on other objects, additional constraints will be added automatically which define the link relationship. In addition, a defining constraint will be added for Outflow setting it equal to the sum of Turbine Release and Spill, along with the upper and lower bound constraints for each of these new variables. So adding a single policy constraint can actually add numerous defining constraints for variables to the Optimization problem.

4.2.3 Physical Constraints

RiverWare automatically adds all relevant physical constraints to the Optimization problem. For example, the mass balance constraint for a reservoir is

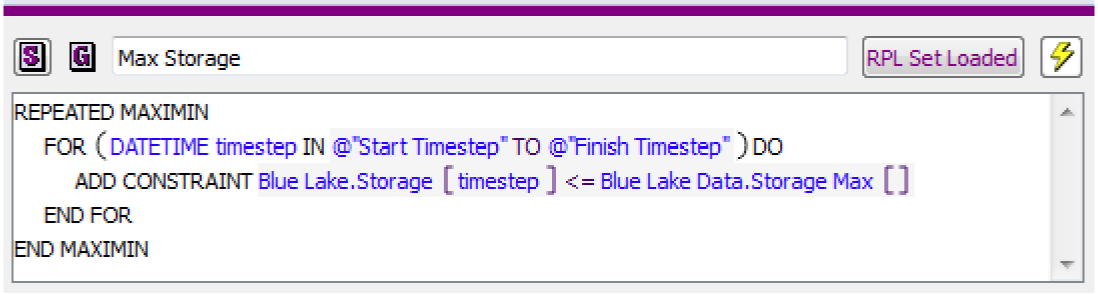
$$\mathrm{Storage}_t = \mathrm{Storage}_{t-1} + \mathrm{TimestepLength} \times \left(\mathrm{Inflow}_t - \mathrm{Outflow}_t + \sum \mathrm{Gains}_t - \sum \mathrm{Losses}_t\right)$$

This is the defining constraint for the Storage, Inflow and Outflow variables at time *t*. RiverWare determines which individual variables to include in Gains and Losses depending on the method selection on the reservoir. For example, Evaporation could be an additional loss variable in the mass balance constraint. For a reach, the defining physical constraints are determined by the selected routing method.

RiverWare only adds physical constraints as necessary based on the policy in the Optimization Goal Set. In this way it minimizes the size of the Optimization problem in order to achieve maximum computational efficiency. Physical constraints are always treated as hard constraints.

4.2.4 Variable Bounds

Upper and lower bounds are required for all decision variables. The bounds represent the absolute minimum and maximum for the slot (variable). The variable bounds remain constant and are automatically converted by RiverWare into hard constraints on the variable for every time step in the Optimization solution. They get used in defining constraint satisfaction for soft constraints. Soft constraint satisfaction is always scaled based on the difference between the constraint value (right-hand-side) and the previous bound, the RHS value from an earlier (higher priority) constraint with the same left-hand-side. If there is no earlier constraint with the same left-hand-side, then the previous bound is taken from the variable bounds. In other words, if a constraint in the Optimization goal set is the first constraint of its form, then the satisfaction of the constraint is calculated relative to the slot (variable) bound. For example, assume the following max storage goal is the first goal in the Optimization Goal Set (priority 1).



```
REPEATED MAXIMIN
   FOR ( DATETIME timestep IN @"Start Timestep" TO @"Finish Timestep" ) DO
      ADD CONSTRAINT Blue Lake.Storage [ timestep ] <= Blue Lake Data.Storage Max [ ]
   END FOR
END MAXIMIN
```

The satisfaction S_t for the constraint on a single timestep would be

$$S_t \le \frac{\text{Blue Lake.Storage[t]} - \text{StorageUpperBound}}{\text{Blue Lake Data.Storage Max} - \text{StorageUpperBound}}$$

$$0 \le S_t \le 1$$

Click **HERE (Section 2.2.2)** for further details about how satisfaction variables in soft constraints are defined.

Often the variable bounds are based on the actual physical minimum and maximum. For example, the Storage Lower and Upper Bounds should be the smallest and largest Storage values in the Elevation Volume table. Turbine Release should be limited to zero as the Lower Bound and Turbine Capacity for the Upper Bound. In other cases, the absolute physical minimum or maximum may not be well-defined, such as maximum Inflow (or if economic variables are included in the model).

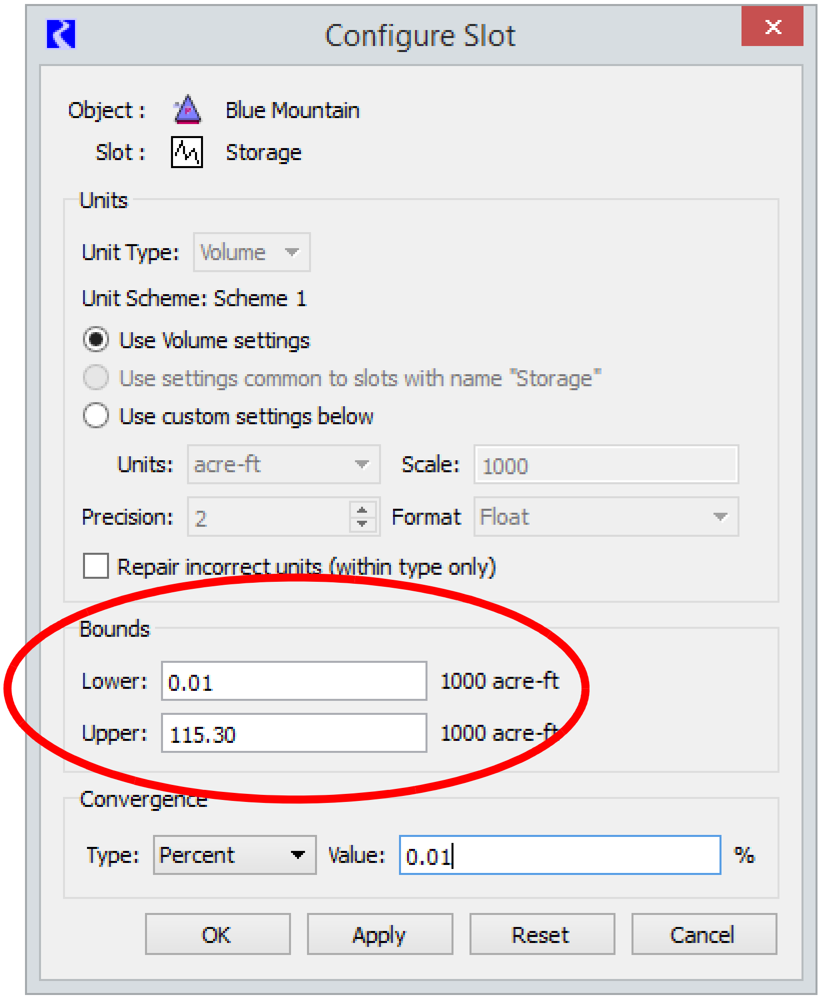
In these cases of variables without an obviously defined minimum or maximum, the slot bounds should not unnecessarily constrain the problem. These types of variables will typically be limited, either directly or indirectly, through the constraints in the Optimization Goal Set. It is also important, however, that the slot bounds not be set arbitrarily large or small (i.e. orders of magnitude different than the realistic limits) as this can result in numerical instability due to scaling issues in the optimization problem.

It is important that slot bounds not be confused with policy constraints, nor be used to apply policy constraints. For example, a hydropower plant might have a non-zero minimum generation requirement that it must meet, but the Power slot Lower Bound should still be zero. The minimum generation requirement would be set through a constraint in the Optimization Goal Set. The slot bounds become hard constraints in the optimization problem and essentially remain constant in the model, whereas policy constraint values (usually stored in custom slots) are typically incorporated as soft constraints and may change for different runs.

The slot bounds do not affect the Simulation solution. They are only applied in the Optimization solution. The Simulation solution is limited only by the physical characteristics described in table slots such as the Elevation Volume Table and Plant Power Table.

The variable bounds are displayed on the Configure Slot dialog. From the slot dialog select **View** ➥

Configure.



Object : Blue Mountain
Slot : Storage
Unit Type: Volume
Unit Scheme: Scheme 1
● Use Volume settings
○ Use settings common to slots with name "Storage"
○ Use custom settings below
Units: acre-ft
Scale: 1000
Precision: 2
Format Float
☐ Repair incorrect units (within type only)
Lower: 0.01 1000 acre-ft
Upper: 115.30 1000 acre-ft
Type: Percent
Value: 0.01 %

Click **HERE (Section 6.5.1)** for further details about setting slot bounds in a model.

4.2.5 User-defined Variables

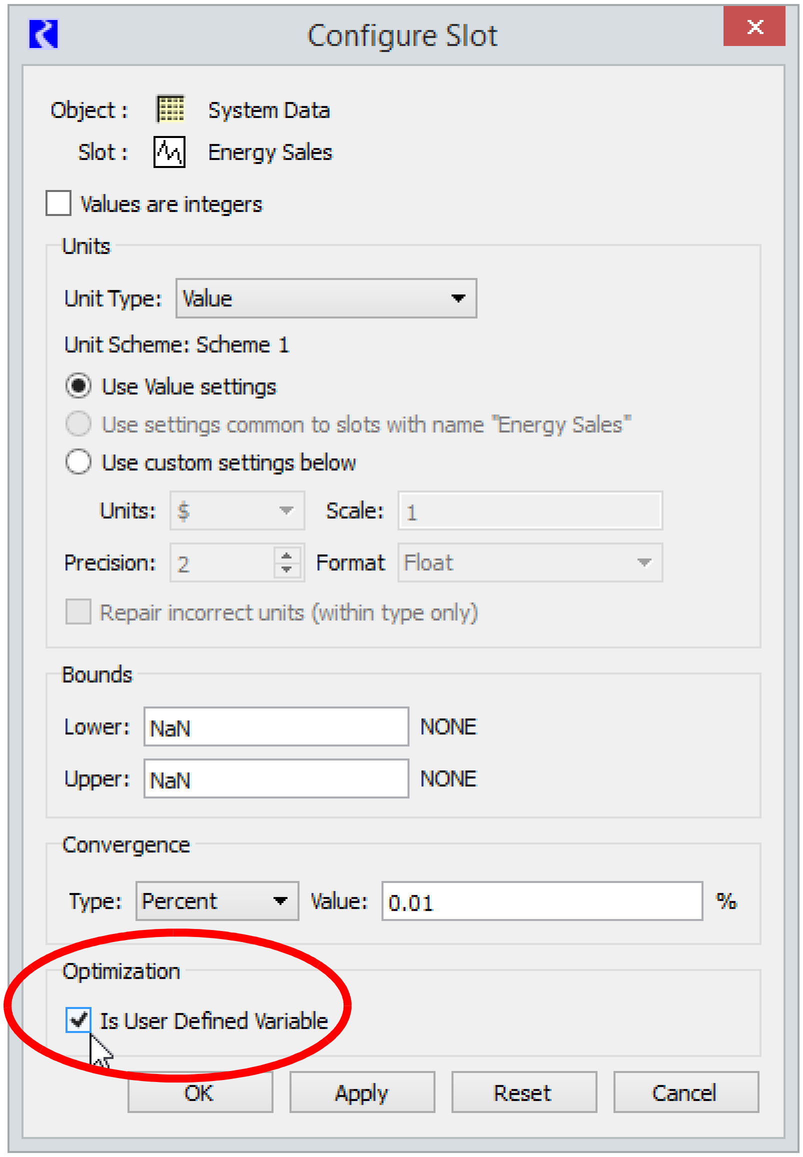
In addition to the set of standard variables that are automatically included in the Optimization problem, it is also possible to add customized user-defined variables. User-defined variables correspond to custom slots that the user adds to the model. The user must define the variables in a constraint that they add to the Goal Set. Then they can use the variable in a later policy constraint or objective.

There are five steps for implementing a user defined variable.

1. Add a series slot to an object in the model specifying appropriate units.
2. Specify that the slot is a user-defined variable.

From the slot dialog select **View ➥ Configure**. In the resulting **Configure Slot** dialog, check the box for

“Is User Defined Variable” near the bottom of the dialog.



Configure Slot

Object : System Data

Slot : Energy Sales

☐ Values are integers

Units

Unit Type: Value

Unit Scheme: Scheme 1

● Use Value settings

○ Use settings common to slots with name "Energy Sales"

○ Use custom settings below

Units: $ Scale: 1

Precision: 2 Format Float

☐ Repair incorrect units (within type only)

Bounds

Lower: NaN NONE

Upper: NaN NONE

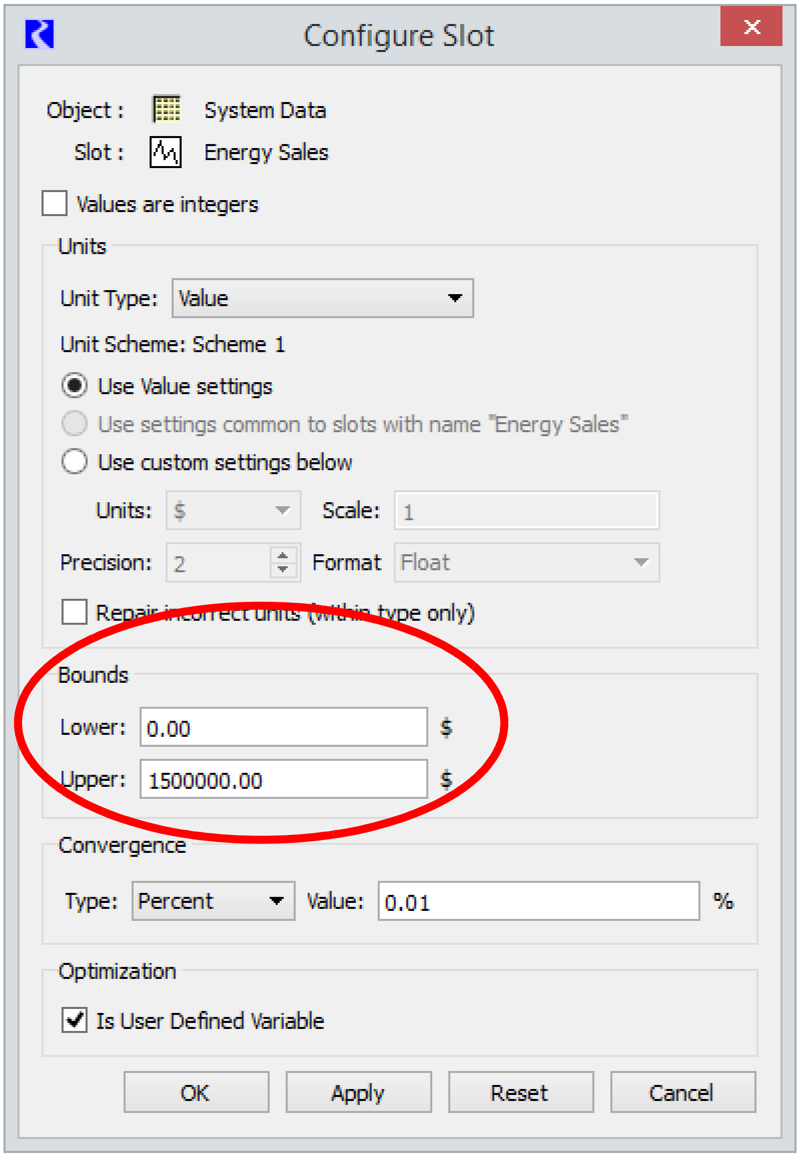
Convergence

Type: Percent Value: 0.01 %

Optimization

☑ Is User Defined Variable

3. Set appropriate Lower and Upper Bounds for the variable in the **Configure Slot** dialog. Click **HERE (Section 4.2.4)** for more information about variable bounds.



Configure Slot

Object : System Data
Slot : Energy Sales

☐ Values are integers

Units
Unit Type: Value
Unit Scheme: Scheme 1
● Use Value settings
○ Use settings common to slots with name "Energy Sales"
○ Use custom settings below
Units: $ Scale: 1
Precision: 2 Format Float
☐ Repair incorrect units (within type only)

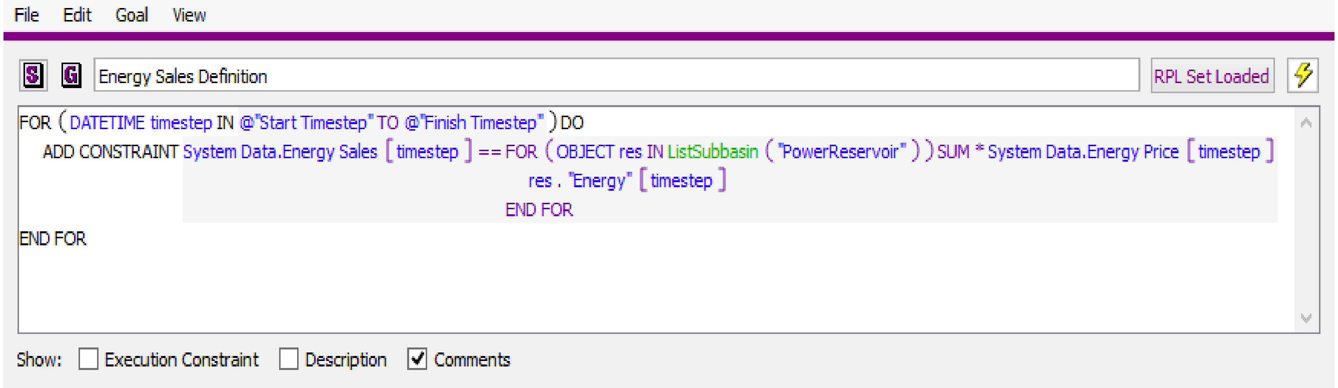
Bounds
Lower: 0.00 $
Upper: 1500000.00 $

Convergence
Type: Percent Value: 0.01 %

Optimization
☑ Is User Defined Variable

4. Define the variable in a constraint in the Goal Set.

An example is shown below that defines the variable Energy Sales at each time step as the total energy generation from all power reservoirs multiplied by the Energy Price. (This assumes that Energy Price is an input.) Variables are typically defined using hard constraints.



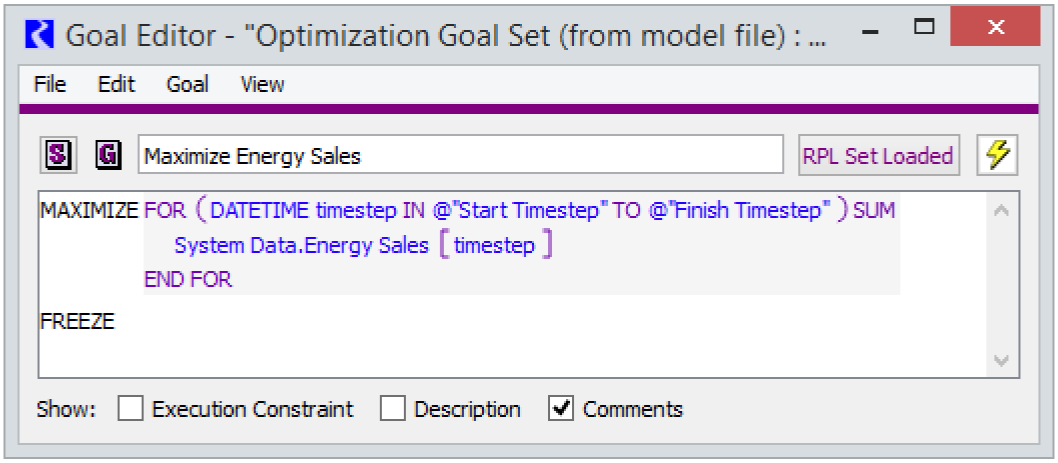
Energy Sales Definition

```
FOR ( DATETIME timestep IN @"Start Timestep" TO @"Finish Timestep" ) DO
    ADD CONSTRAINT System Data.Energy Sales [ timestep ] == FOR ( OBJECT res IN ListSubbasin ( "PowerReservoir" ) ) SUM * System Data.Energy Price [ timestep ]
                                                            res . "Energy" [ timestep ]
                                                          END FOR
END FOR
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

5. Use the variable in a policy constraint or objective.

The goal shown below maximizes the sum over the entire run of the Energy Sales variable defined in the previous step.


Goal Editor - "Optimization Goal Set (from model file) : ...
File
Edit
Goal
View
S
G
RPL Set Loaded

Maximize Energy Sales

```
MAXIMIZE FOR ( DATETIME timestep IN @"Start Timestep" TO @"Finish Timestep" ) SUM
             System Data.Energy Sales [ timestep ]
         END FOR

FREEZE
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

User-defined variables provide much flexibility in extending the existing RiverWare methods for Optimization. The one restriction on user-defined variables is that, as with all Optimization constraints, the defining constraints must be linear expressions of other constraints.

4.3 Linearization, Approximation and Replacement

Some slots, such as Storage and Outflow, correspond directly to decision variables in the Optimization problem. Other slots, such as Power, are translated into linear combinations of other decision variables. RiverWare automatically selects which substitution methods to use based on the slot(s) referenced and the context in which they are referenced in the policy (goal). All substitution is done automatically behind the scenes in RiverWare. For all substitution methods, if the resulting expression is not a function of only decision variables, substitution continues using whatever method is appropriate until only decision variables remain.

Details of the substitution methods are given below.

4.3.1 Link Substitution

Link substitution replaces a linearized slot with the slot(s) that it is linked to. An example is the substitution of a reach Inflow that can be a combination of Outflows from one or more other objects.

4.3.2 Constraint Substitution

Constraint substitution replaces expressions containing one linearized slot by equivalent expressions containing only one decision variable slot. For example, Pool Elevation is a linearized slot that has a direct relationship to Storage, a decision variable slot, defined by the Elevation Volume Table. A constraint such as Pool Elevation greater than or equal to some value can be replaced by an equivalent constraint on Storage.

Direct substitution will not work for constraints with multiple terms. For example, Operating Head cannot be replaced by a constant directly because Operating Head is Pool Elevation minus Tailwater Elevation, and thus there are two linearized slots on the left hand side of the constraint. When it is available, direct substitution is the preferred method for linearization because the only linearization error is the error in the table.

4.3.3 Expression Substitution

Expression substitution replaces a linearized slot with a linear expression of other slots. An example for Precipitation Volume is shown below.

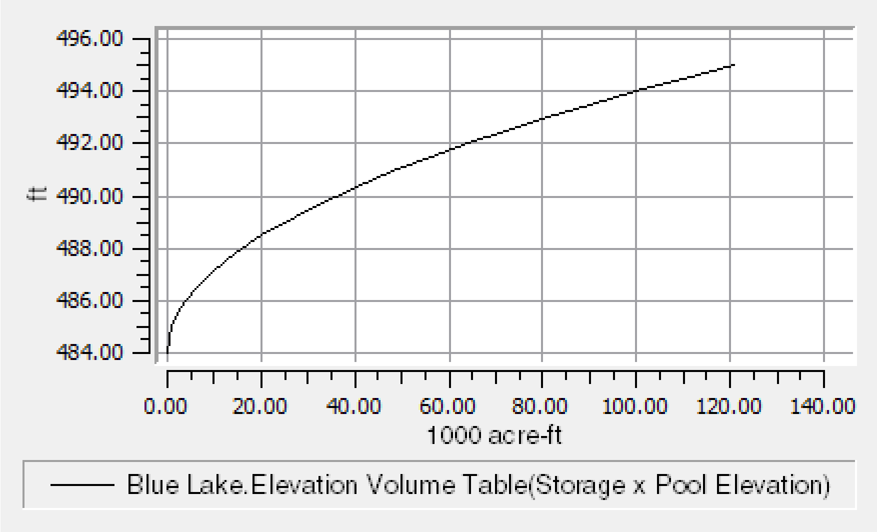
$$\text{Precipitation Volume}_t = \text{TimestepLength} \times \text{Precipitation Rate}_t \times \frac{\text{Surface Area}_t + \text{Surface Area}_{t-1}}{2}$$

In the above expression, Surface Area is itself a linearized variable that is a function of Storage. So Surface Area will be substituted by an expression in terms of Storage.

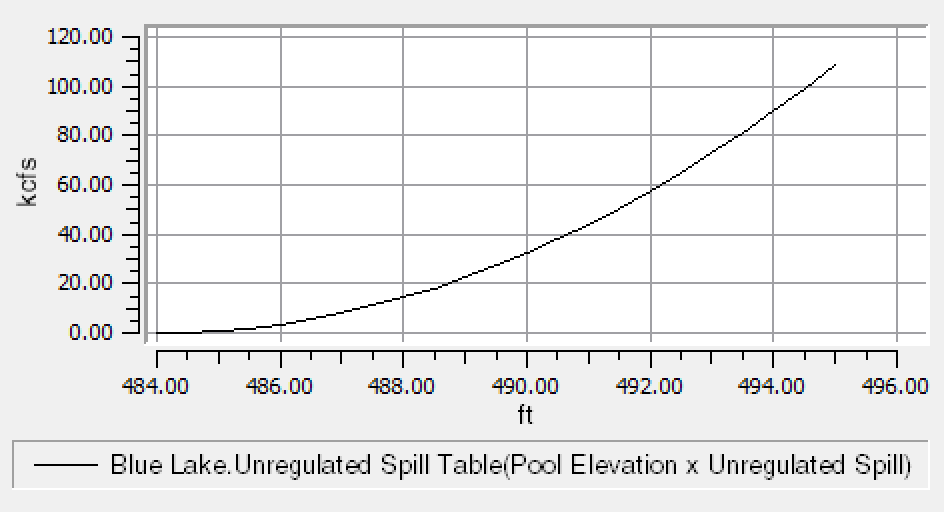
4.3.4 Variable Substitution

Variable substitution replaces a linearized slot with a known linear expression of other slots. Variable substitution can use direct substitution, a two dimensional linearization (tangent, two-point line or piecewise) or three-dimensional linearization (constant and multiple approximation). These methods are described below. RiverWare automatically chooses the correct linearization method to use based on the slot and the context of the policy expression.

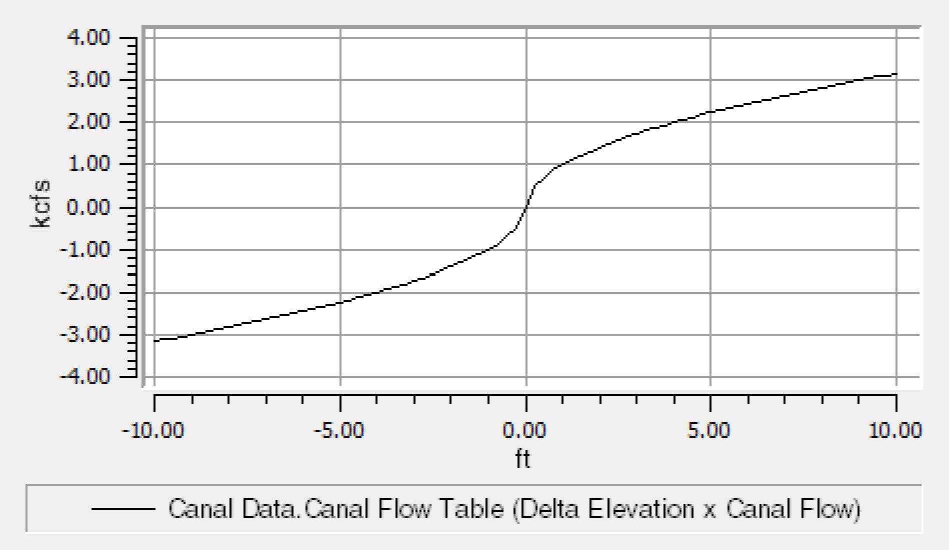
The mathematical functions used by RiverWare to model the relationships between two slots can be classified as convex, concave or nonconvex, nonconcave. Which linearization methods can be used to approximate a functional relationship between two slots and the accuracy of the approximation depends on the convexity of the function. Riverware expects a specific convexity for each functional relationship. If a user input function defined by a table slot does not match the expected convexity, an error is returned. An example of a concave function is the relationship between Pool Elevation and Storage.



A convex function is the relationship between Unregulated Spill and Pool Elevation.



A nonconvex, nonconcave function is the relationship between Canal Flow and Delta Elevation.

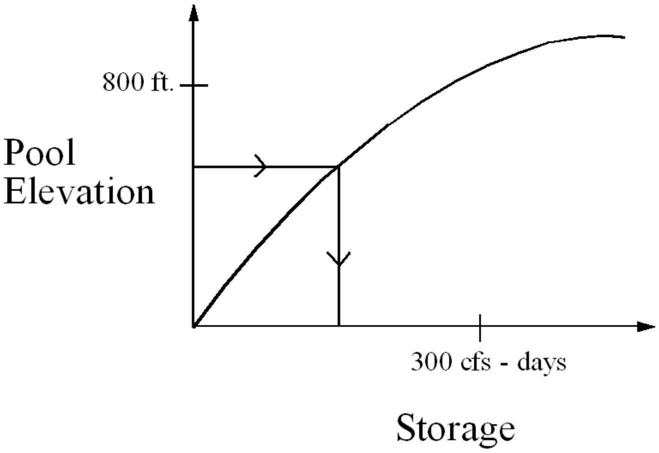


A tangent approximation will over-approximate a concave function and under-approximate a convex function. A two-point (secant) approximation will under-approximate a concave function and over-estimate a convex function.

The points used to define the tangent, two-point line and piecewise approximations are specified in linearization parameter tables. These are slots with LP Param in the name. Details about individual LP Param table slots are given within individual slot and method descriptions in the Objects and Methods section **HERE (Section 5)**.

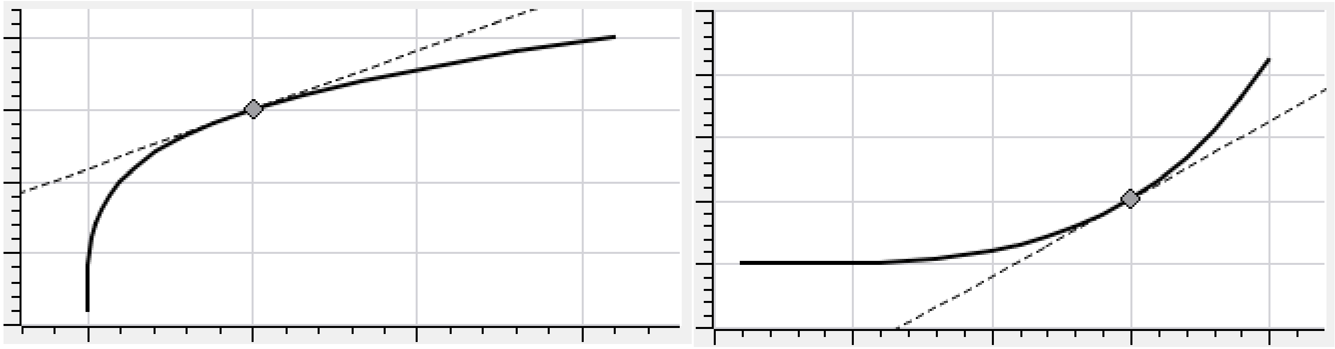
4.3.4.1 Direct Substitution

Substitution replaces the a variable with the value of another variable from the linearization function table. For example, Pool Elevation is replaced by the equivalent Storage. Values that are not listed directly in the table are approximated by linear interpolation. Substitution is only allowed for single term constraints. Substitution is generally the preferred linearization method when it is available because it has the least approximation error. RiverWare automatically uses this approach whenever there is a constraint with a single term.



4.3.4.2 Tangent Approximation

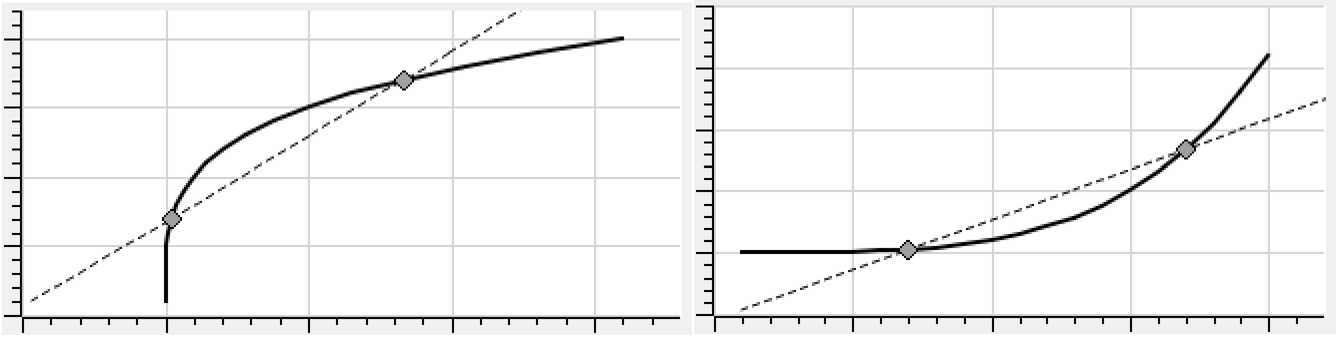
For the tangent approximation a nonlinear function is approximated as the line tangent to the function at a specified point. The same approximation point is used for every time step. Tangent is preferred when an over estimation is acceptable for a concave function or an under approximation is acceptable for a convex function. Tangent approximations are good when the value of a function will not vary greatly from the initial value (approximation point). Tangent approximations are not always available because of the errors they can introduce for certain types of functions. For example, tangent approximations are not used for Spill Capacities because it could result in a non-zero value for a function with a zero input. The best approximation point for a tangent to be taken at is the median value expected during the run.



Tangent Approximations of a Nonlinear Function

4.3.4.3 Two-point Line Approximation

For the two-point line (secant) approximation a nonlinear function is approximated by a line through two points on the function. The two points are fixed and remain the same for every time step. Unlike the tangent approximation, the two point line approximation sometimes underestimates a function and sometimes overestimates a function. The best approximation points for a line are points that will define a line close to the expected values during the run.



Two-Point Line (Secant) Approximation

4.3.4.4 Piecewise Approximation

For a piecewise approximation a nonlinear function is approximated as a piecewise linear function. The x values for an arbitrary number of points are input into the third column of the LP Param table. Lines are generated between adjacent points. Several constraints are added to the LP model that make sure that the approximation will work.

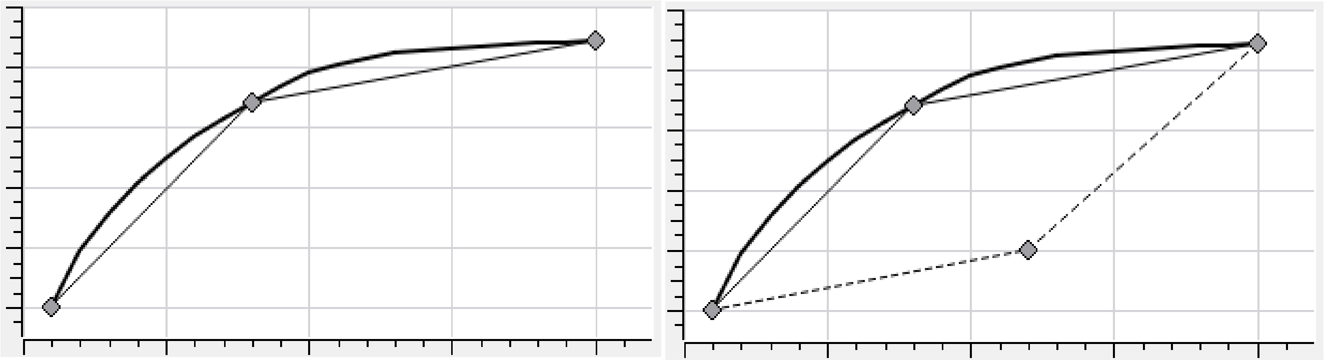
As an example, Power (P) is a function of Turbine Release (Q). A piecewise approximation would be

$$P = aQ_1 + bQ_2$$

with the constraints

$$Q = Q_1 + Q_2$$
$$0 \le Q_i \le u_i$$

Where a and b are the slopes of the approximation lines and u_i is the length of segment i. The optimization process can "cheat" when dealing with less than or equal constraints on concave functions and greater than or equal to constraints on convex functions. The figure below shows a piecewise approximation of a concave function using 2 lines (3 points). If a minimization is done to this function or it is used with a less than or equal to constraint, the solution would try to pick values that correspond to the dashed line rather than the solid line. That is, it would use the pieces out of order. RiverWare prevents this use of pieces out of order by automatically using the two-line approximation in these cases. If substitution is not available, piecewise approximations are the next preferred linearization method because they can fit the curve closely and minimize the approximation error better than line or tangent generally can. With piecewise approximations it is best to cover the range of values of the function, especially in range of the expected values. Additional points should be included where there are breaks in the curve. A minimum of two points are required. As many points as desired may be used to fit the curve, but a larger number of points will increase the size of the optimization problem and could impact performance (run time).



Piecewise Approximation of a Concave Function

4.3.4.5 Lambda Method

This section is under development.

5. Objects and Methods

Following is a description of slots, methods, and constraints that are used by each object in optimization. Presented are

- General Slots: those that exist on an object when the optimization is selected,
- User Methods: used to specify how a certain feature is to be represented
- Numerical Approximations: how certain variables are replaced or approximated.
- Modeling the object: items to consider when building a model of each object.

5.1 Canal

The canal object models gravity flow through a canal connected to two reservoirs.

5.1.1 General Slots

General slots are always present on the object, regardless of selected methods. The following slots are provided on the Canal object when the Optimization Controller is selected. Additional information on general simulation slots, including these slots, can be found **HERE (Objects.pdf, Section 6)**.

☞ **ELEVATION 1**

Type: Agg Series Slot
UNITS: LENGTH
Description: Pool Elevation of reservoir 1
Information:
Defined by: Replacement by the linked reservoir Pool Elevation slot (which in turn may be Numerically Approximated).

☞ **ELEVATION 2**

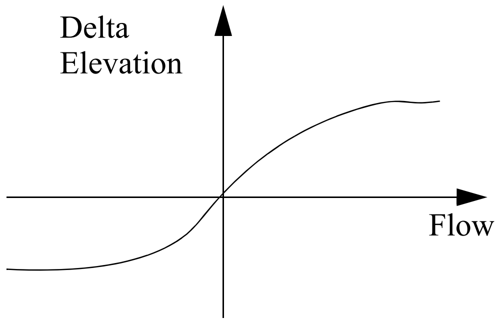
Type: Agg Series Slot
UNITS: LENGTH
Description: Pool Elevation of reservoir 2
Information:
Defined by: Replacement by the linked reservoir Pool Elevation slot (which in turn may be Numerically Approximated).

☞ FLOW 1

Type: Agg Series Slot
UNITS: FLOW
Description: flow to or from the canal end connected to reservoir 1
Information: This slot must be linked to the Canal Flow slot on reservoir 1.
Defined by: Explicit Optimization variable as 0.0 = Flow 1 + Flow 2

☞ FLOW 2

Type: Agg Series Slot
UNITS: FLOW
Description: flow to or from the canal end connected to reservoir 2
Information: This slot must be linked to the Canal Flow slot on reservoir 2.
Defined by: Numerical 3-D Approximation in terms of Lower Elevation and Delta Elevation. Approximation is based on the Canal Flow Table. The Flow 2 LP Param table contains a value for Lower Elevation used to index the Lower Elevation column of the Canal Flow Table. This approximated value, therefore, reduces Flow 2 to a function of Delta Elevation. The length values in the Flow 2 LP Param table are then used as approximation points indexing the Delta Elevation column of the Canal Flow Table. The Canal Flow Table should have increasing values of Lower Elevation and Delta Elevation. The function is expected to be non-convex (see Figure). The preferred order of approximation is substitution, piece-wise, two-point line, tangent.



Non Concave, Non Convex Function

☞ CANAL FLOW TABLE

Type: Table Slot
UNITS: LENGTH, LENGTH, FLOW
Description: 3-D table used to find canal flow by interpolation given a Lower Elevation, Delta Elevation between the lower and higher pool elevations and resulting Flow 2.
Information:
Defined by: user-input

☞ DELTA ELEVATION

Type: Agg Series Slot
UNITS: LENGTH
Description: The average elevation difference between the two reservoirs and across the canal at the current timestep
Information:
Defined by:

☞ **FLOW 2 LP PARAM**

Type:	Table Slot
UNITS:	LENGTH, LENGTH, LENGTH, LENGTH
Description:	specifies the Lower Elevation and the Delta Elevation points used to take the tangent, line, and piecewise approximations for numerical approximation of Flow 2 as specified in the Canal Flow Table. The best Lower Elevation to choose should be close to the expected elevation for the whole run. For Delta Elevation the suggested points for tangent approximation is the estimated value for the whole run and for line approximation are 0 flow and some likely value. piecewise linearization is not used.
Information:	
Defined by:	user-input

☞ **LOWER ELEVATION**

Type:	Series Slot
UNITS:	LENGTH
Description:	The elevation of the lower of the two reservoirs for the current timestep
Information:	
Defined by:	

On the Canal object, there are currently no categories available in RPL Optimization. What exists as the default arrangement, however, reflects the Canal Flow Table simulation method of the CanalFlowCalculationCategory **HERE (Objects.pdf, Section 6.1.1.2)**.

5.1.2 Canal Flow Numerical Approximation

Any time the Canal object is included in the Optimization problem, then Flow 2 is added to the problem, first in terms of a constraint and then as a function of Lower Elevation and Delta Elevation (Numerical 3-D Approximation). The relationship between Lower Elevation, Delta Elevation and Flow 2 comes from the user-input Canal Flow Table. The table will be queried using the points defined in the Flow 2 LP Param table.

5.1.3 Modeling a Canal

Steps to follow in setting up a canal for optimization:

6. Set up a running simulation model with appropriate links to and from the canal. Use the Canal Flow Table method.

7. When RPL Optimization control is selected, fill in the Flow 2 LP Param table.

5.2 Confluence

The Confluence object has no selectable methods in RPL Optimization. A constraint is included in the optimization problem to ensure mass balance for all timesteps:

$$\text{Inflow1} + \text{Inflow 2} = \text{Outflow} \quad \text{(EQ 1)}$$

Please refer to the Confluence documentation **HERE (Objects.pdf, Section 8)** for a full description of slots.

5.3 Data Objects

Data Objects hold slots that can be used in your policy either as reference information or as user-defined optimization variables.

Reference data is just values that are known before the optimization run like minimum flows, guide curves, power targets, etc. You can write policy that uses the values in these slots.

User defined optimization variable are slots that you wish to compute in optimization. They must be a series slot or a column of an agg series slot on a data object. To configure that a series slot on a data object is a user defined optimization variable, on the configuration dialog, check the **Is User Defined Variable** check box. Then, this slot can participate in an optimization problem, but you must still add policy that defines the equation for this variable to pull it into the problem.

The automatically generated post-opt RBS ruleset includes rules to set the user defined variable slots to their optimal values.

5.4 Inline Power Plant

The Inline Power Plant models a run-of-the-river power plant where power facilities rely on the natural slope of the river channel to provide necessary head to generate power. Additionally, this object provides very little or no storage. Please refer to the Inline Power Plant documentation **HERE (Objects.pdf, Section 15)**.

The remainder of this section is under development.

5.5 Level Power Reservoir

The Level Power Reservoir is similar to a Storage Reservoir but with added functionality to model power production facilities on the reservoir. The assumption that the water surface is level is inherent in this object. Additional information can be found **HERE (Objects.pdf, Section 17)**.

5.5.1 General Slots

General slots are always present on the object, regardless of selected methods. The following slots are provided on the reservoir when the optimization controller is selected.

☞ **CANAL FLOW**

Type:	Agg Series
UNITS:	FLOW
Description:	Flow into (out of) the reservoir from (to) a canal
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint

☞ **DIVERSION**

Type:	Series Slot
UNITS:	FLOW
Description:	Flow from the reservoir to a diverting object
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint

☞ **ELEVATION VOLUME TABLE**

Type:	Table
UNITS:	LENGTH VS VOLUME
Description:	Table relating elevation of the reservoir to volume stored in the reservoir
Information:	
I/O:	Input only
Defined by:	Input only

☞ **ENERGY**

Type:	Agg Series Slot
UNITS:	ENERGY
Description:	Product of the power generated by flow through the turbines and the length of the timestep.
Information:	
Defined by:	Replaced by Power * Timestep Length

☞ **FLOW FROM PUMPED STORAGE**

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow into the reservoir from a pumped storage reservoir
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint. This slot should be linked to Outflow on a Pumped Storage object. The Pumped Storage object constrains its Outflow.

Flow TO Pumped Storage

Type:	Agg Series Slot
Units:	Flow
Description:	Flow out of the reservoir into a pumped storage reservoir
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint. This slot should be linked to Pumped Flow on a Pumped Storage object.

Inflow

Type:	MultiSlot
Units:	Flow
Description:	Inflow into the reservoir from upstream
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint

Max Turbine Q

Type:	Table Slot
Units:	Length vs Flow
Description:	Table relating Operating Head to maximum Turbine Capacity
Information:	See power methods for more information

Operating Head

Type:	Agg Series Slot
Units:	Length
Description:	Elevation difference between the average Pool Elevation and the average Tailwater Elevation during a timestep
Information:	
Defined by:	Replacement by (Pool Elevation(t) + Pool Elevation(t-1)) / 2 - Tailwater Elevation

Outflow

Type:	Agg Series Slot
Units:	Flow
Description:	Outflow from the reservoir
Information:	
Defined by:	Explicit Optimization variable as Outflow = Turbine Release + Spill

Plant Power Table

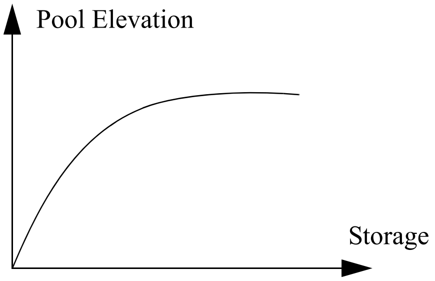
Type: Table
Units: Length vs Flow vs Power
Description: 3D Table relating Operating Head, Turbine Release, and Power
Information: See power methods for more information

Pool Elevation

Type: Series Slot
Units: Length
Description: Elevation of the water surface of the Reservoir
Information: When Pool Elevation is a part of the optimization problem, as it is in all conceivable RiverWare Optimization applications, this slot is numerically approximated as a function of Storage (Numerical 2-D Approximation). The relationship between Pool Elevation and Storage will come from the user-input Elevation Volume Table. The table will be queried either using user-input points defined in the Pool Elevation LP Param table.
Defined by: Numerical 2-D Approximation in terms of Storage, based upon the Elevation Volume Table. The Pool Elevation LP Param table values are used as approximation points indexing the Elevation Volume Table. The Elevation Volume Table should have increasing values of Pool Elevation and Storage. Storage is required to be a concave function of Pool Elevation. The preferred order of approximation is substitution, piece-wise, tangent, two-point line.

Pool Elevation LP Param

Type: Table Slot
Units: Volume
Description: Specifies the Storage points used to take the tangent, line and piecewise approximations for Pool Elevation linearization
Information: This table is used for linearization unless Pool Elevation Linearization Automation category has selected Plant Automation. The best Storage point to choose for tangent approximation would be the expected storage expected during the run; for the line approximation, the expected maximum and minimum Storage; for piecewise approximation, use points that cover the full range of expected Storage during the run with intermediate points such that a piecewise linear curve reasonably approximates the actual curve.
Defined by: User-input



Pool elevation storage relationship. The figure is not drawn to scale.

☞ POWER

Type:	Series Slot
UNITS:	POWER
Description:	Power generated by flow through the turbines
Information:	
Defined by:	Numerical 3-D Approximation in terms of Operating Head and Turbine Release. Approximation is based on the Plant Power Table. The Power LP Param table contains a value for Operating Head used to index the Operating Head column of the Plant Power Table. This approximated value, therefore, reduces the Power to a function of Turbine Release at the given Operating Head. The flow values in the Power LP Param table are then used as approximation points indexing the Turbine Release column of the Plant Power Table. The Plant Power Table should have increasing values of Operating Head and Turbine Release. Power should be a concave function of Operating Head, but concavity is not strictly enforced; mild non-concave regions are permissible to allow for round-off error, etc. The preferred order of approximation is substitution, piece-wise, two-point line, tangent.

☞ RETURN FLOW

Type:	MultiSlot
UNITS:	FLOW
Description:	Flow returning from a diversion object
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint (see Storage)

☞ SPILL

Type:	Series Slot
UNITS:	FLOW
Description:	Sum of the Bypass, Regulated Spill and Unregulated Spill
Information:	
Defined by:	Explicit Optimization variable as Spill = Bypass + Regulated Spill + Unregulated Spill

STORAGE
Type: Series Slot
UNITS: VOLUME
Description: Volume of water stored in the reservoir
Information:
Defined by: Explicit Optimization variable as Storage = Storage(t-1) + Precipitation Volume - Evaporation - Change in Bank Storage + timestep * (Inflow + Canal Flow + Flow TO Pumped Storage + Hydrologic Inflow Net + Return Flow - (Outflow + Diversion + Flow FROM Pumped Storage))

TAILWATER BASE VALUE
Type: Series Slot
UNITS: LENGTH
Description: Elevation of tailwater or base elevation used to compute elevation of tailwater
Information:
Defined by: Explicit Optimization variable should be input or linked. Related constraints can be found **HERE (Section 5.5.2.20)** Tailwater Elevation Numerical Approximation discussion, or on other objects to which the slot is linked.

TAILWATER ELEVATION
Type: Series Slot
UNITS: LENGTH
Description: Water surface elevation on the downstream side of the dam
Information:
Defined by: Various approaches dependent on the method selected in the Optimization Tailwater category.

TURBINE CAPACITY LP PARAM
Type: Table
UNITS: LENGTH, LENGTH, LENGTH
Description: LP Param table for turbine capacity
Information: see power methods for more information

TURBINE CAPACITY
Type: Series Slot
UNITS: FLOW
Description: Flow capacity of the entire power plant’s turbine(s)
Information:
Defined by: Numerical 2-D Approximation in terms of Operating Head, based upon a maximum turbine capacity table. This capacity table is determined in various ways according to the Power method.

☞ **TURBINE RELEASE**

Type: Agg Series Slot
UNITS: FLOW
Description: Flow through the turbines of a power reservoir
Information:
Defined by: Explicit Optimization variable as Turbine Release <= Power Plant Cap Fraction * Turbine Capacity

5.5.2 User Methods in Optimization

The following categories and methods are available for use in Optimization. Because of dependency relationships, you may not be able to see them in your model until you change other methods. In the discussion below, you will see the other methods that need to be selected to enable a given method to be used in your model. When building a model, you will wish to review this to ensure that required dependencies are satisfied so that the desired methods are available for use.

5.5.2.1 Bank Storage

Not all methods are functional in RPL optimization. Of the available methods, “None” and “CRSS Bank Storage” are supported.

5.5.2.1.1 None

Click **HERE (Objects.pdf, Section 17.1.25.1)** for more information.

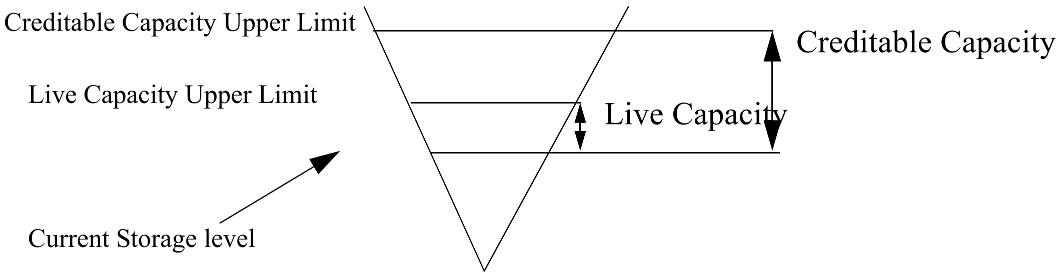
Input Bank Storage

5.5.2.1.2 CRSS Bank Storage

Click **HERE (Objects.pdf, Section 17.1.25.3)** for more information.

5.5.2.2 Creditable Capacity Available

The creditable capacity category conceptually represents the total amount of available storage space above the current storage in the reservoir. As the pool storage in the reservoir increases, the creditable capacity decreases.



5.5.2.2.1 None

If this method is selected creditable capacity is not calculated for the reservoir.

5.5.2.2.2 Constraint and Variable

If this method is selected an additional decision variable and physical constraint are added to the optimization problem:

SLOTS SPECIFIC TO THIS METHOD:

☞ **CREDITABLE CAPACITY**

Type:	Gassers
UNITS:	VOLUME
Description:	The creditable capacity is the total amount of storage space available above the current storage
Information:	
Defined by:	Explicit Optimization Variable included in the Optimization Problem as part of the constraint

$$\text{Max Storage} >= \text{Storage} + \text{Creditable Capacity.} \quad \text{(EQ 2)}$$

Max Storage for the reservoir is entered in the upper bound field of the Storage slot configuration dialog box.

5.5.2.3 Cumul Stor Val Linearization Automation

Appearance of this category is dependent on selecting the Opt Cumulative Storage Value Table method for the Optimization Future Value category (which in turn is dependent on selecting the Cumulative Storage Value Table method for the Future Value category).

This category allows the optimization to automate the creation of the Cumulative Storage Value Table, and the selection of linearization points and linearization method for the Cumulative Storage Value slot. Actual usage of these values occurs in the **HERE (Section 5.5.2.9)** Optimization Future Value category's Opt Cumulative Storage Value Table method.

5.5.2.3.1 None

If this method is selected, no automation will be performed.

5.5.2.3.2 Marginal Value to Table and Lin

This method uses information from the simulation slot Marginal Storage Value Table to generate the Cumul Stor Val Table, select linearization points, and choose a linearization method. The cumulative storage value in the Cumul Stor Val Table can be thought of as the summation of the marginal storage values from a storage of 0 to the current storage.

As an illustration of the automation procedure, consider the following Marginal Storage Value Table:

Marginal Storage Value Table:

Storage	Marginal Value
20	30
60	26
100	24

To parameterize the Cumul Stor Val Table, the automation proceeds as follows:

1) The first row receives a Storage value of 0.

Cumul Stor Val Table

Storage	Cumulative Value
0	

2) For each row i in the Marginal Storage Table, not including the last row, the average Storage (i.e. the midpoint) of row i and row i + 1 is assigned as Storage in each successive row of the Cumul Stor Val Table. For example, row 2 Storage equals the average of 20 and 60; row 3 Storage equals the average of 60 and 100.

Cumul Stor Val Table

Storage	Cumulative Value
0	
40	
80	

3) The last row of the Cumul Stor Val Table receive a Storage value equal to the maximum storage associated with the Storage slot's configuration.

Cumul Stor Val Table

Storage	Cumulative Value
0	
40	

Storage	Cumulative Value
80	
140	

4) Returning to the first row of the Cumul Storage Val Table, a Cumulative Value of 0 is assigned.

Cumul Stor Val Table

Storage	Cumulative Value
0	0
40	
80	
140	

5) For each successive row j = 2, 3, 4 in the Cumul Stor Val Table and corresponding row i = 1, 2, 3 in the Marginal Storage Value Table, the Cumulative Value equals the Storage from row j - 1 + (the change in Storage of row j from row j - 1) * the Marginal Value from row i. For example, row 2 Cumulative Value equals 1200 calculated from 0 + (40 - 0) * 30; row 3 Cumulative Value equals 2240 calculated from 1200 + (80 - 40) * 26; row 4 Cumulative Value equals 3680 calculated from 2240 + (140 - 80) * 24.

Cumul Stor Val Table:

Storage	Cumulative Value
0	0
40	1200
80	2240
140	3680

The Cumul Stor Val LP Param table is then built using Storage values from the Cumul Stor Val Table:

Cumul Stor Val LP Param

Tangent	Line	piecewise
	0	0
	140	40
		80
		140

SLOTS SPECIFIC TO THIS METHOD

There are no slots specific to this method as it uses the slots in the Opt Cumulative Storage Value Table method. However, for clarity in the discussion above, the slots are reshown here.

☞ **CUMUL STOR VAL TABLE**
Type: Table Slot
UNITS: VOLUME VS. VALUE
Description: The estimated total economic value of water stored in the reservoir for discrete storage values.
Information:
Defined by: Automated procedure described above.

☞ **CUMUL STOR VAL LP PARAM**
Type: Table Slot
UNITS: VOLUME, VOLUME, VOLUME
Description: Specifies the storage points used to take the tangent, line and piecewise approximations for Cumul Stor Val Table linearization
Information:
Defined by: Automated procedure described above.

☞ **MARGINAL STORAGE VALUE TABLE**
Type: Table Slot
UNITS: STORAGE VS. $ VALUE
Description: Anticipated Storage versus worth of Cumulative Storage per unit energy
Information: This table should be increasing in storage, and usually decreasing in marginal value
Defined by: Required input

5.5.2.4 Diversion from Reservoir

Not all methods in this category are supported in RPL optimization. Of the available methods, “None” and “Available Flow Based Diversion” are supported; selection of any other method will result in an error during begin run.

5.5.2.4.1 None

Click **HERE (Objects.pdf, Section 17.1.26.1)** for more information.

5.5.2.4.2 Available Flow Based Diversion

Click **HERE (Objects.pdf, Section 17.1.26.2)** for more information.

5.5.2.5 Energy in Storage

In Optimization, currently “None” and “EIS Table Lookup” are supported.

5.5.2.5.1 None

No Energy in storage is considered.

5.5.2.5.2 EIS Table Lookup

With this method selected, Energy in Storage is considered as a function of Pool Elevation. Click **HERE (Objects.pdf, Section 17.1.6.2)** for more information on the simulation method. If the optimization problem uses Energy In Storage, either directly or indirectly, then this slot is added to the problem. Before being passed to the optimization solver, this slot is numerically approximated as a function of Pool Elevation (Numerical 2-D Approximation). The relationship between Pool Elevation and Energy In Storage will come from the user-input Energy In Storage Table. The table will be queried either using user-input points defined in the Energy In Storage LP Param table or using automatically calculated points, depending on method selection in the Pool Elevation Linearization Automation category. If the selected method in the Pool Elevation Linearization Automation is “none”, then the user-input Energy In Storage LP Param table values are used. For other Pool Elevation Linearization Automation methods (Initial Target Input, Min Difference or Range Input, Min Difference) the same points automatically developed for the Pool Elevation linearization will be used.

RELATED SLOTS

☞ **ENERGY IN STORAGE**

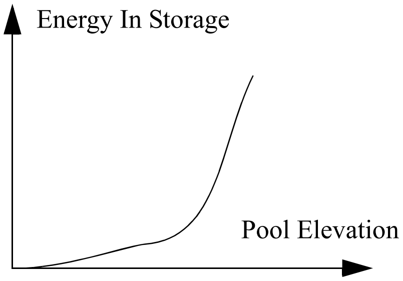
Type: Series Slot
UNITS: ENERGY VS. POWER
Description: Energy in Storage in the reservoir
Information:
Defined by: Numerical 2-D Approximation in terms of Pool Elevation, based upon the Energy In Storage Table. The Energy In Storage LP Param table values are used as approximation points indexing the Energy In Storage Table. The Energy In Storage Table should have increasing values of Pool Elevation and Energy In Storage. Energy In Storage is required to be a convex function of Pool Elevation. The preferred order of approximation is substitution, piece-wise, tangent, two-point line.

☞ **ENERGY IN STORAGE LP PARAM**

Type: Table Slot
UNITS: LENGTH
Description: Specifies the Pool Elevation points used to take the tangent, line and piecewise approximations for Energy In Storage linearization.
Information: This table is used for linearization unless the Pool Elevation Linearization Automation category has a method selected other than “none”.
Defined by: User-input

☞ **ENERGY IN STORAGE TABLE**

Type:	Table Slot
UNITS:	LENGTH VS. ENERGY
Description:	Table defining the relationship between Energy In Storage and Pool Elevation
Information:	
Defined by:	User-input



Energy in Storage pool elevation relationship. Not drawn to scale.

5.5.2.6 Optimization Evaporation

This category can be used to model evaporation and precipitation.

5.5.2.6.1 None

The Optimization Evaporation method “None” is the default method for this category. It does no calculations and requires that “None” be selected for the Evaporation and Precipitation category.

5.5.2.6.2 Opt Input Evaporation

This method is analogous to the Input Evaporation method in simulation and requires the Input Evaporation method to be selected for the Evaporation and Precipitation category. Evaporation Rate and Precipitation Rate are entered as a time series. Evaporation is calculated as a product of Evaporation Rate, Average Surface Area over the timestep and Timestep length. Similarly Precipitation Volume is calculated as the product of Precipitation Rate, Average Surface Area and Timestep length.

Note: The linearization of the Surface Area variable can result in a small approximation error in optimization for Evaporation and Precipitation Volume. This means there can be a small difference between the mass balances in the optimization solution and the post-optimization rulebased simulation when using this method. It is important to use care when setting the approximation points for Surface Area in order to reduce this approximation error. Refer to the information on the Surface Area LP Param slot below. Also caution should be used if applying this method at a monthly timestep. All rates in the optimization mass balance are converted to monthly volumes based on a 30-day month, regardless of the month. This will also produce a difference between the mass balances in the optimization solution and the post-optimization rulebased simulation for a *monthly* timestep.

SLOTS SPECIFIC TO THIS METHOD

ELEVATION AREA TABLE

Type: Table Slot
UNITS: LENGTH VS. AREA
Description: Represents the Elevation-Surface Area relationship
Information: This table must be input. It is used to derive the Volume Area Table.
Defined by: User-input

EVAPORATION

Type: Series Slot
UNITS: VOLUME
Description: The volume of water lost to evaporation over the timestep
Information: If this slot contains user input, it is added directly to the mass balance constraint, otherwise it is defined by the expression below.
Defined by: Either user-input or the following constraint:

$$\text{Evaporation} = \text{EvaporationRate} \times \frac{\text{SurfaceArea}(t-1) + \text{SurfaceArea}}{2} \times \text{Timestep}$$

EVAPORATION RATE

Type: Series Slot
UNITS: VELOCITY
Description: The rate, in length per time, at which water is lost to evaporation at each timestep
Information: This slot can be set as user input. If it is not set as an input, and if Evaporation is not an input, this slot defaults to zero. If Evaporation is an input, this slot is not used.
Defined by: User-input or defaults to zero

PRECIPITATION RATE

Type: Series Slot
UNITS: VELOCITY
Description: The rate, in length per time, at which water is gained from precipitation at each timestep
Information: This slot can be set as user input. If it is not set as an input, it defaults to zero.
Defined by: User-input or defaults to zero

PRECIPITATION VOLUME

Type: Series Slot
UNITS: VOLUME
Description: The volume of water gained from precipitation over the timestep
Information: The Input Evaporation method will not allow this slot to be set as an input.
Defined by: Explicit constraint:

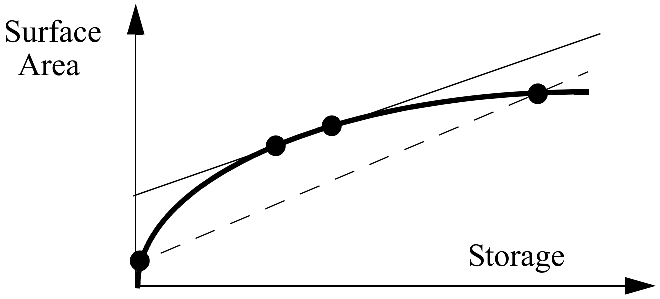
$$\text{Precipitation Volume} = \text{PrecipitationRate} \times \frac{\text{SurfaceArea}(t-1) + \text{SurfaceArea}}{2} \times \text{Timestep}$$

☞ SURFACE AREA

Type:	Series Slot
UNITS:	AREA
Description:	The area of the water surface at the end of the timestep
Information:	This slot is numerically approximated as a function of Storage (Numerical 2-D Approximation). The relationship between Pool Elevation and Storage comes from the automatically generated Volume Area Table. The table is queried using the user-input points defined in the Surface Area LP Param table.
Defined by:	Numerical 2-D Approximation in terms of Storage, based upon the Volume Area Table. The Surface Area LP Param table values are used as approximation points indexing the Volume Area Table. The preferred order of approximation is substitution, piecewise, two-point line, tangent. Most often the two-point line (secant) approximation will be used.

☞ SURFACE AREA LP PARAM

Type:	Table Slot
UNITS:	VOLUME
Description:	Specifies the Storage points used to take the tangent, line and piecewise approximations for Surface Area linearization
Information:	The best Storage point to choose for tangent approximation would be the expected storage during the run; for line approximation, the expected maximum and minimum Storage; for piecewise approximation, use points that cover the full range of expected Storage during the run with intermediate points such that a piecewise linear curve reasonably approximates the actual curve.In most cases, the line (secant) approximation will be used. It is important to set these points carefully to minimize approximation error.
Defined by:	User-input



Surface Area vs. Storage relationship with two alternative line approximations.
The figure is not drawn to scale

The figure above represents two alternative selections of points for the line approximation in the Surface Area LP Param table. The linear approximation represented by the dashed line corresponds to the selection of points near the extremes of the Volume Area table. This approximation will tend to result in an under-estimation of Surface Area, and thus an under-estimation of Evaporation and Precipitation Volume. Evaporation losses in the post-optimization rulebased simulation would be greater than the losses approximated in the optimization solution. The linear approximation represented by the solid line corresponds to the selection of points closer together in the Volume Area Table, and is more similar to the Tangent approximation. This approximation would tend to result in an over estimation of Surface Area, and the losses due to Evaporation in the post-optimization rulebased simulation would be less than the approximation in the optimization solution.

☞ VOLUME AREA TABLE

Type:	Table Slot
UNITS:	VOLUME VS. AREA
Description:	Represents the Storage Volume-Surface Area relationship
Information:	This table is read-only and is automatically generated at the start of the run from the Elevation Area Table and the Elevation Volume table. The method starts by copying the Elevation Area Table and then replaces the Pool Elevation Values with the corresponding Storage values. The Storage values are linearly interpolated based on the Elevation Volume Table
Defined by:	Automatically generated

5.5.2.7 Evaporation Linearization Automation Category

This category allows the approximation points for surface area to be automatically generated. This category requires a method other than “None” to be selected for the Optimization Evaporation category.

5.5.2.7.1 None

This is the default method for this category. There is no approximation point automation. The user will be required to enter approximation points in the Surface Area LP Param slot manually.

5.5.2.7.2 Use Elevation Approximation Points

This method automates the approximation points in the Surface Area LP Param table slot. It copies the same storage values used in the Pool Elevation LP Param slot. If this method is selected it will overwrite any values that have been entered manually in the Surface Area LP Param slot. This method may provide a good starting point for establishing the Surface Area approximation points; however in some cases, it may be important for the user to adjust these points manually (see details **HERE (Section 5.5.2.6.2)** under Surface Area LP Param).

5.5.2.8 Future Value

This category and its methods are not dependent on other method selections.

5.5.2.8.1 None

Click **HERE (Objects.pdf, Section 17.1.12.1)** for more information.

5.5.2.8.2 Cumulative Storage Value Table

In RPL-Optimization, the Cumulative Storage Value is Numerically Approximated as described below.

Click **HERE (Objects.pdf, Section 17.1.12.2)** for more information.

Slots specific to this Method

☞ Cumulative Storage Value

Type: Agg Series Slot
Units: $
Description: Represents the future energy value of the current storage
Information:
Defined by: 2-D approximation in terms of Anticipated Storage, based upon the Cumul Stor Val Table. The Cumul Stor Val LP Param table values (Storage) are used as approximation points indexing the Cumul Stor Val Table. The Cumul Stor Val Table should have increasing values of Storage and Cumulative Value. Cumulative Storage Value is required to be a concave function of Anticipated Storage. The preferred order of approximation is substitution, piece-wise, tangent, two-point line. The Cumul Stor Val Linearization Automation category's Marginal Value to Table and Lin method can automate creation of the Cumul Stor Val LP Param table and the Cumul Stor Val Table.

☞ Anticipated Storage

Type: Agg Series Slot
Units: Volume
Description: The combination of the actual storage plus water that would be expected to enter the reservoir after the Current Timestep but has not yet, due to lagging.
Information:
Defined by:

☞ Cumul Stor Val Table

Type: Table Slot
Units: Volume vs. Value
Description: The estimated total economic value of water stored in the reservoir for discrete storage values.
Information:
Defined by: User-input or by automated procedure if Cumul Stor Val Linearization Automation category has selected the Marginal Value to Table and Lin method.

☞ MARGINAL STORAGE VALUE TABLE

Type: Table Slot
UNITS: STORAGE VS. $ VALUE
Description: Anticipated Storage versus Cumulative Storage Value per unit energy
Information: This table should be increasing in storage, and logically decreasing in marginal value
Defined by: Required input

☞ SPILL COST

Type: Agg Series Slot
UNITS: $
Description:
Information:
Defined by:

5.5.2.9 Optimization Future Value

This category allows the optimization to provide slots relating to the future value of water. Its appearance is dependent on selecting the Cumulative Storage Value Table method for the Future Value category.

5.5.2.9.1 None

If this method is selected, the future value slots will not be visible, and no linearization will be attempted.

5.5.2.9.2 Opt Cumulative Storage Value Table

If this method is selected, the following slots will be visible, and linearization will be allowed.

SLOTS SPECIFIC TO THIS METHOD

☞ CUMUL STOR VAL LP PARAM

Type: Table Slot
UNITS: VOLUME, VOLUME, VOLUME
Description: Specifies the storage points used to take the tangent, line and piecewise approximations for Cumul Stor Val Table linearization
Information:
Defined by: User-input or by automated procedure if Cumul Stor Val Linearization Automation category has selected the Marginal Value to Table and Lin method.

5.5.2.10 Hydrologic Inflow

If any method in this category is selected, hydrologic inflow is included in the reservoir mass balance. Optimization assumes hydrologic inflow to be known (data) and it is not solved for by the reservoir regardless of the method selected. Click **HERE (Objects.pdf, Section 17.1.15)** for more information.

5.5.2.11 Live Capacity Available

The live capacity category conceptually represents the amount of available storage space between the current storage in the reservoir and a user defined maximum. As the pool storage in the reservoir increases, the live capacity decreases.

5.5.2.11.1 None

If this method is selected live capacity is not calculated for the reservoir.

5.5.2.11.2 Constraint and Variable

If this method is selected an additional decision variable and physical constraint are added to the optimization problem. See the figure in the Creditable Capacity Category.

☞ **LIVE CAPACITY**

Type:	Agg Series Slot
UNITS:	VOLUME
Description:	The amount of storage space available between the current storage and a user defined upper limit. This upper limit is entered in the upper bound field of the Live Capacity slot configuration dialogue box.
Defined by:	Explicit constraint

$$\text{Max Live Capacity} >= \text{Storage} + \text{Live Capacity} \quad \text{(EQ 3)}$$

5.5.2.12 Optimization Spill

The Optimization Spill methods determine how spill is calculated for the reservoir and generates physical constraints that correspond to the selected methods.

This category is dependent on selection of the Independent Linearizations method for the Optimization Power category. However, the method selected in the Optimization Spill category must match with the corresponding non-optimization method in the Spill category.

Spill is an Optimization decision variable. The following constraint is always generated for a reservoir:

$$\text{Outflow} = \text{Turbine release (or Release)} + \text{Spill} \quad \text{(EQ 4)}$$

The spill methods generate values applicable to an additional constraint:

$$\text{Spill} = \text{Regulated Spill} + \text{Unregulated Spill} + \text{Bypass}, \quad \text{(EQ 5)}$$

where some of these terms may be omitted if they do not apply to the selected spill method.

Depending on the method selected, some of the following slots will be added. Other slots will be used from the non-optimization method selected. Please click **HERE (Objects.pdf, Section 17.1.8)** for details on non-optimization Spill methods.

As applicable, the following constraints are also added:

Bypass <= Bypass Capacity **(EQ 6)**

Regulated Spill <= Regulated Spill Capacity **(EQ 7)**

Unregulated Spill = Unregulated Spill Capacity **(EQ 8)**

Bypass Capacity

Type: Series Slot
Units: Flow
Description: Bypass capacity
Information:
Defined by: Numerical 2-D Approximation in terms of storage, based upon the Bypass Capacity Table.

Bypass Capacity Table

Type: Table Slot
Units: Volume vs. Flow
Description: Storage vs. corresponding maximum bypass spill values
Information:
Defined by: Internally developed based on Bypass Table and Elevation Volume Table relationships. For each pool elevation in the Bypass Table, the Bypass Capacity Table has a row relating Storage to Bypass Capacity. The Pool Elevations are converted to Storage using the Elevation Volume Table.

Regulated Spill Capacity

Type: Series Slot
Units: Flow
Description: Regulated spill capacity
Information:
Defined by: Numerical 2-D Approximation in terms of storage, based upon the Regulated Spill Capacity Table.

Regulated Spill Capacity Table

Type: Table Slot
Units: Volume vs. Flow
Description: Storage vs. corresponding maximum regulated spill values
Information:
Defined by: Internally developed based on Regulated Spill Table and Elevation Volume Table relationships. For each pool elevation in the Regulated Spill Table, the Regulated Spill Capacity Table has a row relating Storage to Regulated Spill Capacity. The Pool Elevations are converted to Storage using the Elevation Volume Table.

☞ **REGULATED SPILL OR BYPASS LP PARAM**

Type:	Table Slot
UNITS:	VOLUME
Description:	Specifies the Storage points use to take the tangent, line and piecewise approximations for Regulated Spill Capacity linearization and Bypass Spill Capacity linearization.
Information:	
Defined by:	User-input

☞ **UNREGULATED SPILL LINEARIZATION TABLE**

Type:	Table Slot
UNITS:	VOLUME VS. FLOW
Description:	Storage vs. corresponding unregulated spill values
Information:	
Defined by:	Internally developed based on Unregulated Spill Table and Elevation Volume Table relationships. For each pool elevation in the Unregulated Spill Table, the Unregulated Spill Capacity Table has a row relating Storage to Unregulated Spill Capacity. The Pool Elevations are converted to Storage using the Elevation Volume Table.

☞ **UNREGULATED SPILL LP PARAM**

Type:	Table Slot
UNITS:	VOLUME
Description:	Specifies the Storage points use to take the tangent, line and piecewise approximations for Unregulated Spill Linearization Table linearization
Information:	
Defined by:	User-input

5.5.2.12.1 None

If this method is selected the slot bounds in the slot configuration dialog are set to zero. No additional constraints are generated.

5.5.2.12.2 Opt Monthly Spill

This method is not functional in RPL Optimization.

This method sets the lower and upper bounds on spill. The lower bound is set to zero and the default upper bound is set to a very big number (9,999,999 cms). The default upper bound can be revised in the Spill slot configuration, Upper Bound parameter. No additional constraints are generated beyond these bounds.

5.5.2.12.3 Opt Unregulated

If this method is selected only unregulated spill is considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Unregulated Spill} \quad \text{(EQ 9)}$$

5.5.2.12.4 Opt Regulated

When this method is selected only regulated spill is considered. The lower bound on spill is set to zero and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} \quad \text{(EQ 10)}$$

5.5.2.12.5 Opt Regulated and Unregulated

When this method is selected unregulated and regulated spill are considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} + \text{Unregulated Spill} \quad \text{(EQ 11)}$$

5.5.2.12.6 Opt Regulated and Bypass

When this method is selected only regulated and bypass spill are considered and the lower bound on spill is set to zero and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} + \text{Bypass} \quad \text{(EQ 12)}$$

5.5.2.12.7 Opt Regulated, Bypass and Unregulated

When this method is selected unregulated, regulated and bypass spill are considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} + \text{Unregulated Spill} + \text{Bypass} \quad \text{(EQ 13)}$$

5.5.2.12.8 Opt Bypass, Regulated and Unregulated

When this method is selected unregulated, regulated and bypass spill are considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Bypass} + \text{Regulated Spill} + \text{Unregulated Spill} \quad \text{(EQ 14)}$$

5.5.2.13 Pool Elevation Linearization Automation

This category is no longer supported in RPL optimization.

5.5.2.14 Optimization Head Computation

The methods in this category are no longer supported in RPL optimization.Appearance of this category is dependent on selection of the Independent Linearizations method for the Optimization Power category.

5.5.2.15 Optimization Power

The Optimization Power category allows the user to select which type of linearizations will be used for linearizing various slots. For Independent Linearizations all slots are linearized separately according to the user specified methods. This category has no dependencies.

5.5.2.15.1 None

This is the default method. It is an error to have this method selected for Optimization. No slots are added for this method.

5.5.2.15.2 Independent Linearizations

The Independent Linearizations method linearizes all the variables separately according to the user selected methods. The variables that need to be linearized vary greatly depending on the other methods selected. See the various Categories and Linearization Approaches for details.

5.5.2.15.3 Power Coefficient

The Power Coefficient method models Power at each time step as Turbine Release multiplied by a Power Coefficient Estimate.

SLOTS SPECIFIC TO THIS METHOD

☞ **POWER COEFFICIENT ESTIMATE**

Type:	Series Slot
UNITS:	POWERPERFLOW
Description:	This represents the estimated Power Coefficient that gets used in the Optimization definition of Power.
Information:	The Power Coefficient Estimate is a required input for the run. It can either be input directly (manually or by DMI), or it could be set by an initialization rule. For example, an initialization rule could set the Power Coefficient Estimate based on a “Seed” (Slot Cache) value. If the Power Coefficient method is selected, and Power Coefficient Estimate is not an input for all time steps, then the run will abort with an error message.
Defined by:	Input

Power at each time step is then defined as:

$$\text{Power} = \text{Turbine Release} \times \text{Power Coefficient Estimate}$$

5.5.2.15.4 Power Surface Approximation

This method allows for the modeling of dynamic Operating Head to be incorporated into Optimization Power modeling. It provides a significant improvement in the Power approximation error over the linearization using the Power LP Param table, which assumes a constant Operating Head over the entire run period. The improvement is especially significant for projects that exhibit a wide range of Operating Head over the course of the run, particularly if the optimization policy is trying to maximize Power or set Power greater than or equal to a value.

The method introduces a new variable, PSA Head Factor, which represents the weighted contributions of Storage, Spill and Tailwater Base Value to Operating Head. The method generates a set of planes, the Power Surface, to constrain power as a function of Turbine Release and the PSA Head Factor.

The Power Surface can be thought of as pavement over a warped bridge that is level on one end and sloped and higher on the other end. The bridge ends represent zero Turbine Release and maximum Turbine Release respectively. In addition, the bridge is bowed concave across the center line everywhere except the level (zero Turbine Release) end. This bowing reflects Power as a function of the Head Factor for a fixed value of Turbine Release. The edges of the bridge represent Power as a function of Turbine Release for the low and high values of the Head Factor. Additional points defining a grid on the surface correspond to intermediate values of Turbine Release and the Head Factor. The user can define dimensions and points used for this grid. A piecewise linear surface is composed of cutting planes with each plane defining one triangle on the surface. A surface underneath the bridge composed of two planes defines a lower bound on Power given Turbine Release and Head Factor values.

Because only two planes can be defined for the lower bounds on power, additional approximation error can be introduced if the optimization policy has an incentive to minimize power. In these cases the optimization solution can have an incentive to under-approximate Power for a given Turbine Release, and thus the Post-optimization Rulebased Simulation will calculate a larger Power value than the optimization value for Power.

SLOTS SPECIFIC TO THIS METHOD

The user enters data into three slots:

PSA Sample Input,

PSA Head Factor Grid Lines

PSA Turbine Release Grid Lines

The remaining table slots associated with the method are automatically populated by RiverWare. A description of all of the slots associated with this method is given below.

☞ **PSA SAMPLE INPUT**

Type: Table Slot
UNITS: FLOW, VOLUME, FLOW, LENGTH
Description: This table contains user-specified sample values for Turbine Release, Storage, Total Spill (if applicable) and Tailwater Base Value (if applicable) that are used by RiverWare to compute the PSA Head Factor Weights. The four parameters in this

table represent all of the parameters that contribute to Operating Head. In other words, if a value is known for each of these parameters, it is possible to calculate the corresponding Operating Head.

Information: The values in this table should span the full possible range for each parameter for the given run. For example, the Turbine Release column should contain a value of 0, the highest possible turbine release from the reservoir, and any other intermediate values specified by the user. Specifying tighter upper and lower bounds for each parameter will improve the approximation, but it is important that all possible values for each variable within the run are spanned by the range for that parameter in the table. The user can determine how many rows to include in the table. It is expected that typically, 3-4 values will be specified for each parameter. An example is shown below. The Spill column should only contain values if a method other than None has been selected in the Spill category. Otherwise the Spill column should be left empty (will display "NaN"). The Tailwater Base Value column should only contain values if the reservoir's Tailwater Base Value slot is linked to the Pool Elevation slot of a downstream reservoir (i.e. if the Downstream Reservoir Pool Elevation contributes to the calculation of Operating Head). Otherwise the Tailwater Base Value column should be left empty.

Defined by: User Input

Turbine Release	Storage	Spill (Total)	Tailwater Base Value
0	0	0	500
100	1000	50	510
200	2000	100	520

☞ PSA HEAD FACTOR GRID LINES

Type: Scalar Slot

UNITS: NONE

Description: The number of PSA Head Factor values for each Turbine Release value used when calculating the PSA Grid Points

Information: This must be an integer greater than or equal to 2. The value should typically range from 2 to 4. A larger number will improve the approximation but will increase run time.

Defined by: User Input

☞ PSA TURBINE RELEASE GRID LINES

Type:	Scalar Slot
UNITS:	NONE
Description:	The number of Turbine Release values for each PSA Head Factor value used when calculating the PSA Grid Points
Information:	This must be an integer greater than or equal to 2. The value should typically range from 2 to 4. A larger number will improve the approximation but will increase run time.
Defined by:	User Input

☞ PSA SAMPLE OUTPUT

Type:	Table Slot
UNITS:	FLOW, VOLUME, FLOW, VOLUME, POWER
Description:	A table containing all permutations of the values entered in the PSA Sample Input table slot along with the Power calculated for each combination of Sample Input values. This sample output is used to calculate the PSA Head Factor Weights.
Information:	RiverWare will populate this table at the beginning of the run. For the PSA Sample Input table shown above, with three values in each of the four columns, this table would contain 81 rows, one for each permutation. For a given combination of Turbine Release, Storage, Spill and Tailwater Base Value, RiverWare can calculate the corresponding Power. The third column will contain the Downstream Storage values corresponding to the Tailwater Base Value entries in the PSA Sample Input slot.
Defined by:	Populated by RiverWare at the beginning of the run

For each row *i* in the table:

$$\mathrm{SamplePower}_i = f(\mathrm{SampleTurbineRelease}_i, \mathrm{SampleStorage}_i, \mathrm{SampleSpill}_i, \mathrm{SampleTailwaterBaseValue}_i)$$

The calculation of SamplePower depends on the selected Power and Tailwater methods.

☞ PSA HEAD FACTOR WEIGHTS

Type:	Table Slot
UNITS:	PERTIME, NONE, PERTIME
Description:	The weights for Storage, Spill and Downstream Storage used when calculating PSA Head Factor
Information:	This is a 1 x 3 table with a single weight for Storage, Spill and Downstream Storage. The weights are calculated by RiverWare by performing a regression on the values in the PSA Sample Outputs table slot. If one of the parameters is not used (contains NaN for all rows in the PSA Sample Input table), the corresponding weight in this table slot will be zero.
Defined by:	Calculated by regression at the beginning of the run

PSA HEAD FACTOR

Type: Series Slot
UNITS: FLOW
Description: This slot is a variable in the optimization solution and represents the weighted contribution of Storage, Spill and Downstream Storage to Operating Head and thus power. It can be thought of as a storage analog of Operating Head.
Information: This variable is used in the automatically generated constraints that define Power. It will not, generally display any values, but the user can write a post-optimization rule that calculates the value of this variable at each time step using the equation shown below and the values in the PSA Head Factor Weights table slot.
Defined by: $\text{PSA Head Factor} = \alpha \cdot \text{Storage} + \beta \cdot \text{Spill} + \gamma \cdot \text{Downstream Storage}$

The coefficients α, β and γ come from the PSA Head Factor Weights table slot.

PSA GRID POINTS

Type: Table Slot
UNITS: FLOW, VOLUME, FLOW, VOLUME, POWER, FLOW
Description: Points used to calculate the planes for the PSA Max Constraints and PSA Min Constraints, one row for each point with columns for Turbine Release, Storage, Spill, Downstream Storage, Power and PSA Head Factor.
Information: The number of rows (points) in this table equals the product of the number of lines in the PSA Head Factor Grid Lines and PSA Turbine Release Grid Lines scalar slots. For each grid point, RiverWare calculates the value of Power and the PSA Head Factor corresponding to the parameter values in that row.
Defined by: Populated by RiverWare at the beginning of the run

Turbine Release Column: Smallest and largest values from PSA Sample Input and evenly spaced intermediate values corresponding to the number of lines in the PSA Turbine Release Grid Lines scalar slot

Storage, Spill and Downstream Storage Columns: Smallest and largest values from PSA Sample Input and evenly spaced intermediate values corresponding to the number of lines in the PSA Head Factor Grid Lines scalar slot

Power Column: Determined by selected Power method, for each row *i* in the table

$$\text{Power}_i = f(\text{TurbineRelease}_i, \text{Storage}_i, \text{Spill}_i, \text{DownstreamStorage}_i)$$

Head Factor Column: $\text{HeadFactor}_i = \alpha \cdot \text{Storage}_i + \beta \cdot \text{Spill}_i + \gamma \cdot \text{DownstreamStorage}_i$

The coefficients α, β and γ come from the PSA Head Factor Weights table slot.

PSA MAX CONSTRAINTS

Type: Table Slot
UNITS: FLOW, FLOW, FLOW, FLOW, FLOW, FLOW, POWERPERFLOW, POWERPERFLOW, POWER
Description: This table contains the planes that define the upper bounds for power as a function of Turbine Release and PSA Head Factor, one row for each plane. Power is constrained to be less than or equal to the planes defined in this table. The first six columns

	contain three pairs of Turbine Release and PSA Head Factor values that define three points on the plane. The remaining three columns contain the corresponding Turbine Release Coefficient, Head Factor Coefficient and Constant Term that define the same plane and are used in the actual constraint expressions.
Information:	The points used to define the planes come from the Turbine Release and Head Factor points in the PSA Grid Points table slot. RiverWare will not use all of the possible combinations of Turbine Release and Head Factor points but rather will only use the combinations necessary to define the Power Surface.
Defined by:	Populated by RiverWare at the beginning of the run

For each row *i* in the PSA Max Constraints table slot a constraint is added to the optimization problem:

$$\text{Power} \le u_i \cdot \text{Turbine Release} + v_i \cdot \text{PSA Head Factor} + w_i$$

The coefficients u_i, v_i and w_i are the Turbine Release Coefficient, Head Factor Coefficient and Constant Term from row *i* of the PSA Max Constraints table. Turbine Release and PSA Head Factor are the actual variables in the optimization problem.

☞ PSA MIN CONSTRAINTS

Type:	Table Slot
UNITS:	FLOW, FLOW, FLOW, FLOW, FLOW, FLOW, POWERPERFLOW, POWERPERFLOW, POWER
Description:	This table contains the planes that define the lower bounds for power as a function of Turbine Release and PSA Head Factor, one row for each plane. Power is constrained to be greater than or equal to the planes defined in this table. The first six columns contain three pairs of Turbine Release and PSA Head Factor values that define three points on the plane. The remaining three columns contain the corresponding Turbine Release Coefficient, Head Factor Coefficient and Constant Term that define the same plane and are used in the actual constraint expressions
Information:	This table will always contain two rows. Only two planes can be defined for the lower bounds on power.
Defined by:	Populated by RiverWare at the beginning of the run

For each row *i* in the PSA Min Constraints table slot a constraint is added to the optimization problem:

$$\text{Power} \ge u_i \cdot \text{Turbine Release} + v_i \cdot \text{PSA Head Factor} + w_i$$

The coefficients u_i, v_i and w_i are the Turbine Release Coefficient, Head Factor Coefficient and Constant Term from row *i* of the PSA Min Constraints table. Turbine Release and PSA Head Factor are the actual variables in the optimization problem.

METHOD SUMMARY

Three slots require input from the user. Descriptions of these slots are provided above.

PSA Sample Input

PSA Turbine Release Grid Points

PSA Head Factor Grid Points

Then at the start of the run RiverWare carries out the following steps:

- Populate the PSA Sample Output Table with all permutations of parameters in the PSA Sample Input table slot
- Calculate the PSA Head Factor Weights using a regression based on values in the PSA Sample Output table slot
- Add defining constraints for PSA Head Factor using the PSA Head Factor Weights
- Populate the PSA Grid Points table slot based on the number of grid lines specified by the user
- Generate upper bound planes using the PSA Grid Points and add the corresponding constraints on Power to the Optimization problem
- Generate lower bound planes using the PSA Grid Points and add the corresponding constraints on Power to the Optimization problem

METHOD DETAILS

This section provides details of the calculations carried out by RiverWare to generate the constraints that define the Power Surface. This material is included for reference only.

1. RiverWare populates the PSA Sample Output table slot. All permutations of values in the PSA Sample Input table slot are determined, and a row for each is added to the PSA Sample Output table slot. RiverWare may make some adjustments to prevent infeasible combinations. For example, the highest Turbine Release value may not be possible for all combinations of Storage, Spill and Tailwater Base Value. RiverWare will adjust the Turbine Release to use the maximum turbine release for the resulting Operating Head. The table is organized in blocks of constant Turbine Release, with the lowest Turbine Release block first and then increasing Turbine Release. The values in the remaining columns are ordered corresponding to increasing Operating Head. Storage is increasing. Spill and Tailwater Base Value (or Downstream Storage) are decreasing. For each combination of values (each row), RiverWare calculates the corresponding Operating Head and Power. The Power value is added to each row in the PSA Sample Output table. The details of these calculations are shown here for a given row *i* in the PSA Sample Output table. Note that Power_i is the only calculated quantity displayed in the table. Values from the intermediate calculations shown below are not displayed in the table.

$$\mathrm{PoolElevation}_i = f(\mathrm{Storage}_i)$$

Pool Elevation is calculated by interpolation on the reservoir Elevation Volume Table.

If Tailwater Base Value is linked to Pool Elevation of a downstream reservoir:

$$\mathrm{TailwaterBaseValue}_i = \mathrm{DownstreamPoolElevation}_i = f(\mathrm{DownstreamStorage}_i)$$

Tailwater Base Value is calculated by interpolation on the downstream reservoir Elevation Volume Table. Tailwater Base Value is only calculated from the Downstream Storage if either Base Value Plus Lookup Table, Stage Flow Lookup Table or Coefficients Table is the selected Tailwater method *and* Tailwater Base Value is linked to the Pool Elevation of the downstream reservoir. Otherwise Tailwater Base Value is unused, and thus Operating Head and Power are not dependent on Downstream Storage.

$$\text{Outflow}_i = \text{TurbineRelease}_i + \text{Spill}_i$$

Spill will only be non-zero if a method other than None is selected in the Spill category.

$$\text{TailwaterElevation}_i = f(\text{Outflow}_i, \text{TailwaterBaseValue}_i)$$

The tailwater calculation is dependent on the selected method in the Tailwater category.

$$\text{OperatingHead}_i = \text{PoolElevation}_i - \text{TailwaterElevation}_i$$

$$\text{Power}_i = f(\text{TurbineRelease}_i, \text{OperatingHead}_i)$$

The Power calculation is dependent on the selected method in the Power category.

2. RiverWare calculates the PSA Head Factor Weights using a combination of linear regression and averages.

FOR EACH Turbine Release value *i*

FOR EACH combination of Spill and Tailwater Base Value (or Downstream Storage), *k,* perform a linear regression to calculate a temporary Storage coefficient g_{ik}:

$$g_{ik} = \frac{n\sum(\text{Power}_j \times \text{Storage}_j) - \left(\left(\sum \text{Power}_j\right)\left(\sum \text{Storage}_j\right)\right)}{n\left(\sum \text{Storage}_j^2\right) - \left(\sum \text{Storage}_j\right)^2}$$

where *n* is the number of Storage values.

Calculate the temporary Storage coefficient for the given Turbine Release, g_i:

$$g_i = \frac{\sum g_{ik}}{n}$$

where *n* is the number of Spill and Downstream Storage) combinations.

FOR EACH combination of Storage and Downstream Storage, *k,* perform a linear regression to calculate a temporary Spill coefficient e_{ik}:

$$e_{ik} = \frac{n\sum(\text{Power}_j \times \text{Spill}_j) - \left(\left(\sum \text{Power}_j\right)\left(\sum \text{Spill}_j\right)\right)}{n\left(\sum \text{Spill}_j^2\right) - \left(\sum \text{Spill}_j\right)^2}$$

where *n* is the number of Spill values.

Calculate the temporary Spill coefficient for the given Turbine Release, e_i:

$$e_i = \frac{\sum e_{ik}}{n}$$

where *n* is the number of Storage and Downstream Storage combinations.

FOR EACH combination of Storage and Spill, *k*, perform a linear regression to calculate a temporary Downstream Storage coefficient f_{ik}:

$$f_{ik} = \frac{n\sum(\mathrm{Power}_j \times \mathrm{DownstreamStorage}_j) - \left(\left(\sum \mathrm{Power}_j\right)\left(\sum \mathrm{DownstreamStorage}_j\right)\right)}{n\left(\sum \mathrm{DownstreamStorage}_j^{\,2}\right) - \left(\sum \mathrm{DownstreamStorage}_j\right)^2}$$

where *n* is the number of Downstream Storage values.

Calculate the temporary Downstream Storage coefficient for the given Turbine Release, f_i:

$$e_i = \frac{\sum e_{ik}}{n}$$

where *n* is the number of Storage and Spill combinations.

Calculate average ratios over all Turbine Release values:

$$c = \frac{\sum (e_i/g_i)}{n}$$

$$d = \frac{\sum (f_i/g_i)}{n}$$

where *n* is the number of Turbine Release values.

Calculate the final Head Factor Weights:

$$\text{Storage Weight} = \frac{1}{c}$$

$$\text{Spill Weight} = -1$$

If Tailwater Base Value is linked to the downstream Pool Elevation:

$$\text{Downstream Storage Weight} = \frac{d}{c}$$

These three weight values are displayed in the three columns of the PSA Head Factor Weights table slot.

In the optimization solution the PSA Head Factor is defined at a given time step by

If Tailwater Base Value is linked to the downstream Pool Elevation:

$$\text{PSA Head Factor} = \mathrm{StorageWeight} \times \mathrm{Storage} + \mathrm{SpillWeight} \times \mathrm{Spill} + \mathrm{DownstreamStorageWeight} \times \text{Downstream Storage}$$

If Tailwater Base Value is not linked:

$$\text{PSA Head Factor} = \text{StorageWeight} \times \text{Storage} + \text{SpillWeight} \times \text{Spill}$$

3. RiverWare calculates the grid points for the PSA Grid Points table slot. The number of rows (points) in this table equals the product of the number of lines in the PSA Head Factor Grid Lines and PSA Turbine Release Grid Lines scalar slots. The smallest and largest values in the PSA Sample Input slot lead to the smallest and largest values for each of the parameters that affect power (Turbine Release, Storage, Spill and Downstream Storage. One row in this table will correspond to the smallest Turbine Release, smallest Storage, largest Spill and largest Downstream Storage, and thus smallest power. Another row will correspond to the largest Turbine Release, largest Storage, smallest Spill and smallest Downstream Storage, and thus the largest Power. RiverWare generates evenly spaced values between the extremes for each variable.Within the PSA Grid Points table, for a given Turbine Release value, Storage is increasing and Spill and Downstream Storage are decreasing. For each grid point, RiverWare calculates the value of Power and the PSA Head Factor corresponding to the parameter values in each row and adds these to the final two columns in the table.

For each row *i* in the table:

$$\text{Power}_i = f(\text{TurbineRelease}_i, \text{Storage}_i, \text{Spill}_i, \text{DownstreamStorage}_i)$$

The Power calculation depends on the method selected in the Power category.

$$\text{HeadFactor}_i = \alpha \cdot \text{Storage}_i + \beta \cdot \text{Spill}_i + \gamma \cdot \text{DownstreamStorage}_i$$

The coefficients α, β and γ come from the PSA Head Factor Weights table slot.

4. RiverWare calculates the upper bound planes. These are entered in the PSA Max Constraints table slot, one row for each plane, where the planes represent Power as a function of Turbine Release and PSA Head Factor. For each combination of three points in the PSA Grid Points Table, RiverWare calculates the corresponding plane. For any three points, the plane is defined by three linear equations, which can be represented in matrix form.

$$\bar{P} = \mathbf{A}\bar{u}$$

$$\bar{P} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} QT_1 & HF_1 & 1 \\ QT_2 & HF_2 & 1 \\ QT_3 & HF_3 & 1 \end{bmatrix}, \qquad \bar{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

The coefficients that define the plane are solved for by

$$\bar{u} = \mathbf{A}^{-1}\bar{P}$$

RiverWare only uses the planes that are necessary to define the Power Surface. Each plane is checked against all remaining points in the PSA Grid Points table. If the Power value for any of the remaining points lies above the given plane, then that plane is rejected (i.e. it would over-constrain Power, resulting in an under-estimation of Power). For each plane *i* that is used, RiverWare adds the following constraint to the optimization problem for each time step:

$$\text{Power} \le u_i \cdot \text{Turbine Release} + v_i \cdot \text{PSA Head Factor} + w_i$$

5. RiverWare calculates the lower bound planes. These are entered in the PSA Min Constraints table slot, one row for each plane. Only two planes can be calculated for the lower bounds. The points used for the two planes are as follows:

Plane 1:

Point A1 - Min Turbine Release, Max Head Factor

Point B1 - Max Turbine Release, Max Head Factor

Point C1 - Min Turbine Release, Min Head Factor

Plane 2:

Point A2 - Min Turbine Release, Max Head Factor

Point B2 - Max Turbine Release, Max Head Factor

Point C2 - Max Turbine Release, Max Head Factor

The coefficients defining the planes are calculated using the same matrix calculation described for the upper bound planes. For each lower bound plane *i*, RiverWare adds the following constraint to the optimization problem for each time step:

$$\text{Power} \geq u_i \cdot \text{Turbine Release} + v_i \cdot \text{PSA Head Factor} + w_i$$

A note on the lower bound approximation:

Because only two planes can be defined for the lower bounds on power, additional approximation error can be introduced if the optimization policy has an incentive to minimize power. This can occur if Power is included in a Minimize objective. It can also occur if the optimization problem contains constraints that require Power to be less than or equal to value. Note that these types of constraints can be introduced through multiple forms of RPL optimization goals. The first is basic less than or equal to constraint.

ADD CONSTRAINT $\text{Res.Power}[t] \leq \text{Value}$

Policy that requires Power to be equal to a value also adds a less than or equal to constraint. The following RPL statement:

ADD CONSTRAINT $\text{Res.Power}[t] = \text{Value}$

actually adds the following two constraints to the optimization problem:

$\text{Res.Power}[t] \leq \text{Value}$

$\text{Res.Power}[t] \geq \text{Value}$

A constraint to require the sum of Power from multiple reservoirs to be less than or equal to a value will have a similar effect. A final manner in which this can occur is through policy that minimizes Power indirectly through user-defined variables. For example, assume a user-defined variable is defined by the following RPL constraint:

ADD CONSTRAINT $\text{Res.Power}[t] + \text{UserVariable}[t] = \text{Value1}$

Then later the variable is constrained by

ADD CONSTRAINT $\text{UserVariable}[t] \geq \text{Value2}$

An equivalent constraint would be

ADD CONSTRAINT $\text{Res.Power}[t] \leq \text{Value1} - \text{Value2}$

It is possible to reduce the approximation error in these cases by adding constraints to the RPL Optimization Goal Set that essentially create more restrictive planes for the lower bound. These types of constraints are model-specific. Contact CADSWES if assistance is required to add more restrictive lower bounds to the Power Surface Approximation.

5.5.2.16 Power Linearization Automation

The Power Linearization Automation category allows the selection of the approximation points used for linearizing the power related slots to be done automatically or entered by the user.

It is dependent on selection of Independent Linearizations for the Optimization Power category and selection of None or Variable Head for the Optimization Head Computation category. (The Variable Head method is not currently functional in RPL Optimization.)

5.5.2.16.1 None

The method requires the user to input the approximation points for the power, best turbine flow (depending on the power method selected) and turbine capacity into the appropriate Power LP Param, Best Turbine Flow LP Param and Turbine Capacity LP Param tables, respectively.

5.5.2.16.2 Plant Automation

This method's availability for user selection is dependent on selection of Plant Power Coefficient in the Power category.

This method proceeds as follows, beginning with the Turbine Capacity LP Param table:

1) In the Max Turbine Q table, get the head values in the first row and the second to last row of the table, calling them FIRSTHEAD and LASTHEAD respectively.

2) Calculate AVEHEAD as the average of FIRSTHEAD and LASTHEAD.

3) Find the maximum value in the Turbine Capacity column (MaxTurbineCapacity) and its corresponding Operating Head (MaxTCOperatingHead) in the Max Turbine Q table. Given the nature of this table, the maximum value is not necessarily the last value. Also, if successive rows have the same Turbine Capacity, the first one (and its corresponding Operating Head) is chosen.

4) If maxTCOperatingHead > LASTHEAD - 0.5 m, then maxTCOperatingHead is set to LASTHEAD - 1 m.

The Turbine Capacity LP Param table is then defined using the variables above: The tangent approximation value is set as AVEHEAD. The line approximation points are set to FIRSTHEAD + 0.5 m and

LASTHEAD - 0.5 m. The piecewise approximation points are set to FIRSTHEAD + 0.5, MaxTCOperatingHead, and LASTHEAD - 0.5.

The method then repeats the process using the Best Turbine Q table. (In this table the second column is now called Best Capacity whereas the second column in the Max Turbine Q table is called Turbine Capacity.

Next the Power LP Param table is filled in:

1) If the initial Operating Head is not known, it is calculated as initial Pool Elevation - initial Tailwater Elevation. The Elevation Volume Table and initial Storage will be used to determine initial Pool elevation, if necessary. The initial Operating Head is called MIDHEAD.

2) The Maximum Turbine Q table is queried using MIDHEAD to determine the corresponding maximum turbine flow, here called FLOWMAX.

3) The Best Turbine Q table is queried using MIDHEAD to determine the corresponding best turbine flow. FLOWBEST is set to this best turbine flow - 0.01 cms. If FLOWBEST is greater than or equal to FLOWMAX, then reset FLOWBEST to FLOWMAX - 0.01.

The Power LP Param table is then defined using the variables above: The Operating Head value is set to MIDHEAD. The tangent approximation value is set as (FLOWBEST + FLOWMAX) / 2. The line approximation points are set to 0 and (FLOWBEST + FLOWMAX) / 2. The piecewise approximation points are set to 0, FLOWBEST, and FLOWMAX.

Now the method creates the Plant Power Table (defining power as a function of operating head and turbine release). The method adds the best and max power generation (power and flow) points for each operating head by combining the data in the Best Turbine Q, Best Power Coefficient, Max Turbine Q, and Max Power Coefficient tables. The details of the algorithm are as follows. For each row (each Operating Head) in the Max Turbine Q table:

1. Determine the Operating Head (OPHEAD) and Turbine Capacity (MAXQ) values.

2. Query the Best Turbine Q table using OPHEAD to determine Best Turbine Flow (BESTQ).

3. Query the Best Power Coefficient table using the Operating head to determine the Best Power Coefficient.

4. Query the Max Power Coefficient table using the Operating head to determine the Max Power Coefficient.

5. Query the Best Power Coefficient table using the Operating head to determine the Best Power Coefficient.

6. Calculate the MAXPOWER as MAXQ * Max Power Coefficient.

7. Calculate the BESTPOWER as BESTQ * Best Power Coefficient.

8. Create a row in the table using OPHEAD, 0, 0 (Operating Head, Turbine Release, and Power columns, respectively).

9. If BESTQ > 0, then create a row in the table using OPHEAD, BESTQ, BESTPOWER.

10. If MAXQ does not equal BESTQ, then create a row in the table using OPHEAD, MAXQ, MAXPOWER.

11. If MAXQ does not equal MaxTurbineCapacity (retained from the Turbine Capacity LP Param table work above), then create a row in the table using OPHEAD, MaxTurbineCapacity, MAXPOWER.

Finally, the initial (timestep) value of the Power Coefficient slot is assigned as determined by querying the Best Power Coefficient table using MIDHEAD (retained from the Power LP Param table work above) for Operating Head.

5.5.2.16.3 Peak Automation

This method's availability for user selection is dependent on selection of Peak Power in the Power category.

This method proceeds as follows, beginning with the Turbine Capacity LP Param table:

1. The Number of Units table is queried. Its value is herein called NUMUNITS.

2. Get the head values in the first row and the second to last row of the Best Generator Flow table, calling them FIRSTHEAD and LASTHEAD respectively.

3. Calculate AVEHEAD as the average of FIRSTHEAD and LASTHEAD.

4. Find the maximum value of flow in the Type #1 Flow column (MaxFlow) and its corresponding Operating Head (HEADMAXQ) in the Best Generator Flow table. Given the nature of this table, the maximum flow value is not necessarily the last value. Also, if successive rows have the same Type #1 Flow, the first one (and its corresponding Operating Head) is chosen.

5. If HEADMAXQ > LASTHEAD - 0.5 m, then HEADMAXQ is set to LASTHEAD - 1.0 m.

6. Calculate ALLMAXFLOW as MaxFlow * NUMUNITS.

7. The Turbine Capacity LP Param table is then defined using the variables above: The tangent approximation value is set as AVEHEAD. The line approximation points are set to FIRSTHEAD + 0.5 m and LASTHEAD - 0.5 m. The piecewise approximation points are set to FIRSTHEAD + 0.5 m, HEADMAXQ, and LASTHEAD - 0.5 m.

Next the Power LP Param table is filled in:

1. If the initial Operating Head is not known, it is calculated as initial Pool Elevation - initial Tailwater Elevation. The Elevation Volume Table and initial Storage will be used to determine initial Pool elevation, if necessary. The initial Operating Head is called MIDHEAD.

2. The Best Generator flow is queried using MIDHEAD to determine the corresponding Type #1 Flow, here called FLOWMAX.

3. Calculate ALLFLOWMAX as FLOWMAX * NUMUNITS - 0.01 cms (where NUMUNITS is retained from the Turbine Capacity LP Param table work above).

4. Calculate ALLFLOWBEST as ALLFLOWMAX - 0.01 cms.

The Power LP Param table is then defined using the variables above: The Operating Head value is set to MIDHEAD. The tangent approximation value is set as the average of ALLFLOWBEST and ALLFLOWMAX. The line approximation points are set to 0 and the average of ALLFLOWBEST and ALLFLOWMAX. The piecewise approximation points are set to 0 cms, ALLFLOWBEST and ALLFLOWMAX.

Now the method creates the Plant Power Table (defining power as a function of operating head and turbine release). For each row (each Operating Head) in the Best Generator Flow table:

1. Determine the Operating Head (OPHEAD) and Type #1 Flow (BESTQ) values.

2. Query the Best Generator Power table using OPHEAD to determine Type #1 Power (BESTPOWER).

3. Calculate ALLBESTQ = BESTQ * NUMUNITS (retained from the Turbine Capacity LP Param table work above).

4. Calculate ALLBESTPOWER = BESTPOWER * NUMUNITS (retained from the Turbine Capacity LP Param table work above).

5. Create a row in the Plant Power Table with OPHEAD, 0, 0 (Operating Head, Turbine Release, and Power columns, respectively).

6. If ALLBESTQ > 0, the create a row in the table using OPHEAD, ALLBESTQ, ALLBESTPOWER.

7. If ALLBESTQ does not equal ALLMAXFLOW (retained from the Turbine Capacity LP Param table work above), then create a row in the table using OPHEAD, ALLMAXFLOW, and ALLBESTPOWER.

Finally, the method checks that the MIDHEAD value assigned as Operating Head in the Power LP Param table is contained within the Operating Head range of the Plant Power Table. If it is not, and error occurs.

5.5.2.17 Power

For more information on the simulation methods, click **HERE (Objects.pdf, Section 17.1.1)**. Only a subset of available Simulation methods are available in Optimization. Selection of a method other than those that follow will result in an error.

If the Optimization problem uses Power, either directly or indirectly, then this slot is added to the problem. Before being passed to the optimization solver, this slot is numerically approximated as a function of Operating Head and Turbine Release (Numerical 3-D Approximation). The relationship between Operating Head and Turbine Release will come from the Plant Power Table. This table will either be user-input, or automatically parameterized depending on the method selection in the Power Linearization Automation category. The table will be queried using the points defined in the Power LP Param table. The values in the Power LP Param table also will either be user-input or automatically calculated points, depending on the method selection in the Power Linearization Automation category. If the selected method in the Power Linearization Automation is “none”, then the tables should contain user-input values which are used. For other Power Linearization Automation methods (Plant Automation or

Peak Automation) the tables will contain automatically calculated values which are used.

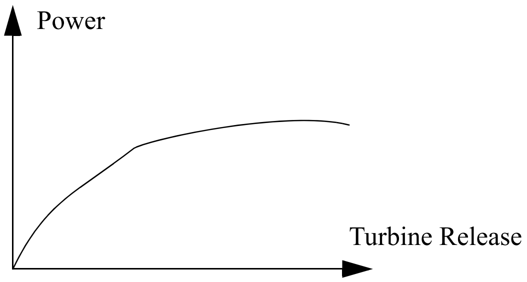
RELATED SLOTS

POWER

Type: Agg Series Slot
UNITS: POWER
Description: Power generated by flow through the turbines
Information:
Defined by: Numerical 3-D Approximation in terms of Operating Head and Turbine Release. Approximation is based on the Plant Power Table. The Power LP Param table contains a value for Operating Head used to index the Operating Head column of the Plant Power Table. This approximated value, therefore, reduces the Power to a function of Turbine Release. The flow values in the Power LP Param table are then used as approximation points indexing the Turbine Release column of the Plant Power Table. The Plant Power Table should have increasing values of Operating Head and Turbine Release. Power should be a concave function of Operating Head, but concavity is not strictly enforced. The preferred order of approximation is substitution, piece-wise, two-point line, tangent.

POWER LP PARAM

Type: Table Slot
UNITS: LENGTH AND FLOW
Description: Specifies the Operating Head and the flow points used to take the tangent, line and piecewise approximations for Power linearization
Information: This table must be user input unless the Power Linearization Automation category has selected either Plant Automation or Peak Automation methods. The best Operating Head to choose should be close to the expected head during optimized period. Tangent approximation is generally not used, nor helpful as it often results in non-zero power for zero flow. The suggested points for a line approximation are 0 flow and best turbine flow for the entire plant. The suggested points for piecewise linearization are 0 flow, 1 unit best turbine flow, 2 units best turbine flow, . . . n units best turbine flow, and maximum turbine flow.
Defined by:



Power Turbine Release relationship. This is for a fixed operating head value. Not drawn to scale.

5.5.2.17.1 Plant Power Coefficient

SLOTS SPECIFIC TO THIS METHOD

In the RPL Optimization mode, the following slot requires special handling.

☞ **BEST TURBINE FLOW**

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow associated with the most efficient power generation for an associated Operating Head
Information:	
Defined by:	Numerical 2-D Approximation in terms of Operating Head, based upon an internal best turbine flow table. For the Plant Power Coefficient method (selected in the Power category) the internal table is the Best Turbine Q table. The Best Turbine Q table is required to have increasing values of Operating Head and Best Capacity and be a concave function of Operating Head. The preferred order of approximation is substitution, piece-wise, tangent, two-point line. The Best Turbine Flow LP Param table values are used as approximation points indexing the best turbine flow table.

If the Optimization problem uses Best Turbine Flow, either directly or indirectly, then this slot is added to the problem. Before being passed to the optimization solver, this slot is numerically approximated as a function of Operating Head (Numerical 2-D Approximation). The relationship between Operating Head and Best Turbine Flow will come from one of two places, depending on other method selection. 1) If the selected method in the Power category is Plant Efficiency Curve, then the relationship is automatically developed from the Plant Power Table. 2) For the Plant Power Coefficient method, the user-input Best Turbine Q table defines the relationship. 3) For other Plant Calculation category methods, Best Turbine Flow must be input.

BEST TURBINE FLOW LP PARAM

Type:	Table Slot
UNITS:	LENGTH
Description:	Specifies the Operating Head points in the best turbine flow relationship used to take the tangent, line and piecewise approximations for Best Turbine Flow linearization
Information:	The best operating head points to choose for a piecewise approximation are generally: minimum Operating Head, Operating Head at max capacity, and maximum Operating Head. These three points typically define most of the shape of the Best Turbine Flow curve.
Defined by:	User-input

BEST TURBINE Q

Type:	Table Slot
UNITS:	LENGTH VS. FLOW
Description:	Table defining the relationship between Operating Head and the most efficient power generation.
Information:	This table is used if the Power category has the Plant Power Coefficient method selected. If the Power category has the Plant Efficiency Curve method selected, then this table is not used, but a similar relationship is automatically developed from the Plant Power Table.
Defined by:	User-input

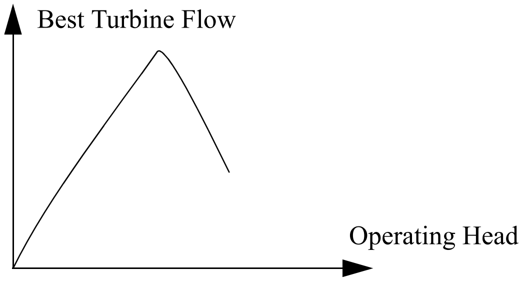
PLANT POWER TABLE

Type:	Table Slot
UNITS:	LENGTH VS. FLOW VS POWER
Description:	Table defining the relationship between Operating Head, Turbine Release, and Power at selected points of operation. In building the Plant Power table, the following approach might be taken. For a given Operating Head create n + 2 rows of data, where n is the number of units comprising the Plant. The first row will reflect no flow through the turbines. The next row will reflect the preferred level of flow through the first unit normally activated. Subsequent rows will reflect incremental flow levels as additional turbines come on line at their respective preferred levels of

flow. The final row will reflect the flow of all available units running at maximum flow capacity.

Information: This table is used to automatically develop the Best Turbine Flow relationship if the Plant Efficiency Curve method IS SELECTED for the Power category. If it is not, then this table is not used; the Best Turbine Q table is used.

Defined by: User-input



Best Turbine Flow - Operating Head relationship. Not drawn to scale.

5.5.2.17.2 Plant Efficiency Curve

The Plant Efficiency Curve method approximates Best Power Flow the same as the Plant Power Calc

5.5.2.17.3 Peak Power

In RPL Optimization, this method creates a turbine release constraint:

Turbine Release <= Power Plant Capacity Fraction * Turbine Capacity * Number of Units. **(EQ 15)**

No actual power value is calculated.

5.5.2.17.4 Peak and Base

In RPL Optimization, this method creates a turbine release constraint:

Turbine Release <= Power Plant Capacity Fraction * Turbine Capacity * Number of Units. **(EQ 16)**

No actual power value is calculated.

5.5.2.17.5 Unit Power Table

The Unit Power slots and simulation algorithm is described **HERE (Objects.pdf, Section 17.1.1.12)**. The optimization formulation is described **HERE (Section 8.2.6)**.

When the Unit Power Table method is selected, the following categories are available: Cavitation, **HERE (Section 5.5.2.24)** , Avoidance Zones, **HERE (Section 5.5.2.25)**, Start Up, **HERE (Section 5.5.2.22)**, and Head Loss, **HERE (Section 5.5.2.23)**, and Frequency Regulation, **HERE (Section 5.5.2.26)**

The status of a unit at a given timestep can be one of the following:

- Available
- Unavailable
- Must run

At the highest level, an entire plant may never be able to provide an ancillary service like Regulation. The user could indicate this by selecting "None" in the appropriate category. On the other hand, the following slot settings will allow users to specify unit availability / must run for individual time steps for generation and ancillary services. Without any inputs, the units will default to being available.

If the Unit Power Table method, but not the Frequency Regulation, method is selected, the user could specify through optimization RPL policy (note, setting certain slots may be possible but not preferred):

- Unit u must generate at time t:
  - Unit Is Generating [t,u] = 1, or
  - Unit Energy [t,u] = value
- Unit u is unavailable at time t:
  - Unit Is Generating [t,u] = 0, or
  - Unit Energy [t,u] = 0

If the Frequency Regulation method is selected, the user could specify through optimization policy (RPL):

- Unit u must regulate up at time t:
  - Regulation Up [t,u] = value
- Unit u must regulate down at time t:
  - Regulation Down [t,u] = value
- Unit u must regulate at time t:
  - Regulation [t,u] = value (which implies Regulation Up [t,u] = value and Regulation Down [t,u] = value)

If a unit is unavailable for some type of regulation, then a value can be 0.

5.5.2.18 Turbine Capacity

If the selected method for the Optimization Head Computation category is either None or Variable Head (not supported in RPL Optimization), then Turbine Capacity is numerically approximated before being passed to the optimization solver. The slot is approximated as a function of Operating Head (Numerical 2-D Approximation). The relationship between Operating Head and Turbine Capacity will come from one of several places, depending on the method selected in the Power category. 1) If plant-Efficiency Curve is chosen, the relationship is automatically developed from the Plant Power Table. 2) Otherwise, if Peak Power or Peak and Base is chosen, the relationship comes from the Best Generator Flow table. 3) If none of these methods are chosen, then the relationship is contained in the user-input Max Turbine Q slot.

RELATED SLOTS

☞ TURBINE CAPACITY

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow capacity of the turbine(s)
Information:	
Defined by:	Numerical 2-D Approximation in terms of Operating Head, based upon a maximum turbine capacity table. This capacity table is determined in various ways according to the Power method:

Plant Efficiency Curve: an automatically generated maximum turbine flow table developed from the Plant Power Table

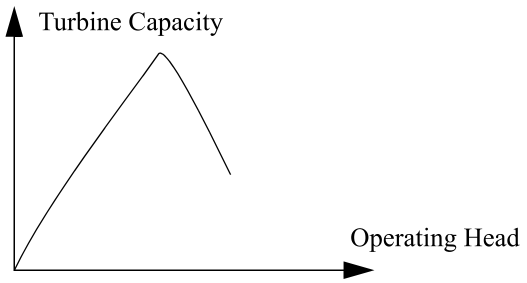
Peak Power or Peak and Base: Best Generator Flow table

All Others: Max Turbine Q

The Turbine Capacity LP Param table values are used as approximation points indexing the selected maximum turbine capacity table. The maximum turbine capacity table is required to have increasing values of Operating Head. Turbine Capacity in that table is required to be a concave function of Operating Head. The preferred order of approximation is substitution, piece-wise, tangent, two-point line.

☞ TURBINE CAPACITY LP PARAM

Type:	Table Slot
UNITS:	LENGTH VS. LENGTH VS. LENGTH
Description:	Specifies the operational head points used to take the tangent, line and piecewise approximations for Turbine Capacity linearization
Information:	For the piecewise approximation the best operating head points to chose are generally: minimum Operating Head, Operating Head at max capacity, and maximum Operating Head. These three points typically define most of the shape of the Turbine Capacity curve.
Defined by:	User-input unless the selected method in the Power Linearization Automation category is not “None”



Turbine Capacity Operating Head relationship. Not drawn to scale.

5.5.2.19 Optimization Tailwater

Click **HERE (Objects.pdf, Section 17.1.7)** for more information on the Simulation methods for Tailwater. Only a subset of available methods are supported in Optimization. These methods are: Linked or Input, Base Value Only, Base Value Plus Lookup Table, Stage Flow Lookup Table, and Coefficients Table. Selection of a method other than those will result in an error. The method selected in the Optimization Tailwater category should correspond to the one selected in the simulation Tailwater category.

Tailwater Elevation may be part of the Optimization problem and it is handled differently for each method. Below is a description of the behavior of Tailwater Elevation.

5.5.2.19.1 Opt Linked or Input

If Tailwater Elevation is not input, then Tailwater Elevation is replaced by TailwaterBaseValue: Tailwater Elevation = Tailwater Base Value.

☞ **TAILWATER BASE VALUE**

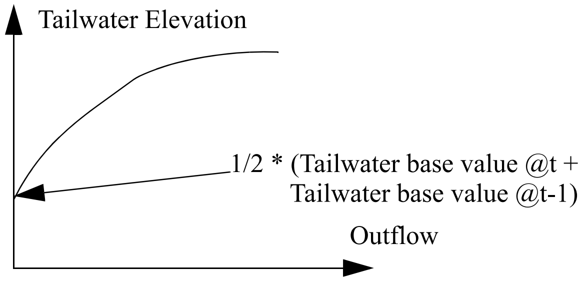
Type:	Series
UNITS:	LENGTH
Description:	Described in the simulation documentation **HERE (Objects.pdf, Section 17.1.7.2)**

5.5.2.19.2 Opt Base Value Only

If Tailwater Elevation is not input, then Tailwater Elevation is replaced by (Tailwater Base Value(t) + Tailwater Base Value(t-1)) / 2

☞ **TAILWATER BASE VALUE**

Type:	Series
UNITS:	LENGTH
Description:	Described in the simulation documentation **HERE (Objects.pdf, Section 17.1.7.3)**



Tailwater Outflow relationship. Not drawn to scale.

5.5.2.19.3 Opt Base Value Plus Lookup Table

If Tailwater Elevation is not input, this slot is replaced by a mathematical expression based on Tailwater Base Value and a numerical approximation of Tailwater Elevation as a function of Outflow (Numerical

2-D Approximation) as defined by the user in the Tailwater Table

Tailwater Elevation is constrained to be (tailwaterBaseValue(t) + tailwaterBaseValue(t-1))/2 + tempTWLookupValue, where tempTWLookupValue is a Numerical 2-D Approximation of increase in Tailwater Elevation due to flow as described in the Tailwater Table. The Tailwater Table Lookup LP Param table values for outflow are used as approximation points indexing the Tailwater Table to determine tempTWLookupValue from the Tailwater Elevation column.

☞ TAILWATER BASE VALUE

Type:	Series
UNITS:	LENGTH
Description:	Described in the simulation documentation **HERE (Objects.pdf, Section 17.1.7.4)**

☞ TAILWATER TABLE

Type:	Series Slot
UNITS:	FLOW VS. LENGTH
Description:	Reservoir outflow vs. either the tailwater elevation or the tailwater elevation increment
Information:	If the Tailwater Base Value is non-zero, the Tailwater Table gives values of incremental increase in Tailwater Elevation over the Base value. If the Tailwater Base Value is zero, the table simply gives the Tailwater Elevation values. The Tailwater Table should have increasing values of outflow. Tailwater Elevation is required to be a convex function of outflow. The preferred order of approximation is substitution, piece-wise, two-point line, tangent. Please see Object / Simulation documentation for further information.
I/O:	User-input

☞ TAILWATER TABLE LOOKUP LP PARAM

Type:	Table Slot
UNITS:	FLOW, FLOW, FLOW
Description:	LP Parameter approximation points for Outflow.
Information:	The Tailwater Table Lookup LP Param table values for outflow are used as approximation points indexing the Tailwater Table to determine tempTWLookupValue from the Tailwater Elevation column.
I/O:	User Input only
Links:	Not Linkable

☞ **TEMP TAILWATER LOOKUP VALUE**

Type: Series
Units: Length
Description: Numerical 2-D Approximation of increase in Tailwater Elevation due to flow as described in the Tailwater Table.

5.5.2.19.4 Opt Stage Flow Lookup Table

For more information on the Stage Flow Lookup Table method, click **HERE (Objects.pdf, Section 17.1.7.5)**. Tailwater Elevation is numerically approximated as a function of Stage and Flow as defined by the user in the Stage Flow Tailwater Table. The tables will be queried either using user-input points defined in the Tailwater Elevation LP Param table or using automatically calculated points, depending on method selection in the Tailwater Linearization Automation category. If the selected method in the Tailwater Linearization Automation category is “None”, then the user-input Tailwater LP Param table values are used. For the Range Input Automation method, the automatically developed points will be used.

Tailwater Elevation is approximated numerically (3-D approximation) as a function of Outflow and Tailwater Base Value based on the Stage Flow Tailwater Table. The Tailwater Elevation LP Param table Tailwater Base Value and flow values are used as approximation points indexing the Downstream Stage and Outflow columns of the Stage Flow Tailwater Table, respectively.

☞ **TAILWATER BASE VALUE**

Type: Series
Units: Length
Description: Described in the simulation documentation **HERE (Objects.pdf, Section 17.1.7.5)**

☞ **STAGE FLOW TAILWATER TABLE**

Type: Table Slot
Units: Flow vs. Length vs. Length
Description: Reservoir Outflow vs. Downstream Elevation (Tailwater Base Value) vs. Tailwater Elevation
Information: Data must be entered into the table in increasing blocks of the same Outflow value for the 3-dimensional table interpolation to work correctly. For every block of same Outflows in column 1, Downstream Stages should be listed in increasing order in column 2, and the corresponding Tailwater Elevations in column 3. Tailwater elevation is required to be a concave function of Outflow. The preferred order of approximation is substitution, piece-wise, two-point line, tangent. Internally, the Stage Flow Tailwater Table is rearranged to use the Stage as the primary index (column) with outflow as the secondary index (column). This rearranged table may be labeled as “Convolved Stage Flow Tailwater Table” in output messaging. Also, because of this rearrangement, it is desirable to repeat stage values for each outflow.
Defined by: User-input

☞ **TAILWATER ELEVATION LP PARAM**

Type: Table Slot
UNITS: LENGTH, FLOW, FLOW, FLOW
Description: Specifies the fixed tailwater base value point and the outflow points used to take the tangent, line and piecewise approximations for tailwater elevation linearization
Information: This table is used for linearization. The best Tailwater Base Value point to choose for tangent approximation would be an outflow equal to the expected value of outflow during the run; for the line approximation, the minimum and maximum values expected during the run; for piecewise approximation, the minimum and maximum values expected during the run plus additional intermediate values to more closely fit the tailwater curve.
Defined by: User-input unless the selected method in the Tailwater Linearization Automation category is not “None”.

☞ **TAIL WATER REFERENCE ELEVATION**

Type: Table Slot
UNITS: LENGTH
Description: Lowest Reservoir discharge Elevation when there are no backwater effects from a downstream pool (reservoir)
Information: Although this is part of the Stage Flow Lookup Table method (and its corresponding Opt method), **this slot does not influence optimization**. Please see Object / Simulation documentation for further information.
I/O: User-input

5.5.2.19.5 Opt Coefficients Table

This method sets up the physical Tailwater Elevation constraints using the same equation and coefficients used in the Coefficients Table tailwater method, **HERE (Objects.pdf, Section 17.1.7.7)**:

$$
\begin{aligned}
\text{Tailwater Elevation} = {} & \text{Constant Coeff}[t] + \text{Constant Coeff}[t-1] + \\
& \text{Outflow Coeff}[t] \times \text{flow} + \text{Outflow Coeff}[t-1] \times \text{Outflow}[t-1] + \\
& \text{TW Base Val Coeff}[t] \times \text{TailwaterBaseValueTemp}[t] + \\
& \text{TW Base Val Coeff}[t-1] \times \text{Tailwater BaseValue}[t-1] + \\
& \text{TW Elev Coeff}[t-1] \times \text{Tailwater Elevation}[t-1]
\end{aligned}
$$

☞ **TAILWATER BASE VALUE**

Type: Series
UNITS: LENGTH
Description: See Simulation description **HERE (Objects.pdf, Section 17.1.7.7)**.

☞ **TAILWATER COEFFICIENTS**

Type: Table
UNITS: MULTI
Description: See Simulation description **HERE (Objects.pdf, Section 17.1.7.7)**.
Information: If the Power Surface Approximation method is selected, then there are further restrictions on the coefficients that can be specified. With the Power Surface

Approximation, having terms for Outflow[t-1], Tailwater BaseVal[t] and Tailwater[t-1] are not allowed.

5.5.2.20 Tailwater Linearization Automation

This method category allows the user to choose whether they want to enter the approximation points used to the linearization approximations of tailwater or have the selection done automatically by riverware. The default method is none, which requires the points be input by the user. The automation method Range Input determines the points using the expected range of outflow values.

This category is dependent on 1) Optimization Head Calculation category selection of either Variable Head (not available in RPL Optimization) or None, and 2) Optimization Power category selection of Independent Linearizations, and 3) Optimization Tailwater category selection of either Opt Base Value Plus Lookup Table or Opt Stage Flow Lookup Table.

5.5.2.20.1 None

This method requires the input of the approximation points for tailwater linearization into the Tailwater LP param table.

5.5.2.20.2 Range Input Automation

The Range Input Automation method selects the linearization points for the linear approximations of tailwater in the Tailwater Elevation LP Param table. Tailwater is linearized as a 2- or 3-dimensional function depending on the method selected in the Optimization Tailwater category. If Opt Base Value Plus Lookup Table is selected tailwater is a 2-D function of flow. If Opt Stage Flow Lookup Table is selected tailwater is a 3-D function of flow and downstream stage. The same flow points are used for both the 2- and 3-D approximations. For the 3-D approximation the fixed point is required for the downstream stage. This point must still be input by the user.

SLOTS SPECIFIC TO THIS METHOD

☞ EXPECTED OUTFLOW RANGE

Type:	Table Slot
UNITS:	FLOW, FLOW
Information:	
Information:	Provide the high and low expected outflow values for the run.
Defined by:	User Input

The user inputs high and low expected values into the Expected Outflow Range table. The average of the values is calculated. The average is used as the tangent approximation point. The low and high values are used as the line approximation points. The low, average and high points are used as the piecewise points. The fixed point for three dimensional approximations is still required to be user input, but the validity of the point is checked during the automation process.

5.5.2.21 Optimization Reserves

This category depends on selecting either Plant Power Coefficient or Plant Efficiency Curve in the Power category. The methods in this category can be used to account for power reserve requirements in the optimization policy. There is no analogous simulation method associated with this category.

5.5.2.21.1 None

This is the default, do-nothing method. No new slots will be added, and no new variables or constraints will be added to the optimization problem.

5.5.2.21.2 Constraint Based Single Timestep

This method introduces a set of duplicate variables that represent the value of standard variables assuming full deployment of either upward or downward reserves. In the RPL Optimization Goal Set, users can then write parallel constraints using the duplicate reserve variables for any constraint that cannot be violated while deploying reserves. The purpose of the new variables and their parallel constraints is so that the only reserves that are credited to a hydropower project are those that can actually be reasonably deployed. In other words, it will not “count” reserves that could not be deployed without violating specified constraints. The new method also introduces new variables for upward and downward reserves, which allow the user to write policy on the reserves themselves, for example, project minimum reserve requirements or total system reserve requirements.

One significant assumption is made that the deployment of these reserves are temporary and only meant to cover a single timestep. It is assumed that in the event reserves are deployed in actual operations, adjustments would be made for later time periods, including reestablishing reserves. With this assumption, the consequences of deploying reserves at a reservoir are limited to that reservoir and that timestep; constraints on downstream reservoirs and later timesteps are not considered when calculating the reserves that can be credited at a given timestep.

The deployment variables are constrained similarly to the original variables with two exceptions. First, the mass balance constraints with other reservoirs or intermediate reaches are omitted (within RiverWare). Second, if there are any constraints that can be violated during deployment then these constraints can also be omitted (within the Optimization Goal Set). The deployment variables will typically use the same linear approximations as the original variables. For example, Pool Elevation with Up Reserve will be approximated using the Pool Elevation LP Param table slot.

Note that this method has no analogous simulation method. The slots introduced by this method can be referenced in the RPL optimization policy to constrain the solution, but they will not, generally, display values after the post-optimization rulebased simulation like the standard variables, such as Outflow or Pool Elevation. The user does have the option of writing customized post-optimization rules to set values in these slots based on user-specified logic or using the OptValue or OptValuePiecewise predefend functions.

SLOTS SPECIFIC TO THIS METHOD

This method will add the following series slots, all of which will be available as variables for the RPL optimization policy.

☞ UP RESERVE

Type: Series Slot
UNITS: POWER
Description: The upward reserves available; the amount the reservoir could increase generation on the given timestep
Information: Introducing this variable in the optimization policy forces a two-point line approximation to be used for both Power and Power with Up Reserve if Independent Linearizations is selected in the Optimization Power category. This may introduce additional approximation error.
Defined by: $\text{Up Reserve} = \text{Power with Up Reserve} - \text{Power}$

☞ DOWN RESERVE

Type: Series Slot
UNITS: POWER
Description: The downward reserves available; the amount the reservoir could decrease generation on the given timestep
Information: Introducing this variable forces a two-point line approximation to be used for both Power and Power with Down Reserve if Independent Linearizations is selected in the Optimization Power category. This may introduce additional approximation error.
Defined by: $\text{Down Reserve} = \text{Power} - \text{Power with Down Reserve}$

☞ POWER WITH UP RESERVE

Type: Series Slot
UNITS: POWER
Description: The average Power generated if full upward reserves were deployed
Information: This variable is defined by the Power approximation as a function of Turbine Release with Up Reserve and a combination of other variables. The set of other variables depends on which Power approximation is being applied. The Power approximation that is applied is dependent on the selected method in the Optimization Power category.
Defined by: $\text{Power with Up Reserve} = f(\text{Turbine Release with Up Reserve}, \ldots)$

☞ POWER WITH DOWN RESERVE

Type: Series Slot
UNITS: POWER
Description: The average Power generated if full downward reserves were deployed
Information: This variable is defined by the Power linear approximation as a function of Turbine Release with Down Reserve and a combination of other variables. The set of other variables depends on which Power approximation is being applied. The Power approximation that is applied is dependent on the selected method in the Optimization Power category.
Defined by: $\text{Power with Down Reserve} = f(\text{Turbine Release with Down Reserve}, \ldots)$

Turbine Release with Up Reserve

Type: Series Slot
UNITS: FLOW
Description: The Turbine Release at full deployment of upward reserves
Information: This variable is constrained to be less than or equal to Turbine Capacity with Up Reserve.
Defined by: $\text{Turbine Release with Up Reserve} = \text{Turbine Release} + \text{Turbine Change with Up Reserve}$

and

$$\text{Turbine Release with Up Reserve} \le \text{Turbine Capacity with Up Reserve}$$

Turbine Change with Up Reserve

Type: Series Slot
UNITS: FLOW
Description: The amount Turbine Release would change to fully deploy all upward reserves
Information: This variable is expected to always be non-negative, and thus the slot configuration lower bound should be set to 0.
Defined by: $\text{Turbine Release with Up Reserve} = \text{Turbine Release} + \text{Turbine Change with Up Reserve}$

Turbine Capacity with Up Reserve

Type: Series Slot
UNITS: FLOW
Description: The maximum Turbine Release possible given the full deployment of upward reserves
Information: This variable is defined by the Turbine Capacity approximation as a function of Operating Head with Up Reserve.
Defined by: $\text{Turbine Capacity with Up Reserve} = f(\text{Operating Head with Up Reserve})$

Operating Head with Up Reserve

Type: Series Slot
UNITS: FLOW
Description: The Operating Head at full deployment of upward reserves.
Information:
Defined by:

$$\text{Operating Head with Up Reserve} = \frac{\text{Pool Elevation with Up Reserve} + \text{Pool Elevation}[t-1]}{2} - \text{Tailwater Elevation with Up Reserve}$$

Spill with Up Reserve

Type: Series Slot
UNITS: FLOW
Description: The amount of total Spill at full deployment of upward reserves
Information: This could be equal to Spill, or it could be less than Spill if flow is allowed to be shifted from Spill to Turbine Release during deployment. This depends on how the RPL policy is formulated.
Defined by:

$$\text{Spill with Up Reserve} = \text{Unregulated Spill with Up Reserve} + \text{Regulated Spill with Up Reserve} + \text{Bypass with Up Reserve}$$

☞ UNREGULATED SPILL WITH UP RESERVE

Type: Series Slot
UNITS: FLOW
Description: The amount of Unregulated Spill at full deployment of upward reserves
Information: This quantity is defined by the Unregulated Spill linear approximation as a function of Storage with Up Reserve. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Unregulated Spill, then the value of this variable will automatically be set to zero.
Defined by: $\text{Unregulated Spill with Up Reserve} = f(\text{Storage with Up Reserve})$

☞ REGULATED SPILL WITH UP RESERVE

Type: Series Slot
UNITS: FLOW
Description: The amount of Regulated Spill at full deployment of upward reserves
Information: This could be equal to Regulated Spill, or it could be less than Regulated Spill if flow is allowed to be shifted from Spill to Turbine Release during deployment, or it could be greater than Regulated Spill if the reservoir has a minimum spill requirement as a percentage of total outflow. This depends on how the RPL policy is formulated. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Regulated Spill, then the value of this variable will automatically be set to zero, and the slot should not be referenced in the RPL optimization policy.
Defined by: $\text{Regulated Spill with Up Reserve} = \text{Regulated Spill} + \text{Regulated Spill Change with Up Reserve}$

and

$$\text{Regulated Spill with Up Reserve} \le \text{Regulated Spill Capacity with Up Reserve}$$

☞ REGULATED SPILL CHANGE WITH UP RESERVE

Type: Series Slot
UNITS: FLOW
Description: The amount Regulated Spill would change to fully deploy all upward reserves
Information: Often this variable will be negative (or 0, and thus the slot Upper Bound will often be set to 0); however it could be positive if the reservoir has a minimum spill requirement as a percentage of total outflow. It could be constrained to be 0, directly or indirectly, by RPL policy, or it could be allowed to be greater than or less than zero. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Regulated Spill, then the slot should not be referenced in the RPL optimization policy.
Defined by: $\text{Regulated Spill with Up Reserve} = \text{Regulated Spill} + \text{Regulated Spill Change with Up Reserve}$

☞ REGULATED SPILL CAPACITY WITH UP RESERVE

Type: Series Slot
UNITS: FLOW
Description: The maximum Regulated Spill given full deployment of upward reserves
Information: This variable is defined by the Regulated Spill Capacity approximation as a function of Storage with Up Reserve. This slot will get added regardless of which Spill method is selected. Users will not typically need to do anything with this slot.
Defined by: $\text{Regulated Spill Capacity with Up Reserve} = f(\text{Storage with Up Reserve})$

☞ BYPASS WITH UP RESERVE

Type: Series Slot
UNITS: FLOW
Description: The amount of Bypass at full deployment of upward reserves
Information: This could be equal to Bypass, or it could be less than Bypass if flow is allowed to be shifted from Bypass to Turbine Release during deployment, or it could be greater than Bypass if the reservoir has a minimum bypass requirement as a percentage of total outflow. This depends on how the RPL policy is formulated. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Bypass, then the value of this variable will automatically be set to zero, and the slot should not be referenced in the RPL optimization policy.
Defined by: $\text{Bypass with Up Reserve} = \text{Bypass} + \text{Bypass Change with Up Reserve}$

and

$$\text{Bypass with Up Reserve} \le \text{Bypass Capacity with Up Reserve}$$

☞ BYPASS CHANGE WITH UP RESERVE

Type: Series Slot
UNITS: FLOW
Description: The amount Bypass would change to fully deploy all upward reserves
Information: Often this variable will be negative (or 0, and thus the slot Upper Bound will often be set to 0); however it could be positive if the reservoir has a minimum bypass requirement as a percentage of total outflow. It could be constrained to be 0, directly or indirectly, by RPL policy, or it could be allowed to be greater than or less than zero. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Bypass, then the slot should not be referenced in the RPL optimization policy.
Defined by: $\text{Bypass with Up Reserve} = \text{Bypass} + \text{Bypass Change with Up Reserve}$

Bypass Capacity with Up Reserve

Type: Series Slot
Units: Flow
Description: The maximum Bypass given full deployment of upward reserves
Information: This variable is defined by the Bypass Capacity approximation as a function of Storage with Up Reserve. This slot will get added regardless of which Spill method is selected. Users will not typically need to do anything with this slot.
Defined by: $\text{Bypass Capacity with Up Reserve} = f(\text{Storage with Up Reserve})$

Outflow with Up Reserve

Type: Series Slot
Units: Flow
Description: The amount of total Outflow at full deployment of upward reserves
Information:
Defined by: $\text{Outflow with Up Reserve} = \text{Turbine Release with Up Reserve} + \text{Spill with Up Reserve}$

Storage with Up Reserve

Type: Series Slot
Units: Volume
Description: The end of timestep Storage if full upward reserves were deployed
Information: This variable definition differs from the standard Storage variable only in the Outflow term. The previous Storage and the Net Inflow are the same. Net Inflow represents all gains and losses, including Inflow, Hydrologic Inflow, Precipitation, Evaporation, etc.
Defined by: $\text{Storage with Up Reserve} = \text{Storage}[t-1] + (\text{NetInflow} - \text{Outflow with Up Reserve}) \times \text{Timestep}$

Pool Elevation with Up Reserve

Type: Series Slot
Units: Length
Description: The end of timestep Pool Elevation if full upward reserves were deployed
Information: This variable is defined by the Pool Elevation linear approximation as a function of Storage with Up Reserve.
Defined by: $\text{Pool Elevation with Up Reserve} = f(\text{Storage with Up Reserve})$

Tailwater Elevation with Up Reserve

Type: Series Slot
Units: Length
Description: The average Tailwater Elevation if full upward reserves were deployed
Information: This variable is defined by the Tailwater Elevation linear approximation as a function of Outflow with Up Reserve and the Tailwater Base Value (if applicable based on the selected Optimization Tailwater method). Tailwater Base Value is unaffected by the Reserves modeling and will be the same for both Tailwater Elevation and Tailwater Elevation with Up Reserves. The Tailwater Elevation linear

approximation that is applied is dependent on the selected method in the Optimization Tailwater category.

Defined by: Tailwater Elevation with Up Reserve = f(Outflow with Up Reserve, Tailwater Base Value)

☞ TEMP TAILWATER LOOKUP WITH UP RESERVE

Type: Series Slot

UNITS: LENGTH

Description: This slot only applies when the Base Value Plus Lookup Table method is selected in the Tailwater category. It represents the lookup value from the Tailwater Table as a function of Outflow with Up Reserve using the Tailwater linear approximation.

Information: This slot will get added regardless of which Tailwater method is selected but will not get used unless the Base Value Plus Lookup method is selected. Users should not need to do anything with this slot.

Defined by: Temp Tailwater Lookup with Up Reserve = f(Outflow with Up Reserve)

☞ PSA HEAD FACTOR WITH UP RESERVE

Type: Series Slot

UNITS: FLOW

Description: This slot only applies when the Power Surface Approximation method is selected in the Optimization Power category. It represents the component of Operating Head that is dependent on Storage with Up Reserve and Spill with Up Reserve. It is calculated using the same coefficients used for the Standard PSA Head Factor variable.

Information: This slot will get added regardless of which Optimization Power method is selected but will not get used unless the Power Surface Approximation method is selected. Users should not need to do anything with this slot.

Defined by: PSA Head Adjustment with Up Reserve = f(Storage with Up Reserve, Spill with Up Reserve)

☞ TURBINE RELEASE WITH DOWN RESERVE

Type: Series Slot

UNITS: FLOW

Description: The Turbine Release at full deployment of downward reserves

Information:

Defined by: Turbine Release with Down Reserve = Turbine Release + Turbine Change with Down Reserve

and

Turbine Release with Down Reserve ≤ Turbine Capacity with Down Reserve

Turbine Change with Down Reserve

Type: Series Slot
Units: Flow
Description: The amount Turbine Release would change to fully deploy all downward reserves
Information: This variable is expected to always be negative or zero, and thus the slot configuration Upper Bound should be set to 0.
Defined by:

$$\text{Turbine Release with Down Reserve} = \text{Turbine Release} + \text{Turbine Change with Down Reserve}$$

Turbine Capacity with Down Reserve

Type: Series Slot
Units: Flow
Description: The maximum Turbine Release possible given the full deployment of downward reserves
Information: This variable is defined by the Turbine Capacity approximation as a function of Operating Head with Down Reserve.
Defined by:

$$\text{Turbine Capacity with Down Reserve} = f(\text{Operating Head with Down Reserve})$$

Operating Head with Down Reserve

Type: Series Slot
Units: Flow
Description: The Operating Head at full deployment of downward reserves.
Information:
Defined by:

$$\text{Operating Head with Down Reserve} = \frac{\text{Pool Elevation with Down Reserve} + \text{Pool Elevation}[t-1]}{2} - \text{Tailwater Elevation with Down Reserve}$$

Spill with Down Reserve

Type: Series Slot
Units: Flow
Description: The amount of total Spill at full deployment of downward reserves
Information: This could be equal to Spill, or it could be greater than Spill if flow is allowed to be shifted from Turbine Release to Spill during deployment. This depends on how the RPL policy is formulated.
Defined by:

$$\text{Spill with Down Reserve} = \text{Unregulated Spill with Down Reserve} + \text{Regulated Spill with Down Reserve} + \text{Bypass with Down Reserve}$$

Unregulated Spill with Down Reserve

Type: Series Slot
Units: Flow
Description: The amount of Unregulated Spill at full deployment of downward reserves
Information: This quantity is defined by the Unregulated Spill linear approximation as a function of Storage with Down Reserve. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Unregulated Spill, then the value of this variable will automatically be set to zero.
Defined by:

$$\text{Unregulated Spill with Down Reserve} = f(\text{Storage with Down Reserve})$$

☞ REGULATED SPILL WITH DOWN RESERVE

Type:	Series Slot
UNITS:	FLOW
Description:	The amount of Regulated Spill at full deployment of downward reserves
Information:	This could be equal to Regulated Spill, or it could be greater than Regulated Spill if flow is allowed to be shifted from Turbine Release to Spill during deployment. This depends on how the RPL policy is formulated. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Regulated Spill, then the value of this variable will automatically be set to zero, and the slot should not be referenced in the RPL optimization policy.
Defined by:	Regulated Spill with Down Reserve = Regulated Spill + Regulated Spill Change with Down Reserve

and

Regulated Spill with Down Reserve ≤ Reguluated Spill Capacity with Down Reserve

☞ REGULATED SPILL CHANGE WITH DOWN RESERVE

Type:	Series Slot
UNITS:	FLOW
Description:	The amount Regulated Spill would change to fully deploy all downward reserves
Information:	This variable will typically by non-negative. It could be constrained to be 0, directly or indirectly, by RPL policy, or it could be allowed to be greater than zero. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Regulated Spill, then the slot should not be referenced in the RPL optimization policy.
Defined by:	Regulated Spill with Down Reserve = Regulated Spill + Regulated Spill Change with Down Reserve

☞ REGULATED SPILL CAPACITY WITH DOWN RESERVE

Type:	Series Slot
UNITS:	FLOW
Description:	The maximum Regulated Spill given full deployment of downward reserves
Information:	This variable is defined by the Regulated Spill Capacity approximation as a function of Storage with Down Reserve. This slot will get added regardless of which Spill method is selected. Users will not typically need to do anything with this slot
Defined by:	Regulated Spill Capacity with Down Reserve = f(Storage with Down Reserve)

☞ BYPASS WITH DOWN RESERVE

Type:	Series Slot
UNITS:	FLOW
Description:	The amount of Bypass at full deployment of downward reserves
Information:	This could be equal to Bypass, or it could be greater than Bypass if flow is allowed to be shifted from Turbine Release to Bypass during deployment. This depends on how the RPL policy is formulated. his slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Bypass, then the

value of this variable will automatically be set to zero, and the slot should not be referenced in the RPL optimization policy.

Defined by: $\text{Bypass with Down Reserve} = \text{Bypass} + \text{Bypass Change with Down Reserve}$

and

$\text{Bypass with Down Reserve} \leq \text{Bypass Capacity with Down Reserve}$

Bypass Change with Down Reserve

Type: Series Slot
Units: Flow
Description: The amount Bypass would change to fully deploy all downward reserves
Information: This variable will typically be non-negative. It could be constrained to be 0, directly or indirectly, by RPL policy, or it could be allowed to be greater than zero. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Bypass, then the slot should not be referenced in the RPL optimization policy.
Defined by: $\text{Bypass with Down Reserve} = \text{Bypass} + \text{Bypass Change with Down Reserve}$

Bypass Capacity with Down Reserve

Type: Series Slot
Units: Flow
Description: The maximum Bypass given full deployment of downward reserves
Information: This variable is defined by the Bypass Capacity approximation as a function of Storage with Down Reserve. This slot will get added regardless of which Spill method is selected. Users will not typically need to do anything with this slot
Defined by: $\text{Bypass Capacity with Down Reserve} = f(\text{Storage with Down Reserve})$

Outflow with Down Reserve

Type: Series Slot
Units: Flow
Description: The amount of total Outflow at full deployment of downward reserves
Information:
Defined by: $\text{Outflow with Down Reserve} = \text{Turbine Release with Down Reserve} + \text{Spill with Down Reserve}$

Storage with Down Reserve

Type: Series Slot
Units: Volume
Description: The end of timestep Storage if full downward reserves were deployed
Information: This variable definition differs from the standard Storage variable only in the Outflow term. The previous Storage and the Net Inflow are the same. Net Inflow represents all gains and losses, including Inflow, Hydrologic Inflow, Precipitation, Evaporation, etc.
Defined by:

$$\text{Storage with Down Reserve} = \text{Storage}[t-1] + (\text{NetInflow} - \text{Outflow with Down Reserve}) \times \text{Timestep}$$

☞ Pool Elevation with Down Reserve

Type: Series Slot
Units: Length
Description: The end of timestep Pool Elevation if full downward reserves were deployed
Information: This variable is defined by the Pool Elevation linear approximation as a function of Storage with Down Reserve.
Defined by: Pool Elevation with Down Reserve = f(Storage with Down Reserve)

☞ Tailwater Elevation with Down Reserve

Type: Series Slot
Units: Length
Description: The average Tailwater Elevation if full downward reserves were deployed
Information: This variable is defined by the Tailwater Elevation linear approximation as a function of Outflow with Down Reserve and the Tailwater Base Value (if applicable based on the selected Optimization Tailwater method). Tailwater Base Value is unaffected by the Reserves modeling and will be the same for both Tailwater Elevation and Tailwater Elevation with Up Reserves. The Tailwater Elevation linear approximation that is applied is dependent on the selected method in the Optimization Tailwater category.
Defined by: Tailwater Elevation with Down Reserve = f(Outflow with Down Reserve, Tailwater Base Value)

☞ Temp Tailwater Lookup with Down Reserve

Type: Series Slot
Units: Length
Description: This slot only applies when the Base Value Plus Lookup Table method is selected in the Tailwater category. It represents the lookup value from the Tailwater Table as a function of Outflow with Down Reserve using the Tailwater linear approximation.
Information: This slot will get added regardless of which Tailwater method is selected but will not get used unless the Base Value Plus Lookup method is selected. Users should not need to do anything with this slot.
Defined by: Temp Tailwater Lookup with Down Reserve = f(Outflow with Down Reserve)

☞ PSA Head Factor with Down Reserve

Type: Series Slot
Units: Flow
Description: This slot only applies when the Power Surface Approximation method is selected in the Optimization Power category. It represents the component of Operating Head that is dependent on Storage with Down Reserve and Spill with Up Reserve. It is calculated using the same coefficients used for the Standard PSA Head Factor variable.
Information: This slot will get added regardless of which Optimization Power method is selected but will not get used unless the Power Surface Approximation method is selected. Users should not need to do anything with this slot.
Defined by: PSA Head Adjustment with Down Reserve =
f(Storage with Down Reserve, Spill with Down Reserve)

Sample RPL Goal Set Implementation:

Within a RPL Goal set, any constraint that should not be violated when deploying reserves should include a constraint on the standard variable(s) as well as a duplicated constraint on the corresponding reserves variable(s). For example, assume that a reservoir has a maximum pool elevation that cannot be violated when reserves are deployed, and that maximum pool elevation is stored in a series slot on a data object called Res_Data.PE_Max. The RPL goal might look like:

```
REPEATED MAXIMIN
    FOR(DATETIME t IN @"Start Timestep" TO @"Finish Timestep") DO
        ADD CONSTRAINT Res.Pool Elevation[t] <= Res_Data.PE_Max[t]
        ADD CONSTRAINT Res.Pool Elevation with Down Reserve[t] <= Res_Data.PE_Max[t]
    END FOR
END REPEATED MAXIMIN
```

The reservoir might also have a target elevation constraint that is a lower priority and can be violated by the deployment of reserves. In this case, the target goal would only include a constraint on the Pool Elevation slot but not a constraint on the Pool Elevation with Down Reserve slot.

It is important to note that referencing any of the reserve slots in a RPL policy statement will typically draw a large number of additional variables and constraints into the optimization problem. For performance reasons, the user will probably not want these additional constraints to be added unless they are necessary. Assume, for example, that it is known prior to a run that a particular reservoir is not available to provide reserves during select hours. This may be indicated by setting an input value of 0 MW for those hours in a data object series slot called Res_Data.Down_Reserve_Max (similarly for Up Reserve). It would not be necessary to model the reserve versions of constraints for those time steps because it is already known that reserves will be zero. In this case the goal with the Pool Elevation constraints shown previously might be written as:

```
REPEATED MAXIMIN
    FOR(DATETIME t IN @"Start Timestep" TO @"Finish Timestep") DO
        ADD CONSTRAINT Res.Pool Elevation[t] <= Res_Data.PE_Max[t]
        IF(Res_Data.Down_Reserve_Max[t] != 0 MW)
            ADD CONSTRAINT Res.Pool Elevation with Down Reserve[t] <=
                                                        Res_Data.PE_Max[t]
        END IF
    END FOR
END REPEATED MAXIMIN
```

In this way, the constraint on Pool Elevation with Down Reserve would only be added when the reservoir is available to provide reserves (the max reserves are not constrained to zero).

Users could then write policy on the reserves to be carried at individual projects. Assume that the data object series slots Res_Data.Up_Reserve_Min and Res_Data.Up_Reserve_Max store the minimum upward reserves that the reservoir must carry and the maximum upward reserves the project can be credited respectively. A goal for the reservoir might be:

```
REPEATED MAXIMIN
    FOR(DATETIME t IN @”Start Timestep” TO @”Finish Timestep”) DO
        IF(Res_Data.Up_Reserve_Max[t] != 0 MW)
            ADD CONSTRAINT Res.Up Reserve[t] >= Res_Data.Up_Reserve_Min[t]
            ADD CONSTRAINT Res.Up Reserve[t] <= Res_Data.Up_Reserve_Max[t]
        END IF
    END FOR
END REPEATED MAXIMIN
```

These two constraints may not necessarily be included in the same goal. Note that the first constraint based on the minimum required reserves will constrain the physical operations of the reservoir in order to meet the required reserve obligation. The second constraint, based on the maximum credited reserves will not directly constraint the physical operations, rather it will only limit the amount of reserves that can be counted. In other words, the solution may operate the reservoir such that, according to physical limits and other specified constraints, the reservoir may have a larger amount of reserves available, but this constraint allows the user to say that only a limited amount will be counted.

Users may then write goals which sum reserves across multiple projects to meet a system reserve requirement.

```
REPEATED MAXIMIN
    FOR(DATETIME t IN @”Start Timestep” TO @”Finish Timestep”) DO
        ADD CONSTRAINT (FOR(OBJECT res IN ProjectsAvailableForUpReserve(t)) SUM
                                res.Up Reserve[t]
                            END FOR)
                                  >= System_Data.Up_Reserve_Min[t]
    END FOR
END REPEATED MAXIMIN
```

In this goal, ProjectsAvailableForUpReserve represents a user-defined function that returns a list of reservoirs that can carry reserves, again to prevent adding numerous constraints on the reserves version of variables unnecessarily. The details of this function would be model-specific.

5.5.2.22 Startup

This category depends on selecting the Unit Power Table method, **HERE (Section 5.5.2.17.5)**, and describes how the monetary cost associated with starting up or shutting down a unit (turbine) will be modeled. There are two methods in this category, one which does not model these costs (effectively assigning them a value of 0) and one which uses a table describing the combined costs for starting up and shutting down a unit.

5.5.2.22.1 None

This is the default, do-nothing method.

5.5.2.22.2 Unit Lumped Cost Method

For each unit, this method lumps the cost of startup and shutdown into one value. Slots are described **HERE (Objects.pdf, Section 17.1.38.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.5.2.23 Head Loss

This category depends on the Unit Power Table method, **HERE (Section 5.5.2.17.5)**, and contains methods for modeling additional head loss that occurs. This head loss may come from the configuration of the penstocks for bringing water to the turbines.

5.5.2.23.1 None

In this method, there is no additional head loss to be used in the power calculation. In terms of penstock head loss, this method should be selected if the penstocks for the units are independent and the penstock losses are typically incorporated in the power data. Thus the power data is specified in terms of operating head.

5.5.2.23.2 Shared Penstock Head Loss method

In this method, there is additional head loss that results because units share a common penstock. The operating head losses in the penstock depend on the total turbine release and are shared for all units. The net head is calculated by subtracting penstock losses from the operating head. The unit data and power must be specified in terms of unit Net Heads instead of Operating Head. Slots are described **HERE (Objects.pdf, Section 17.1.39.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.5.2.24 Cavitation

This category depends on selecting the Unit Power Table method, **HERE (Section 5.5.2.17.5)**, and contains methods for dealing with the problem of cavitation on turbines. Cavitation is the sudden formation and collapse of low-pressure bubbles in liquids by means of mechanical forces and this process can cause damage to turbines under certain operating conditions.

5.5.2.24.1 None

This is the default, do-nothing method.

5.5.2.24.2 Unit Head and Tailwater Based Regions

This method allows the user to specify the regions of operation in which cavitation does NOT occurs, so that these regions can be avoided. These regions can be dependent on both operating head and tailwater. Slots are described **HERE (Objects.pdf, Section 17.1.40.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.5.2.25 Avoidance Zones

This category depends on selecting the Unit Power Table method, **HERE (Section 5.5.2.17.5)**, and contains methods for modeling the existence of undesirable regions of operation for turbines. There are two methods in this category, one which does not model avoidance zones at all, and one which

5.5.2.25.1 None

This the default, do-nothing method; avoidance zones are not considered.

5.5.2.25.2 Unit Head Based Avoidance Zones

This method allows the user to specify a table that defines the conditions in which the turbines should not be operated. Slots are described **HERE (Objects.pdf, Section 17.1.41.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.5.2.26 Frequency Regulation

This category depends on selecting the Unit Power Table method, **HERE (Section 5.5.2.17.5)**, although in the future it might be enabled for other power methods. The frequency regulation methods model the provision of the frequency regulation ancillary service, that is, how the reservoir can be made available to flexibly follow a load demand within a specified range during a certain period in order to affect the frequency of the generated power.

5.5.2.26.1 None

This is the default, do-nothing method; no regulation is modeled.

5.5.2.26.2 Unit Frequency Regulation

NOTE: THIS METHOD IS NOT YET IMPLEMENTED

When frequency regulation is scheduled, it allows the unit to follow the real time load. Exactly what will happen in real time is unknowable. This results in two sets of values at scheduling time, nominally scheduled power and turbine release. It is uncertain if the real time operators will actually use the service. At present, we distinguish between the nominal "scheduled" power (and turbine release) that the regulation is allowed to depart from and the "expected" power generation (and turbine release) that will take place when regulation is allowed. Both are important. The scheduled power sets the baseline for regulation and should be communicated to the power dispatchers. The expected power and release are more useful for coordinating a plant with the rest of the system. Slots are described **HERE (Objects.pdf, Section 17.1.42.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.5.3 Modeling a Level Power Reservoir

Steps to follow in setting up a level power reservoir for optimization

1. Set up a running simulation model, **using methods that are compatible with optimization for Power, Tailwater, Spill, etc.**

2. Chose an Optimization Power method.

3. Choose an Optimization Head Computation Method. In Optimization, None must be chosen.

4. Select the Optimization Spill method corresponding to the method selected in the Spill category.

5. Fill in the LP Param tables related to Power, Turbine Capacity and Best Turbine Flow (if plant power is used). Alternatively, the Power Linearization Automation Method can also be chosen to automatically determine the values for the LP Param tables.

6. Select the Optimization Tailwater method corresponding to the method selected in Tailwater category. If Opt Base Value Plus Lookup Table or Opt Stage Flow Lookup Table is selected, fill in the Tailwater Elevation LP Param table. Alternatively, select a Tailwater Linearization Automation method (Range Input Automation is currently available).

7. If future value is needed, select a method in the Future Value category, followed by the Optimization Future Value category, and if desired, Cumul Stor Val Linearization Automation.

8. Selection of the remaining methods and linearizations can be done in any order. Pool elevation linearizations, Pool Elevation Linearization Automation, Evaporation and Precipitation, Optimize Evaporation Computation, Evaporation Linearization Automations, Bank Storage, Hydrologic Inflow, Energy In Storage, and Diversion from Reservoir. The appropriate data must be entered for each method.

5.6 Pumped Storage Reservoir

The Pumped Storage Reservoir object models a level reservoir where there are facilities that can release water to generate power but can also be pump water into the reservoir to increase storage. Additional information can be found **HERE (Objects.pdf, Section 21)**.

5.6.1 General Slots

General slots are always present on the object, regardless of selected methods. The following slots are provided on the reservoir when the optimization controller is selected.

☞ Canal Flow

Type: Agg Series
Units: Flow
Description: Flow into (out of) the reservoir from (to) a canal
Information:
Defined by: Explicit Optimization variable in the mass balance constraint

☞ Diversion

Type: Series Slot
Units: Flow
Description: Flow from the reservoir to a diverting object
Information:
Defined by: Explicit Optimization variable in the mass balance constraint

☞ Elevation Volume Table

Type: Table
Units: Length vs Volume
Description: Table relating elevation of the reservoir to volume stored in the reservoir
Information:
I/O: Input only
Defined by: Input only

☞ ENERGY

Type:	Agg Series Slot
UNITS:	ENERGY
Description:	Product of the power generated by flow through the turbines and the length of the timestep.
Information:	
Defined by:	Replaced by Power * Timestep Length
Description:	

☞ FLOW FROM PUMPED STORAGE

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow into the reservoir from a pumped storage reservoir
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint. This slot should be linked to Outflow on a Pumped Storage object. The Pumped Storage object constrains its Outflow.

☞ FLOW TO PUMPED STORAGE

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow out of the reservoir into a pumped storage reservoir
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint. This slot should be linked to Pumped Flow on a Pumped Storage object.

☞ INFLOW

Type:	MultiSlot
UNITS:	FLOW
Description:	Inflow into the reservoir from upstream
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint

Max Turbine Q

Type: Table Slot

Units: Length vs Flow

Description: Table relating Operating Head to maximum Turbine Capacity

Information: See power methods for more information

Operating Head

Type: Agg Series Slot

Units: Length

Description: Elevation difference between the average Pool Elevation and the average Tailwater Elevation during a timestep

Information:

Defined by: Replacement by (Pool Elevation(t) + Pool Elevation(t-1)) / 2 - Tailwater Elevation

Outflow

Type: Agg Series Slot

Units: Flow

Description: Outflow from the reservoir

Information:

Defined by: Explicit Optimization variable as Outflow = Turbine Release + Spill

Plant Power Table

Type: Table

Units: Length vs Flow vs Power

Description: 3D Table relating Operating Head, Turbine Release, and Power

Information: See power methods for more information

Pool Elevation

Type: Agg Series Slot

Units: Length

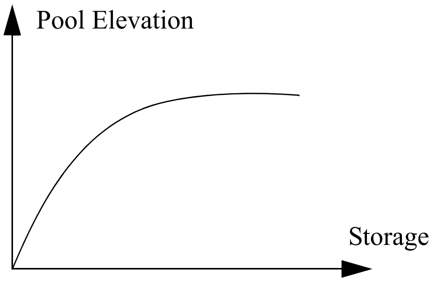
Description: Elevation of the water surface of the Reservoir

Information: When Pool Elevation is a part of the optimization problem, as it is in all conceivable RiverWare Optimization applications, this slot is numerically approximated as a function of Storage (Numerical 2-D Approximation). The relationship between Pool Elevation and Storage will come from the user-input Elevation Volume Table. The table will be queried either using user-input points defined in the Pool Elevation LP Param table.

Defined by: Numerical 2-D Approximation in terms of Storage, based upon the Elevation Volume Table. The Pool Elevation LP Param table values are used as approximation points indexing the Elevation Volume Table. The Elevation Volume Table should have increasing values of Pool Elevation and Storage. Storage is required to be a concave function of Pool Elevation. The preferred order of approximation is substitution, piece-wise, tangent, two-point line.

☞ POOL ELEVATION LP PARAM

Type:	Table Slot
UNITS:	VOLUME
Description:	Specifies the Storage points used to take the tangent, line and piecewise approximations for Pool Elevation linearization
Information:	This table is used for linearization unless Pool Elevation Linearization Automation category has selected Plant Automation. The best Storage point to choose for tangent approximation would be the expected storage expected during the run; for the line approximation, the expected maximum and minimum Storage; for piecewise approximation, use points that cover the full range of expected Storage during the run with intermediate points such that a piecewise linear curve reasonably approximates the actual curve.
Defined by:	User-input



Pool elevation storage relationship. The figure is not drawn to scale.

☞ POWER

Type:	Agg Series Slot
UNITS:	POWER
Description:	Power generated by flow through the turbines
Information:	
Defined by:	Numerical 3-D Approximation in terms of Operating Head and Turbine Release. Approximation is based on the Plant Power Table. The Power LP Param table contains a value for Operating Head used to index the Operating Head column of the Plant Power Table. This approximated value, therefore, reduces the Power to a function of Turbine Release at the given Operating Head. The flow values in the Power LP Param table are then used as approximation points indexing the Turbine Release column of the Plant Power Table. The Plant Power Table should have increasing values of Operating Head and Turbine Release. Power should be a concave function of Operating Head, but concavity is not strictly enforced; mild non-concave regions are permissible to allow for round-off error, etc. The preferred order of approximation is substitution, piece-wise, two-point line, tangent.

RETURN FLOW

Type: MultiSlot
UNITS: FLOW
Description: Flow returning from a diversion object
Information:
Defined by: Explicit Optimization variable in the mass balance constraint (see Storage)

SPILL

Type: Agg Series Slot
UNITS: FLOW
Description: Sum of the Bypass, Regulated Spill and Unregulated Spill
Information:
Defined by: Explicit Optimization variable as Spill = Bypass + Regulated Spill + Unregulated Spill

STORAGE

Type: Series Slot
UNITS: VOLUME
Description: Volume of water stored in the reservoir
Information:
Defined by: Explicit Optimization variable as Storage = Storage(t-1) + Precipitation Volume - Evaporation - Change in Bank Storage + timestep * (Inflow + Canal Flow + Flow TO Pumped Storage + Hydrologic Inflow Net + Return Flow - (Outflow + Diversion + Flow FROM Pumped Storage))

TAILWATER BASE VALUE

Type: Series Slot
UNITS: LENGTH
Description: Elevation of tailwater or base elevation used to compute elevation of tailwater
Information:
Defined by: Explicit Optimization variable should be input or linked. Related constraints can be found **HERE (Optimization.pdf, Section 5.6.2.20)**Tailwater Elevation Numerical Approximation discussion, or on other objects to which the slot is linked.

TAILWATER ELEVATION

Type: Agg Series Slot
UNITS: LENGTH
Description: Water surface elevation on the downstream side of the dam
Information:
Defined by: Various approaches dependent on the method selected in the Optimization Tailwater category.

☞ **TURBINE CAPACITY LP PARAM**

Type:	Table
UNITS:	LENGTH, LENGTH, LENGTH
Description:	LP Param table for turbine capacity
Information:	see power methods for more information

☞ **TURBINE CAPACITY**

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow capacity of the entire power plant's turbine(s)
Information:	
Defined by:	Numerical 2-D Approximation in terms of Operating Head, based upon a maximum turbine capacity table. This capacity table is determined in various ways according to the Power method.

☞ **TURBINE RELEASE**

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow through the turbines of a power reservoir
Information:	
Defined by:	Explicit Optimization variable as Turbine Release <= Power Plant Cap Fraction * Turbine Capacity

5.6.2 User Methods in Optimization

The following categories and methods are available for use in Optimization. Because of dependency relationships, you may not be able to see them in your model until you change other methods. In the discussion below, you will see the other methods that need to be selected to enable a given method to be used in your model. When building a model, you will wish to review this to ensure that required dependencies are satisfied so that the desired methods are available for use.

5.6.2.1 Bank Storage

Not all methods are functional in RPL optimization. Of the available methods, "None" and "CRSS Bank Storage" are supported.

5.6.2.1.1 None

Click **HERE (Objects.pdf, Section 21.1.24.1)** for more information.

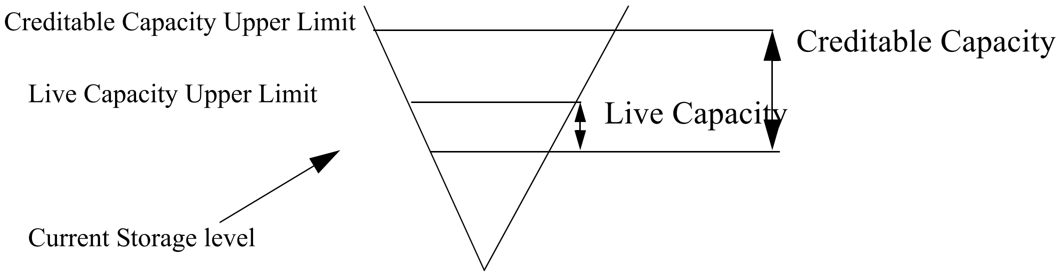
Input Bank Storage

5.6.2.1.2 CRSS Bank Storage

Click **HERE (Objects.pdf, Section 21.1.24.3)** for more information.

5.6.2.2 Creditable Capacity Available

The creditable capacity category conceptually represents the total amount of available storage space above the current storage in the reservoir. As the pool storage in the reservoir increases, the creditable capacity decreases.



5.6.2.2.1 None

If this method is selected creditable capacity is not calculated for the reservoir.

5.6.2.2.2 Constraint and Variable

If this method is selected an additional decision variable and physical constraint are added to the optimization problem:

SLOTS SPECIFIC TO THIS METHOD:

☞ **CREDITABLE CAPACITY**

Type:	Gassers
UNITS:	VOLUME
Description:	The creditable capacity is the total amount of storage space available above the current storage
Information:	
Defined by:	Explicit Optimization Variable included in the Optimization Problem as part of the constraint

$$\text{Max Storage} >= \text{Storage} + \text{Creditable Capacity.} \quad \text{(EQ 17)}$$

Max Storage for the reservoir is entered in the Upper Bound field of the Storage slot configuration dialog box.

5.6.2.3 Cumul Stor Val Linearization Automation

Appearance of this category is dependent on selecting the Opt Cumulative Storage Value Table method for the Optimization Future Value category (which in turn is dependent on the Cumulative Storage Value Table method being selected for the Future Value category).

This category allows the optimization to automate the creation of the Cumulative Storage Value Table, and the selection of linearization points and linearization method for the Cumulative Storage Value slot. Actual usage of these values occurs in the **HERE (Section 5.6.2.9)** Optimization Future Value category's Opt Cumulative Storage Value Table method.

5.6.2.3.1 None

If this method is selected, no automation will be performed.

5.6.2.3.2 Marginal Value to Table and Lin

This method uses information from the simulation slot Marginal Storage Value Table to generate the Cumul Stor Val Table, select linearization points, and choose a linearization method. The cumulative storage value in the Cumul Stor Val Table can be thought of as the summation of the marginal storage values from a storage of 0 to the current storage.

As an illustration of the automation procedure, consider the following Marginal Storage Value Table:

Marginal Storage Value Table:

Storage	Marginal Value
20	30
60	26
100	24

To parameterize the Cumul Stor Val Table, the automation proceeds as follows:

1) The first row receives a Storage value of 0.

Cumul Stor Val Table

Storage	Cumulative Value
0	

2) For each row i in the Marginal Storage Table, not including the last row, the average Storage (i.e. the midpoint) of row i and row i + 1 is assigned as Storage in each successive row of the Cumul Stor Val Table. For example, row 2 Storage equals the average of 20 and 60; row 3 Storage equals the average of 60 and 100.

Cumul Stor Val Table

Storage	Cumulative Value
0	
40	
80	

3) The last row of the Cumul Stor Val Table receive a Storage value equal to the upper bound storage associated with the Storage slot’s configuration.

Cumul Stor Val Table

Storage	Cumulative Value
0	
40	
80	
140	

4) Returning to the first row of the Cumul Storage Val Table, a Cumulative Value of 0 is assigned.

Cumul Stor Val Table

Storage	Cumulative Value
0	0
40	
80	
140	

5) For each successive row j = 2, 3, 4 in the Cumul Stor Val Table and corresponding row i = 1, 2, 3 in the Marginal Storage Value Table, the Cumulative Value equals the Storage from row j - 1 + (the change in Storage of row j from row j - 1) * the Marginal Value from row i. For example, row 2 Cumulative Value equals 1200 calculated from 0 + (40 - 0) * 30; row 3 Cumulative Value equals 2240 calculated from 1200 + (80 - 40) * 26; row 4 Cumulative Value equals 3680 calculated from 2240 + (140 - 80) * 24.

Cumul Stor Val Table:

Storage	Cumulative Value
0	0
40	1200
80	2240
140	3680

The Cumul Stor Val LP Param table is then built using Storage values from the Cumul Stor Val Table:

Cumul Stor Val LP Param

Tangent	Line	piecewise
	0	0
	140	40
		80
		140

SLOTS SPECIFIC TO THIS METHOD

There are no slots specific to this method as it uses the slots in the Opt Cumulative Storage Value Table method. However, for clarity in the discussion above, the slots are reshown here.

☞ CUMUL STOR VAL TABLE

Type:	Table Slot
UNITS:	VOLUME VS. VALUE
Description:	The estimated total economic value of water stored in the reservoir for discrete storage values.
Information:	
Defined by:	Automated procedure described above.

☞ CUMUL STOR VAL LP PARAM

Type:	Table Slot
UNITS:	VOLUME, VOLUME, VOLUME
Description:	Specifies the storage points used to take the tangent, line and piecewise approximations for Cumul Stor Val Table linearization
Information:	
Defined by:	Automated procedure described above.

☞ **MARGINAL STORAGE VALUE TABLE**

Type:	Table Slot
Units:	Storage vs. $ Value
Description:	Anticipated Storage versus worth of Cumulative Storage per unit energy
Information:	This table should be increasing in storage, and usually decreasing in marginal value
Defined by:	Required input

5.6.2.4 Diversion from Reservoir

Not all methods in this category are supported in RPL optimization. Of the available methods, “None” and “Available Flow Based Diversion” are supported; selection of any other method will result in an error during begin run.

5.6.2.4.1 None

Click **HERE (Objects.pdf, Section 19.1.24.1)** for more information.

5.6.2.4.2 Available Flow Based Diversion

Click **HERE (Objects.pdf, Section 19.1.24.2)** for more information.

5.6.2.5 Energy in Storage

In Optimization, currently “None” and “EIS Table Lookup” are supported.

5.6.2.5.1 None

No Energy in storage is considered.

5.6.2.5.2 EIS Table Lookup

With this method selected, Energy in Storage is considered as a function of Pool Elevation. Click **HERE (Objects.pdf, Section 21.1.6.2)** for more information on the simulation method. If the optimization problem uses Energy In Storage, either directly or indirectly, then this slot is added to the problem. Before being passed to the optimization solver, this slot is numerically approximated as a function of Pool Elevation (Numerical 2-D Approximation). The relationship between Pool Elevation and Energy In Storage will come from the user-input Energy In Storage Table. The table will be queried either using user-input points defined in the Energy In Storage LP Param table or using automatically calculated points, depending on method selection in the Pool Elevation Linearization Automation category. If the selected method in the Pool Elevation Linearization Automation is “none”, then the user-input Energy In Storage LP Param table values are used. For other Pool Elevation Linearization Automation methods (Initial Target Input, Min Difference or Range Input, Min Difference) the same points automatically developed for the Pool Elevation linearization will be used.

RELATED SLOTS

☞ ENERGY IN STORAGE

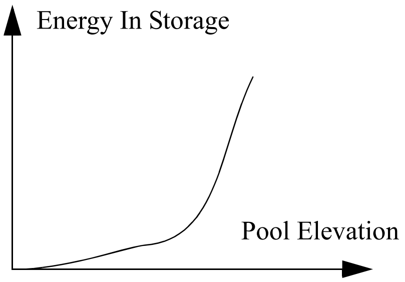
Type:	Series Slot
UNITS:	ENERGY VS. POWER
Description:	Energy in Storage in the reservoir
Information:	
Defined by:	Numerical 2-D Approximation in terms of Pool Elevation, based upon the Energy In Storage Table. The Energy In Storage LP Param table values are used as approximation points indexing the Energy In Storage Table. The Energy In Storage Table should have increasing values of Pool Elevation and Energy In Storage. Energy In Storage is required to be a convex function of Pool Elevation. The preferred order of approximation is substitution, piece-wise, tangent, two-point line.

☞ ENERGY IN STORAGE LP PARAM

Type:	Table Slot
UNITS:	LENGTH
Description:	Specifies the Pool Elevation points used to take the tangent, line and piecewise approximations for Energy In Storage linearization.
Information:	This table is used for linearization unless the Pool Elevation Linearization Automation category has a method selected other than “none”.
Defined by:	User-input

☞ ENERGY IN STORAGE TABLE

Type:	Table Slot
UNITS:	LENGTH VS. ENERGY
Description:	Table defining the relationship between Energy In Storage and Pool Elevation
Information:	
Defined by:	User-input



Energy in Storage pool elevation relationship. Not drawn to scale.

5.6.2.6 Optimization Evaporation

This category can be used to model evaporation and precipitation.

5.6.2.6.1 None

The Optimization Evaporation method "None" is the default method for this category. It does no calculations and requires that "None" be selected for the Evaporation and Precipitation category.

5.6.2.6.2 Opt Input Evaporation

This method is analogous to the Input Evaporation method in simulation and requires the Input Evaporation method to be selected for the Evaporation and Precipitation category. Evaporation Rate and Precipitation Rate are entered as a time series. Evaporation is calculated as a product of Evaporation Rate, Average Surface Area over the timestep and Timestep length. Similarly Precipitation Volume is calculated as the product of Precipitation Rate, Average Surface Area and Timestep length.

Note: The linearization of the Surface Area variable can result in a small approximation error in optimization for Evaporation and Precipitation Volume. This means there can be a small difference between the mass balances in the optimization solution and the post-optimization rulebased simulation when using this method. It is important to use care when setting the approximation points for Surface Area in order to reduce this approximation error. Refer to the information on the Surface Area LP Param slot below. Also caution should be used if applying this method at a monthly timestep. All rates in the optimization mass balance are converted to monthly volumes based on a 30-day month, regardless of the month. This will also produce a difference between the mass balances in the optimization solution and the post-optimization rulebased simulation for a *monthly* timestep.

SLOTS SPECIFIC TO THIS METHOD

☞ **ELEVATION AREA TABLE**

Type:	Table Slot
UNITS:	LENGTH VS. AREA
Description:	Represents the Elevation-Surface Area relationship
Information:	This table must be input. It is used to derive the Volume Area Table.
Defined by:	User-input

☞ **EVAPORATION**

Type:	Series Slot
UNITS:	VOLUME
Description:	The volume of water lost to evaporation over the timestep
Information:	If this slot contains user input, it is added directly to the mass balance constraint, otherwise it is defined by the expression below.
Defined by:	Either user-input or the following constraint:

$$\text{Evaporation} = \text{EvaporationRate} \times \frac{\text{SurfaceArea}(t-1) + \text{SurfaceArea}}{2} \times \text{Timestep}$$

☞ EVAPORATION RATE

Type: Series Slot
UNITS: VELOCITY
Description: The rate, in length per time, at which water is lost to evaporation at each timestep
Information: This slot can be set as user input. If it is not set as an input, and if Evaporation is not an input, this slot defaults to zero. If Evaporation is an input, this slot is not used.
Defined by: User-input or defaults to zero

☞ PRECIPITATION RATE

Type: Series Slot
UNITS: VELOCITY
Description: The rate, in length per time, at which water is gained from precipitation at each timestep
Information: This slot can be set as user input. If it is not set as an input, it defaults to zero.
Defined by: User-input or defaults to zero

☞ PRECIPITATION VOLUME

Type: Series Slot
UNITS: VOLUME
Description: The volume of water gained from precipitation over the timestep
Information: The Input Evaporation method will not allow this slot to be set as an input.
Defined by: Explicit constraint:

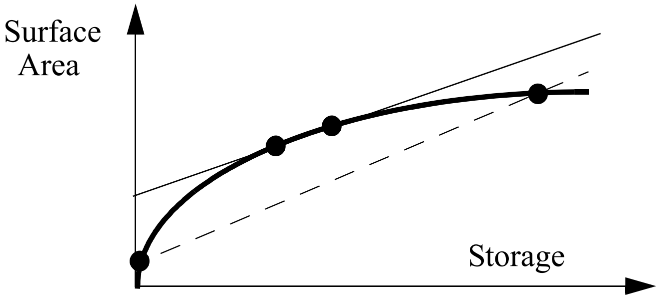
$$\text{Precipitation Volume} = \text{PrecipitationRate} \times \frac{\text{SurfaceArea}(t-1) + \text{SurfaceArea}}{2} \times \text{Timestep}$$

☞ SURFACE AREA

Type: Series Slot
UNITS: AREA
Description: The area of the water surface at the end of the timestep
Information: This slot is numerically approximated as a function of Storage (Numerical 2-D Approximation). The relationship between Pool Elevation and Storage comes from the automatically generated Volume Area Table. The table is queried using the user-input points defined in the Surface Area LP Param table.
Defined by: Numerical 2-D Approximation in terms of Storage, based upon the Volume Area Table. The Surface Area LP Param table values are used as approximation points indexing the Volume Area Table. The preferred order of approximation is substitution, piecewise, two-point line, tangent. Most often the two-point line (secant) approximation will be used.

☞ SURFACE AREA LP PARAM

Type:	Table Slot
UNITS:	VOLUME
Description:	Specifies the Storage points used to take the tangent, line and piecewise approximations for Surface Area linearization
Information:	The best Storage point to choose for tangent approximation would be the expected storage during the run; for line approximation, the expected maximum and minimum Storage; for piecewise approximation, use points that cover the full range of expected Storage during the run with intermediate points such that a piecewise linear curve reasonably approximates the actual curve.In most cases, the line (secant) approximation will be used. It is important to set these points carefully to minimize approximation error.
Defined by:	User-input



Surface Area vs. Storage relationship with two alternative line approximations.
The figure is not drawn to scale

The figure above represents two alternative selections of points for the line approximation in the Surface Area LP Param table. The linear approximation represented by the dashed line corresponds to the selection of points near the extremes of the Volume Area table. This approximation will tend to result in an under-estimation of Surface Area, and thus an under-estimation of Evaporation and Precipitation Volume. Evaporation losses in the post-optimization rulebased simulation would be greater than the losses approximated in the optimization solution. The linear approximation represented by the solid line corresponds to the selection of points closer together in the Volume Area Table, and is more similar to the Tangent approximation. This approximation would tend to result in an over estimation of Surface Area, and the losses due to Evaporation in the post-optimization rulebased simulation would be less than the approximation in the optimization solution.

☞ VOLUME AREA TABLE

Type:	Table Slot
UNITS:	VOLUME VS. AREA
Description:	Represents the Storage Volume-Surface Area relationship
Information:	This table is read-only and is automatically generated at the start of the run from the Elevation Area Table and the Elevation Volume table. The method starts by copying

the Elevation Area Table and then replaces the Pool Elevation Values with the corresponding Storage values. The Storage values are linearly interpolated based on the Elevation Volume Table

Defined by: Automatically generated

5.6.2.7 Evaporation Linearization Automation Category

This category allows the approximation points for surface area to be automatically generated. This category requires a method other than “None” to be selected for the Optimization Evaporation category.

5.6.2.7.1 None

This is the default method for this category. There is no approximation point automation. The user will be required to enter approximation points in the Surface Area LP Param slot manually.

5.6.2.7.2 Use Elevation Approximation Points

This method automates the approximation points in the Surface Area LP Param table slot. It copies the same storage values used in the Pool Elevation LP Param slot. If this method is selected it will overwrite any values that have been entered manually in the Surface Area LP Param slot. This method may provide a good starting point for establishing the Surface Area approximation points; however in some cases, it may be important for the user to adjust these points manually (see details **HERE (Section 5.6.2.6.2)** under Surface Area LP Param).

5.6.2.8 Future Value

This category and its methods are not dependent on other method selections.

5.6.2.8.1 None

Click **HERE (Objects.pdf, Section 17.1.15.1)** for more information.

5.6.2.8.2 Cumulative Storage Value Table

In RPL-Optimization, the Cumulative Storage Value is Numerically Approximated as described below.

Click **HERE (Objects.pdf, Section 21.1.15.2)** for more information.

SLOTS SPECIFIC TO THIS METHOD

Cumulative Storage Value

Type: Agg Series Slot
Units: $
Description: Represents the future energy value of the current storage
Information:
Defined by: 2-D approximation in terms of Anticipated Storage, based upon the Cumul Stor Val Table. The Cumul Stor Val LP Param table values (Storage) are used as approximation points indexing the Cumul Stor Val Table. The Cumul Stor Val Table should have increasing values of Storage and Cumulative Value. Cumulative Storage Value is required to be a concave function of Anticipated Storage. The preferred order of approximation is substitution, piece-wise, tangent, two-point line. The Cumul Stor Val Linearization Automation category’s Marginal Value to Table and Lin method can automate creation of the Cumul Stor Val LP Param table and the Cumul Stor Val Table.

Anticipated Storage

Type: Agg Series Slot
Units: Volume
Description: The combination of the actual storage plus water that would be expected to enter the reservoir after the Current Timestep but has not yet, due to lagging.
Information:
Defined by:

Cumul Stor Val Table

Type: Table Slot
Units: Volume vs. Value
Description: The estimated total economic value of water stored in the reservoir for discrete storage values.
Information:
Defined by: User-input or by automated procedure if Cumul Stor Val Linearization Automation category has selected the Marginal Value to Table and Lin method.

Marginal Storage Value Table

Type: Table Slot
Units: Storage vs. $ Value
Description: Anticipated Storage versus Cumulative Storage Value per unit energy
Information: This table should be increasing in storage, and logically decreasing in marginal value
Defined by: Required input

☞ **SPILL COST**

Type:	Agg Series Slot
UNITS:	$
Description:	
Information:	
Defined by:	

5.6.2.9 Optimization Future Value

This category allows the optimization to provide slots relating to the future value of water. Its appearance is dependent on the Cumulative Storage Value Table method being selected for the Future Value category.

5.6.2.9.1 None

If this method is selected, the future value slots will not be visible, and no linearization will be attempted.

5.6.2.9.2 Opt Cumulative Storage Value Table

If this method is selected, the following slots will be visible, and linearization will be allowed.

SLOTS SPECIFIC TO THIS METHOD

☞ **CUMUL STOR VAL LP PARAM**

Type:	Table Slot
UNITS:	VOLUME, VOLUME, VOLUME
Description:	Specifies the storage points used to take the tangent, line and piecewise approximations for Cumul Stor Val Table linearization
Information:	
Defined by:	User-input or by automated procedure if Cumul Stor Val Linearization Automation category has selected the Marginal Value to Table and Lin method.

5.6.2.10 Hydrologic Inflow

If any method in this category is selected, hydrologic inflow is included in the reservoir mass balance. Optimization assumes hydrologic inflow to be known (data) and it is not solved for by the reservoir regardless of the method selected. Click **HERE (Objects.pdf, Section 21.1.18)** for more information.

5.6.2.11 Live Capacity Available

The live capacity category conceptually represents the amount of available storage space between the current storage in the reservoir and a user defined maximum. As the pool storage in the reservoir increases, the live capacity decreases.

5.6.2.11.1 None

If this method is selected live capacity is not calculated for the reservoir.

5.6.2.11.2 Constraint and Variable

If this method is selected an additional decision variable and physical constraint are added to the optimization problem. See the figure in the Creditable Capacity Category.

☞ **LIVE CAPACITY**

Type:	Agg Series Slot
UNITS:	VOLUME
Description:	The amount of storage space available between the current storage and a user defined upper limit. This upper limit is entered in the Upper Bound field of the Live Capacity slot configuration dialogue box.
Defined by:	Explicit constraint

$$\text{Max Live Capacity} >= \text{Storage} + \text{Live Capacity} \quad \textbf{(EQ 18)}$$

5.6.2.12 Optimization Spill

The Optimization Spill methods determine how spill is calculated for the reservoir and generates physical constraints that correspond to the selected methods.

This category is dependent on selection of the Independent Linearizations method for the Optimization Power category. However, the method selected in the Optimization Spill category must match with the corresponding non-optimization method in the Spill category.

Spill is an Optimization decision variable. The following constraint is always generated for a reservoir:

$$\text{Outflow} = \text{Turbine release (or Release)} + \text{Spill} \quad \textbf{(EQ 19)}$$

The spill methods generate values applicable to an additional constraint:

$$\text{Spill} = \text{Regulated Spill} + \text{Unregulated Spill} + \text{Bypass}, \quad \textbf{(EQ 20)}$$

where some of these terms may be omitted if they do not apply to the selected spill method.

Depending on the method selected, some of the following slots will be added. Other slots will be used from the non-optimization method selected. Please click **HERE (Objects.pdf, Section 21.1.10)** for details on non-optimization Spill methods.

As applicable, the following constraints are also added:

$$\text{Bypass} <= \text{Bypass Capacity} \quad \textbf{(EQ 21)}$$

$$\text{Regulated Spill} <= \text{Regulated Spill Capacity} \quad \textbf{(EQ 22)}$$

$$\text{Unregulated Spill} = \text{Unregulated Spill Capacity} \quad \textbf{(EQ 23)}$$

☞ BYPASS CAPACITY

Type:	Series Slot
UNITS:	FLOW
Description:	Bypass capacity
Information:	
Defined by:	Numerical 2-D Approximation in terms of storage, based upon the Bypass Capacity Table.

☞ BYPASS CAPACITY TABLE

Type:	Table Slot
UNITS:	VOLUME VS. FLOW
Description:	Storage vs. corresponding maximum bypass spill values
Information:	
Defined by:	Internally developed based on Bypass Table and Elevation Volume Table relationships. For each pool elevation in the Bypass Table, the Bypass Capacity Table has a row relating Storage to Bypass Capacity. The Pool Elevations are converted to Storage using the Elevation Volume Table.

☞ REGULATED SPILL CAPACITY

Type:	Series Slot
UNITS:	FLOW
Description:	Regulated spill capacity
Information:	
Defined by:	Numerical 2-D Approximation in terms of storage, based upon the Regulated Spill Capacity Table.

☞ REGULATED SPILL CAPACITY TABLE

Type:	Table Slot
UNITS:	VOLUME VS. FLOW
Description:	Storage vs. corresponding maximum regulated spill values
Information:	
Defined by:	Internally developed based on Regulated Spill Table and Elevation Volume Table relationships. For each pool elevation in the Regulated Spill Table, the Regulated Spill Capacity Table has a row relating Storage to Regulated Spill Capacity. The Pool Elevations are converted to Storage using the Elevation Volume Table.

☞ REGULATED SPILL OR BYPASS LP PARAM

Type:	Table Slot
UNITS:	VOLUME
Description:	Specifies the Storage points use to take the tangent, line and piecewise approximations for Regulated Spill Capacity linearization and Bypass Spill Capacity linearization.
Information:	
Defined by:	User-input

☞ UNREGULATED SPILL LINEARIZATION TABLE

Type:	Table Slot
UNITS:	VOLUME VS. FLOW
Description:	Storage vs. corresponding unregulated spill values
Information:	
Defined by:	Internally developed based on Unregulated Spill Table and Elevation Volume Table relationships. For each pool elevation in the Unregulated Spill Table, the Unregulated Spill Capacity Table has a row relating Storage to Unregulated Spill Capacity. The Pool Elevations are converted to Storage using the Elevation Volume Table.

☞ UNREGULATED SPILL LP PARAM

Type:	Table Slot
UNITS:	VOLUME
Description:	Specifies the Storage points use to take the tangent, line and piecewise approximations for Unregulated Spill Linearization Table linearization
Information:	
Defined by:	User-input

5.6.2.12.1 None

If this method is selected the slot bounds in the slot configuration dialog are set to zero. No additional constraints are generated.

5.6.2.12.2 Opt Monthly Spill

This method is not functional in RPL Optimization.

This method sets the lower and upper bounds on spill. The lower bound is set to zero and the default upper bound is set to a very big number (9,999,999 cms). The default upper bound can be revised in the Spill slot configuration, Upper Bound parameter. No additional constraints are generated beyond these bounds.

5.6.2.12.3 Opt Unregulated

If this method is selected only unregulated spill is considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Unregulated Spill} \tag{EQ 24}$$

5.6.2.12.4 Opt Regulated

When this method is selected only regulated spill is considered. The lower bound on spill is set to zero and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} \tag{EQ 25}$$

5.6.2.12.5 Opt Regulated and Unregulated

When this method is selected unregulated and regulated spill are considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} + \text{Unregulated Spill} \tag{EQ 26}$$

5.6.2.12.6 Opt Regulated and Bypass

When this method is selected only regulated and bypass spill are considered and the lower bound on spill is set to zero and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} + \text{Bypass} \tag{EQ 27}$$

5.6.2.12.7 Opt Regulated, Bypass and Unregulated

When this method is selected unregulated, regulated and bypass spill are considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} + \text{Unregulated Spill} + \text{Bypass} \tag{EQ 28}$$

5.6.2.12.8 Opt Bypass, Regulated and Unregulated

When this method is selected unregulated, regulated and bypass spill are considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Bypass} + \text{Regulated Spill} + \text{Unregulated Spill} \tag{EQ 29}$$

5.6.2.13 Pool Elevation Linearization Automation

This category is no longer supported in RPL optimization.

5.6.2.14 Optimization Head Computation

The methods in this category are no longer supported in RPL optimization. Appearance of this category is dependent on selection of the Independent Linearizations method for the Optimization Power category.

5.6.2.15 Optimization Power

The Optimization Power category allows the user to select which type of linearizations will be used for linearizing various slots. In the RPL Optimization, only the Independent Linearizations method is functional; the Lambda Method is no longer supported. For Independent Linearizations all slots are linearized separately according to the user specified methods. This category has no dependencies.

5.6.2.15.1 None

This is the default method. It is an error to have this method selected for Optimization. No slots are added for this method.

5.6.2.15.2 Independent Linearizations

The Independent Linearizations method linearizes all the variables separately according to the user selected methods. The variables that need to be linearized vary greatly depending on the other methods selected. See the various Categories and Linearization Approaches for details.

5.6.2.15.3 Power Coefficient

The Power Coefficient method models Power at each time step as Turbine Release multiplied by a Power Coefficient Estimate.

SLOTS SPECIFIC TO THIS METHOD

☞ **POWER COEFFICENT ESTIMATE**

Type:	Series Slot
UNITS:	POWERPERFLOW
Description:	This represents the estimated Power Coefficient that gets used in the Optimization definition of Power.
Information:	The Power Coefficient Estimate is a required input for the run. It can either be input directly (manually or by DMI), or it could be set by an initialization rule. For example, an initialization rule could set the Power Coefficient Estimate based on a “Seed” (Slot Cache) value. If the Power Coefficient method is selected, and Power Coefficient Estimate is not an input for all time steps, then the run will abort with an error message.
Defined by:	Input

Power at each time step is then defined as:

$$\text{Power} = \text{Turbine Release} \times \text{Power Coefficient Estimate}$$

5.6.2.15.4 Power Surface Approximation

This method allows for the modeling of dynamic Operating Head to be incorporated into Optimization Power modeling. It provides a significant improvement in the Power approximation error over the linearization using the Power LP Param table, which assumes a constant Operating Head over the entire run period. The improvement is especially significant for projects that exhibit a wide range of Operating Head over the course of the run, particularly if the optimization policy is trying to maximize Power or set Power greater than or equal to a value.

The method introduces a new variable, PSA Head Factor, which represents the weighted contributions of Storage, Spill and Tailwater Base Value to Operating Head. The method generates a set of planes, the Power Surface, to constrain power as a function of Turbine Release and the PSA Head Factor.

The Power Surface can be thought of as pavement over a warped bridge that is level on one end and sloped and higher on the other end. The bridge ends represent zero Turbine Release and maximum Turbine Release respectively. In addition, the bridge is bowed concave across the center line everywhere except the level (zero Turbine Release) end. This bowing reflects Power as a function of the Head Factor for a fixed value of Turbine Release. The edges of the bridge represent Power as a function of Turbine Release for the low and high values of the Head Factor. Additional points defining a grid on the surface correspond to intermediate values of Turbine Release and the Head Factor. The user can define dimensions and points used for this grid. A piecewise linear surface is composed of cutting planes with each plane defining one triangle on the surface. A surface underneath the bridge composed of two planes defines a lower bound on Power given Turbine Release and Head Factor values.

Because only two planes can be defined for the lower bounds on power, additional approximation error can be introduced if the optimization policy has an incentive to minimize power. In these cases the optimization solution can have an incentive to under-approximate Power for a given Turbine Release, and thus the Post-optimization Rulebased Simulation will calculate a larger Power value than the optimization value for Power.

Slots Specific to this Method

The user enters data into three slots:

PSA Sample Input,

PSA Head Factor Grid Lines

PSA Turbine Release Grid Lines

The remaining table slots associated with the method are automatically populated by RiverWare. A description of all of the slots associated with this method is given below.

☞ PSA Sample Input

Type:	Table Slot
Units:	Flow, Volume, Flow, Length
Description:	This table contains user-specified sample values for Turbine Release, Storage, Total Spill (if applicable) and Tailwater Base Value (if applicable) that are used by RiverWare to compute the PSA Head Factor Weights. The four parameters in this table represent all of the parameters that contribute to Operating Head. In other words, if a value is known for each of these parameters, it is possible to calculate the corresponding Operating Head.
Information:	The values in this table should span the full possible range for each parameter for the given run. For example, the Turbine Release column should contain a value of 0, the highest possible turbine release from the reservoir, and any other intermediate values specified by the user. Specifying tighter upper and lower bounds for each parameter will improve the approximation, but it is important that all possible values for each variable within the run are spanned by the range for that parameter in the table. The user can determine how many rows to include in the table. It is expected that typically, 3-4 values will be specified for each parameter. An example is shown

below. The Spill column should only contain values if a method other than None has been selected in the Spill category. Otherwise the Spill column should be left empty (will display “NaN”). The Tailwater Base Value column should only contain values if the reservoir’s Tailwater Base Value slot is linked to the Pool Elevation slot of a downstream reservoir (i.e. if the Downstream Reservoir Pool Elevation contributes to the calculation of Operating Head). Otherwise the Tailwater Base Value column should be left empty.

Defined by: User Input

Turbine Release	Storage	Spill (Total)	Tailwater Base Value
0	0	0	500
100	1000	50	510
200	2000	100	520

☞ PSA HEAD FACTOR GRID LINES

Type: Scalar Slot
UNITS: NONE
Description: The number of PSA Head Factor values for each Turbine Release value used when calculating the PSA Grid Points
Information: This must be an integer greater than or equal to 2. The value should typically range from 2 to 4. A larger number will improve the approximation but will increase run time.
Defined by: User Input

☞ PSA TURBINE RELEASE GRID LINES

Type: Scalar Slot
UNITS: NONE
Description: The number of Turbine Release values for each PSA Head Factor value used when calculating the PSA Grid Points
Information: This must be an integer greater than or equal to 2. The value should typically range from 2 to 4. A larger number will improve the approximation but will increase run time.
Defined by: User Input

PSA SAMPLE OUTPUT

Type:	Table Slot
UNITS:	FLOW, VOLUME, FLOW, VOLUME, POWER
Description:	A table containing all permutations of the values entered in the PSA Sample Input table slot along with the Power calculated for each combination of Sample Input values. This sample output is used to calculate the PSA Head Factor Weights.
Information:	RiverWare will populate this table at the beginning of the run. For the PSA Sample Input table shown above, with three values in each of the four columns, this table would contain 81 rows, one for each permutation. For a given combination of Turbine Release, Storage, Spill and Tailwater Base Value, RiverWare can calculate the corresponding Power. The third column will contain the Downstream Storage values corresponding to the Tailwater Base Value entries in the PSA Sample Input slot.
Defined by:	Populated by RiverWare at the beginning of the run

For each row *i* in the table:

$$\mathrm{SamplePower}_i = f(\mathrm{SampleTurbineRelease}_i, \mathrm{SampleStorage}_i, \mathrm{SampleSpill}_i, \mathrm{SampleTailwaterBaseValue}_i)$$

The calculation of SamplePower depends on the selected Power and Tailwater methods.

PSA HEAD FACTOR WEIGHTS

Type:	Table Slot
UNITS:	PERTIME, NONE, PERTIME
Description:	The weights for Storage, Spill and Downstream Storage used when calculating PSA Head Factor
Information:	This is a 1 x 3 table with a single weight for Storage, Spill and Downstream Storage. The weights are calculated by RiverWare by performing a regression on the values in the PSA Sample Outputs table slot. If one of the parameters is not used (contains NaN for all rows in the PSA Sample Input table), the corresponding weight in this table slot will be zero.
Defined by:	Calculated by regression at the beginning of the run

PSA HEAD FACTOR

Type:	Series Slot
UNITS:	FLOW
Description:	This slot is a variable in the optimization solution and represents the weighted contribution of Storage, Spill and Downstream Storage to Operating Head and thus power. It can be thought of as a storage analog of Operating Head.
Information:	This variable is used in the automatically generated constraints that define Power. It will not, generally display any values, but the user can write a post-optimization rule that calculates the value of this variable at each time step using the equation shown below and the values in the PSA Head Factor Weights table slot.
Defined by:	$\text{PSA Head Factor} = \alpha \cdot \text{Storage} + \beta \cdot \text{Spill} + \gamma \cdot \text{Downstream Storage}$

The coefficients α, β and γ come from the PSA Head Factor Weights table slot.

☞ PSA GRID POINTS

Type:	Table Slot
UNITS:	FLOW, VOLUME, FLOW, VOLUME, POWER, FLOW
Description:	Points used to calculate the planes for the PSA Max Constraints and PSA Min Constraints, one row for each point with columns for Turbine Release, Storage, Spill, Downstream Storage, Power and PSA Head Factor.
Information:	The number of rows (points) in this table equals the product of the number of lines in the PSA Head Factor Grid Lines and PSA Turbine Release Grid Lines scalar slots. For each grid point, RiverWare calculates the value of Power and the PSA Head Factor corresponding to the parameter values in that row.
Defined by:	Populated by RiverWare at the beginning of the run

Turbine Release Column: Smallest and largest values from PSA Sample Input and evenly spaced intermediate values corresponding to the number of lines in the PSA Turbine Release Grid Lines scalar slot

Storage, Spill and Downstream Storage Columns: Smallest and largest values from PSA Sample Input and evenly spaced intermediate values corresponding to the number of lines in the PSA Head Factor Grid Lines scalar slot

Power Column: Determined by selected Power method, for each row *i* in the table

$$\text{Power}_i = f(\text{TurbineRelease}_i, \text{Storage}_i, \text{Spill}_i, \text{DownstreamStorage}_i)$$

Head Factor Column: $\text{HeadFactor}_i = \alpha \cdot \text{Storage}_i + \beta \cdot \text{Spill}_i + \gamma \cdot \text{DownstreamStorage}_i$

The coefficients α, β and γ come from the PSA Head Factor Weights table slot.

☞ PSA MAX CONSTRAINTS

Type:	Table Slot
UNITS:	FLOW, FLOW, FLOW, FLOW, FLOW, FLOW, POWERPERFLOW, POWERPERFLOW, POWER
Description:	This table contains the planes that define the upper bounds for power as a function of Turbine Release and PSA Head Factor, one row for each plane. Power is constrained to be less than or equal to the planes defined in this table. The first six columns contain three pairs of Turbine Release and PSA Head Factor values that define three points on the plane. The remaining three columns contain the corresponding Turbine Release Coefficient, Head Factor Coefficient and Constant Term that define the same plane and are used in the actual constraint expressions.
Information:	The points used to define the planes come from the Turbine Release and Head Factor points in the PSA Grid Points table slot. RiverWare will not use all of the possible combinations of Turbine Release and Head Factor points but rather will only use the combinations necessary to define the Power Surface.
Defined by:	Populated by RiverWare at the beginning of the run

For each row *i* in the PSA Max Constraints table slot a constraint is added to the optimization problem:

$$\text{Power} \le u_i \cdot \text{Turbine Release} + v_i \cdot \text{PSA Head Factor} + w_i$$

The coefficients u_i, v_i and w_i are the Turbine Release Coefficient, Head Factor Coefficient and Constant Term from row *i* of the PSA Max Constraints table. Turbine Release and PSA Head Factor are the actual variables in the optimization problem.

☞ PSA MIN CONSTRAINTS

Type: Table Slot
UNITS: FLOW, FLOW, FLOW, FLOW, FLOW, FLOW, POWERPERFLOW, POWERPERFLOW, POWER
Description: This table contains the planes that define the lower bounds for power as a function of Turbine Release and PSA Head Factor, one row for each plane. Power is constrained to be greater than or equal to the planes defined in this table. The first six columns contain three pairs of Turbine Release and PSA Head Factor values that define three points on the plane. The remaining three columns contain the corresponding Turbine Release Coefficient, Head Factor Coefficient and Constant Term that define the same plane and are used in the actual constraint expressions
Information: This table will always contain two rows. Only two planes can be defined for the lower bounds on power.
Defined by: Populated by RiverWare at the beginning of the run

For each row *i* in the PSA Min Constraints table slot a constraint is added to the optimization problem:

$$\text{Power} \ge u_i \cdot \text{Turbine Release} + v_i \cdot \text{PSA Head Factor} + w_i$$

The coefficients u_i, v_i and w_i are the Turbine Release Coefficient, Head Factor Coefficient and Constant Term from row *i* of the PSA Min Constraints table. Turbine Release and PSA Head Factor are the actual variables in the optimization problem.

METHOD SUMMARY

Three slots require input from the user. Descriptions of these slots are provided above.

PSA Sample Input

PSA Turbine Release Grid Points

PSA Head Factor Grid Points

Then at the start of the run RiverWare carries out the following steps:

- Populate the PSA Sample Output Table with all permutations of parameters in the PSA Sample Input table slot
- Calculate the PSA Head Factor Weights using a regression based on values in the PSA Sample Output table slot
- Add defining constraints for PSA Head Factor using the PSA Head Factor Weights
- Populate the PSA Grid Points table slot based on the number of grid lines specified by the user
- Generate upper bound planes using the PSA Grid Points and add the corresponding constraints on Power to the Optimization problem

- Generate lower bound planes using the PSA Grid Points and add the corresponding constraints on Power to the Optimization problem

METHOD DETAILS

This section provides details of the calculations carried out by RiverWare to generate the constraints that define the Power Surface. This material is included for reference only.

1. RiverWare populates the PSA Sample Output table slot. All permutations of values in the PSA Sample Input table slot are determined, and a row for each is added to the PSA Sample Output table slot. RiverWare may make some adjustments to prevent infeasible combinations. For example, the highest Turbine Release value may not be possible for all combinations of Storage, Spill and Tailwater Base Value. RiverWare will adjust the Turbine Release to use the maximum turbine release for the resulting Operating Head. The table is organized in blocks of constant Turbine Release, with the lowest Turbine Release block first and then increasing Turbine Release. The values in the remaining columns are ordered corresponding to increasing Operating Head. Storage is increasing. Spill and Tailwater Base Value (or Downstream Storage) are decreasing. For each combination of values (each row), RiverWare calculates the corresponding Operating Head and Power. The Power value is added to each row in the PSA Sample Output table. The details of these calculations are shown here for a given row *i* in the PSA Sample Output table. Note that Power_i is the only calculated quantity displayed in the table. Values from the intermediate calculations shown below are not displayed in the table.

$$\text{PoolElevation}_i = f(\text{Storage}_i)$$

Pool Elevation is calculated by interpolation on the reservoir Elevation Volume Table.

If Tailwater Base Value is linked to Pool Elevation of a downstream reservoir:

$$\text{TailwaterBaseValue}_i = \text{DownstreamPoolElevation}_i = f(\text{DownstreamStorage}_i)$$

Tailwater Base Value is calculated by interpolation on the downstream reservoir Elevation Volume Table. Tailwater Base Value is only calculated from the Downstream Storage if either Base Value Plus Lookup Table, Stage Flow Lookup Table or Coefficients Table is the selected Tailwater method *and* Tailwater Base Value is linked to the Pool Elevation of the downstream reservoir. Otherwise Tailwater Base Value is unused, and thus Operating Head and Power are not dependent on Downstream Storage.

$$\text{Outflow}_i = \text{TurbineRelease}_i + \text{Spill}_i$$

Spill will only be non-zero if a method other than None is selected in the Spill category.

$$\text{TailwaterElevation}_i = f(\text{Outflow}_i, \text{TailwaterBaseValue}_i)$$

The tailwater calculation is dependent on the selected method in the Tailwater category.

$$\text{OperatingHead}_i = \text{PoolElevation}_i - \text{TailwaterElevation}_i$$

$$\text{Power}_i = f(\text{TurbineRelease}_i, \text{OperatingHead}_i)$$

The Power calculation is dependent on the selected method in the Power category.

2. RiverWare calculates the PSA Head Factor Weights using a combination of linear regression and averages.

FOR EACH Turbine Release value *i*

FOR EACH combination of Spill and Tailwater Base Value (or Downstream Storage), *k*, perform a linear regression to calculate a temporary Storage coefficient g_{ik}:

$$g_{ik} = \frac{n\sum(\mathrm{Power}_j \times \mathrm{Storage}_j) - \left(\left(\sum \mathrm{Power}_j\right)\left(\sum \mathrm{Storage}_j\right)\right)}{n\left(\sum \mathrm{Storage}_j^2\right) - \left(\sum \mathrm{Storage}_j\right)^2}$$

where *n* is the number of Storage values.

Calculate the temporary Storage coefficient for the given Turbine Release, g_i:

$$g_i = \frac{\sum g_{ik}}{n}$$

where *n* is the number of Spill and Downstream Storage) combinations.

FOR EACH combination of Storage and Downstream Storage, *k*, perform a linear regression to calculate a temporary Spill coefficient e_{ik}:

$$e_{ik} = \frac{n\sum(\mathrm{Power}_j \times \mathrm{Spill}_j) - \left(\left(\sum \mathrm{Power}_j\right)\left(\sum \mathrm{Spill}_j\right)\right)}{n\left(\sum \mathrm{Spill}_j^2\right) - \left(\sum \mathrm{Spill}_j\right)^2}$$

where *n* is the number of Spill values.

Calculate the temporary Spill coefficient for the given Turbine Release, e_i:

$$e_i = \frac{\sum e_{ik}}{n}$$

where *n* is the number of Storage and Downstream Storage combinations.

FOR EACH combination of Storage and Spill, *k*, perform a linear regression to calculate a temporary Downstream Storage coefficient f_{ik}:

$$f_{ik} = \frac{n\sum(\mathrm{Power}_j \times \mathrm{DownstreamStorage}_j) - \left(\left(\sum \mathrm{Power}_j\right)\left(\sum \mathrm{DownstreamStorage}_j\right)\right)}{n\left(\sum \mathrm{DownstreamStorage}_j^2\right) - \left(\sum \mathrm{DownstreamStorage}_j\right)^2}$$

where *n* is the number of Downstream Storage values.

Calculate the temporary Downstream Storage coefficient for the given Turbine Release, f_i:

$$e_i = \frac{\sum e_{ik}}{n}$$

where *n* is the number of Storage and Spill combinations.

Calculate average ratios over all Turbine Release values:

$$c = \frac{\sum (e_i / g_i)}{n}$$

$$d = \frac{\sum (f_i / g_i)}{n}$$

where *n* is the number of Turbine Release values.

Calculate the final Head Factor Weights:

$$\text{Storage Weight} = \frac{1}{c}$$

$$\text{Spill Weight} = -1$$

If Tailwater Base Value is linked to the downstream Pool Elevation:

$$\text{Downstream Storage Weight} = \frac{d}{c}$$

These three weight values are displayed in the three columns of the PSA Head Factor Weights table slot.

In the optimization solution the PSA Head Factor is defined at a given time step by

If Tailwater Base Value is linked to the downstream Pool Elevation:

$$\text{PSA Head Factor} = \text{StorageWeight} \times \text{Storage} + \text{SpillWeight} \times \text{Spill} + \text{DownstreamStorageWeight} \times \text{Downstream Storage}$$

If Tailwater Base Value is not linked:

$$\text{PSA Head Factor} = \text{StorageWeight} \times \text{Storage} + \text{SpillWeight} \times \text{Spill}$$

3. RiverWare calculates the grid points for the PSA Grid Points table slot. The number of rows (points) in this table equals the product of the number of lines in the PSA Head Factor Grid Lines and PSA Turbine Release Grid Lines scalar slots. The smallest and largest values in the PSA Sample Input slot lead to the smallest and largest values for each of the parameters that affect power (Turbine Release, Storage, Spill and Downstream Storage. One row in this table will correspond to the smallest Turbine Release, smallest Storage, largest Spill and largest Downstream Storage, and thus smallest power. Another row will correspond to the largest Turbine Release, largest Storage, smallest Spill and smallest Downstream Storage, and thus the largest Power. RiverWare generates evenly spaced values between the extremes for each variable.Within the PSA Grid Points table, for a given Turbine Release value, Storage is increasing and Spill and Downstream Storage are decreasing. For each grid point, RiverWare calculates the value of Power and the PSA Head Factor corresponding to the parameter values in each row and adds these to the final two columns in the table.

For each row *i* in the table:

$$\text{Power}_i = f(\text{TurbineRelease}_i, \text{Storage}_i, \text{Spill}_i, \text{DownstreamStorage}_i)$$

The Power calculation depends on the method selected in the Power category.

$$\text{HeadFactor}_i = \alpha \cdot \text{Storage}_i + \beta \cdot \text{Spill}_i + \gamma \cdot \text{DownstreamStorage}_i$$

The coefficients α, β and γ come from the PSA Head Factor Weights table slot.

4. RiverWare calculates the upper bound planes. These are entered in the PSA Max Constraints table slot, one row for each plane, where the planes represent Power as a function of Turbine Release and PSA Head Factor. For each combination of three points in the PSA Grid Points Table, RiverWare calculates the corresponding plane. For any three points, the plane is defined by three linear equations, which can be represented in matrix form.

$$\bar{P} = \mathbf{A}\bar{u}$$

$$\bar{P} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} QT_1 & HF_1 & 1 \\ QT_2 & HF_2 & 1 \\ QT_3 & HF_3 & 1 \end{bmatrix}, \qquad \bar{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

The coefficients that define the plane are solved for by

$$\bar{u} = \mathbf{A}^{-1}\bar{P}$$

RiverWare only uses the planes that are necessary to define the Power Surface. Each plane is checked against all remaining points in the PSA Grid Points table. If the Power value for any of the remaining points lies above the given plane, then that plane is rejected (i.e. it would over-constrain Power, resulting in an under-estimation of Power). For each plane *i* that is used, RiverWare adds the following constraint to the optimization problem for each time step:

$$\text{Power} \le u_i \cdot \text{Turbine Release} + v_i \cdot \text{PSA Head Factor} + w_i$$

5. RiverWare calculates the lower bound planes. These are entered in the PSA Min Constraints table slot, one row for each plane. Only two planes can be calculated for the lower bounds. The points used for the two planes are as follows:

Plane 1:

Point A1 - Min Turbine Release, Max Head Factor

Point B1 - Max Turbine Release, Max Head Factor

Point C1 - Min Turbine Release, Min Head Factor

Plane 2:

Point A2 - Min Turbine Release, Max Head Factor

Point B2 - Max Turbine Release, Max Head Factor

Point C2 - Max Turbine Release, Max Head Factor

The coefficients defining the planes are calculated using the same matrix calculation described for the upper bound planes. For each lower bound plane *i*, RiverWare adds the following constraint to the optimization problem for each time step:

$$\text{Power} \geq u_i \cdot \text{Turbine Release} + v_i \cdot \text{PSA Head Factor} + w_i$$

A note on the lower bound approximation:

Because only two planes can be defined for the lower bounds on power, additional approximation error can be introduced if the optimization policy has an incentive to minimize power. This can occur if Power is included in a Minimize objective. It can also occur if the optimization problem contains constraints that require Power to be less than or equal to value. Note that these types of constraints can be introduced through multiple forms of RPL optimization goals. The first is basic less than or equal to constraint.

ADD CONSTRAINT $\text{Res.Power}[t] \leq \text{Value}$

Policy that requires Power to be equal to a value also adds a less than or equal to constraint. The following RPL statement:

ADD CONSTRAINT $\text{Res.Power}[t] = \text{Value}$

actually adds the following two constraints to the optimization problem:

$$\text{Res.Power}[t] \leq \text{Value}$$

$$\text{Res.Power}[t] \geq \text{Value}$$

A constraint to require the sum of Power from multiple reservoirs to be less than or equal to a value will have a similar effect. A final manner in which this can occur is through policy that minimizes Power indirectly through user-defined variables. For example, assume a user-defined variable is defined by the following RPL constraint:

ADD CONSTRAINT $\text{Res.Power}[t] + \text{UserVariable}[t] = \text{Value1}$

Then later the variable is constrained by

ADD CONSTRAINT $\text{UserVariable}[t] \geq \text{Value2}$

An equivalent constraint would be

ADD CONSTRAINT $\text{Res.Power}[t] \leq \text{Value1} - \text{Value2}$

It is possible to reduce the approximation error in these cases by adding constraints to the RPL Optimization Goal Set that essentially create more restrictive planes for the lower bound. These types of constraints are model-specific. Contact CADSWES if assistance is required to add more restrictive lower bounds to the Power Surface Approximation.

5.6.2.16 Power Linearization Automation

The Power Linearization Automation category allows the selection of the approximation points used for linearizing the power related slots to be done automatically or entered by the user.

It is dependent on selection of Independent Linearizations for the Optimization Power category and selection of None or Variable Head for the Optimization Head Computation category. (The Variable Head method is not currently functional in RPL Optimization.)

5.6.2.16.1 None

The method requires the user to input the approximation points for the power, best turbine flow (depending on the power method selected) and turbine capacity into the appropriate Power LP Param, Best Turbine Flow LP Param and Turbine Capacity LP Param tables, respectively.

5.6.2.16.2 Plant Automation

This method's availability for user selection is dependent on selection of Plant Power Coefficient in the Power category.

This method proceeds as follows, beginning with the Turbine Capacity LP Param table:

1) In the Max Turbine Q table, get the head values in the first row and the second to last row of the table, calling them FIRSTHEAD and LASTHEAD respectively.

2) Calculate AVEHEAD as the average of FIRSTHEAD and LASTHEAD.

3) Find the maximum value in the Turbine Capacity column (MaxTurbineCapacity) and its corresponding Operating Head (MaxTCOperatingHead) in the Max Turbine Q table. Given the nature of this table, the maximum value is not necessarily the last value. Also, if successive rows have the same Turbine Capacity, the first one (and its corresponding Operating Head) is chosen.

4) If maxTCOperatingHead > LASTHEAD - 0.5 m, then maxTCOperatingHead is set to LASTHEAD - 1 m.

The Turbine Capacity LP Param table is then defined using the variables above: The tangent approximation value is set as AVEHEAD. The line approximation points are set to FIRSTHEAD + 0.5 m and LASTHEAD - 0.5 m. The piecewise approximation points are set to FIRSTHEAD + 0.5, MaxTCOperatingHead, and LASTHEAD - 0.5.

The method then repeats the process using the Best Turbine Q table. (In this table the second column is now called Best Capacity whereas the second column in the Max Turbine Q table is called Turbine Capacity.

Next the Power LP Param table is filled in:

1) If the initial Operating Head is not known, it is calculated as initial Pool Elevation - initial Tailwater Elevation. The Elevation Volume Table and initial Storage will be used to determine initial Pool elevation, if necessary. The initial Operating Head is called MIDHEAD.

2) The Maximum Turbine Q table is queried using MIDHEAD to determine the corresponding

maximum turbine flow, here called FLOWMAX.

3) The Best Turbine Q table is queried using MIDHEAD to determine the corresponding best turbine flow. FLOWBEST is set to this best turbine flow - 0.01 cms. If FLOWBEST is greater than or equal to FLOWMAX, then reset FLOWBEST to FLOWMAX - 0.01.

The Power LP Param table is then defined using the variables above: The Operating Head value is set to MIDHEAD. The tangent approximation value is set as (FLOWBEST + FLOWMAX) / 2. The line approximation points are set to 0 and (FLOWBEST + FLOWMAX) / 2. The piecewise approximation points are set to 0, FLOWBEST, and FLOWMAX.

Now the method creates the Plant Power Table (defining power as a function of operating head and turbine release). The method adds the best and max power generation (power and flow) points for each operating head by combining the data in the Best Turbine Q, Best Power Coefficient, Max Turbine Q, and Max Power Coefficient tables. The details of the algorithm are as follows. For each row (each Operating Head) in the Max Turbine Q table:

1. Determine the Operating Head (OPHEAD) and Turbine Capacity (MAXQ) values.

2. Query the Best Turbine Q table using OPHEAD to determine Best Turbine Flow (BESTQ).

3. Query the Best Power Coefficient table using the Operating head to determine the Best Power Coefficient.

4. Query the Max Power Coefficient table using the Operating head to determine the Max Power Coefficient.

5. Query the Best Power Coefficient table using the Operating head to determine the Best Power Coefficient.

6. Calculate the MAXPOWER as MAXQ * Max Power Coefficient.

7. Calculate the BESTPOWER as BESTQ * Best Power Coefficient.

8. Create a row in the table using OPHEAD, 0, 0 (Operating Head, Turbine Release, and Power columns, respectively).

9. If BESTQ > 0, then create a row in the table using OPHEAD, BESTQ, BESTPOWER.

10. If MAXQ does not equal BESTQ, then create a row in the table using OPHEAD, MAXQ, MAXPOWER.

11. If MAXQ does not equal MaxTurbineCapacity (retained from the Turbine Capacity LP Param table work above), then create a row in the table using OPHEAD, MaxTurbineCapacity, MAXPOWER.

Finally, the initial (timestep) value of the Power Coefficient slot is assigned as determined by querying the Best Power Coefficient table using MIDHEAD (retained from the Power LP Param table work above) for Operating Head.

5.6.2.16.3 Peak Automation

This method's availability for user selection is dependent on selection of Peak Power in the Power category.

This method proceeds as follows, beginning with the Turbine Capacity LP Param table:

1. The Number of Units table is queried. Its value is herein called NUMUNITS.

2. Get the head values in the first row and the second to last row of the Best Generator Flow table, calling them FIRSTHEAD and LASTHEAD respectively.

3. Calculate AVEHEAD as the average of FIRSTHEAD and LASTHEAD.

4. Find the maximum value of flow in the Type #1 Flow column (MaxFlow) and its corresponding Operating Head (HEADMAXQ) in the Best Generator Flow table. Given the nature of this table, the maximum flow value is not necessarily the last value. Also, if successive rows have the same Type #1 Flow, the first one (and its corresponding Operating Head) is chosen.

5. If HEADMAXQ > LASTHEAD - 0.5 m, then HEADMAXQ is set to LASTHEAD - 1.0 m.

6. Calculate ALLMAXFLOW as MaxFlow * NUMUNITS.

7. The Turbine Capacity LP Param table is then defined using the variables above: The tangent approximation value is set as AVEHEAD. The line approximation points are set to FIRSTHEAD + 0.5 m and LASTHEAD - 0.5 m. The piecewise approximation points are set to FIRSTHEAD + 0.5 m, HEADMAXQ, and LASTHEAD - 0.5 m.

Next the Power LP Param table is filled in:

1. If the initial Operating Head is not known, it is calculated as initial Pool Elevation - initial Tailwater Elevation. The Elevation Volume Table and initial Storage will be used to determine initial Pool elevation, if necessary. The initial Operating Head is called MIDHEAD.

2. The Best Generator flow is queried using MIDHEAD to determine the corresponding Type #1 Flow, here called FLOWMAX.

3. Calculate ALLFLOWMAX as FLOWMAX * NUMUNITS - 0.01 cms (where NUMUNITS is retained from the Turbine Capacity LP Param table work above).

4. Calculate ALLFLOWBEST as ALLFLOWMAX - 0.01 cms.

The Power LP Param table is then defined using the variables above: The Operating Head value is set to MIDHEAD. The tangent approximation value is set as the average of ALLFLOWBEST and ALLFLOWMAX. The line approximation points are set to 0 and the average of ALLFLOWBEST and ALLFLOWMAX. The piecewise approximation points are set to 0 cms, ALLFLOWBEST and ALLFLOWMAX.

Now the method creates the Plant Power Table (defining power as a function of operating head and turbine release). For each row (each Operating Head) in the Best Generator Flow table:

1. Determine the Operating Head (OPHEAD) and Type #1 Flow (BESTQ) values.

2. Query the Best Generator Power table using OPHEAD to determine Type #1 Power (BESTPOWER).

3. Calculate ALLBESTQ = BESTQ * NUMUNITS (retained from the Turbine Capacity LP Param table work above).

4. Calculate ALLBESTPOWER = BESTPOWER * NUMUNITS (retained from the Turbine Capacity LP Param table work above).

5. Create a row in the Plant Power Table with OPHEAD, 0, 0 (Operating Head, Turbine Release, and Power columns, respectively).

6. If ALLBESTQ > 0, the create a row in the table using OPHEAD, ALLBESTQ, ALLBESTPOWER.

7. If ALLBESTQ does not equal ALLMAXFLOW (retained from the Turbine Capacity LP Param table work above), then create a row in the table using OPHEAD, ALLMAXFLOW, and ALLBESTPOWER.

Finally, the method checks that the MIDHEAD value assigned as Operating Head in the Power LP Param table is contained within the Operating Head range of the Plant Power Table. If it is not, and error occurs.

5.6.2.17 Power category

For more information on the simulation methods, click **HERE (Objects.pdf, Section 21.1.1)**. Only a subset of available Simulation methods are available in Optimization. Selection of a method other than those that follow will result in an error.

If the Optimization problem uses Power, either directly or indirectly, then this slot is added to the problem. Before being passed to the optimization solver, this slot is numerically approximated as a function of Operating Head and Turbine Release (Numerical 3-D Approximation). The relationship between Operating Head and Turbine Release will come from the Plant Power Table. This table will either be user-input, or automatically parameterized depending on the method selection in the Power Linearization Automation category. The table will be queried using the points defined in the Power LP Param table. The values in the Power LP Param table also will either be user-input or automatically calculated points, depending on the method selection in the Power Linearization Automation category. If the selected method in the Power Linearization Automation is “none”, then the tables should contain user-input values which are used. For other Power Linearization Automation methods (Plant Automation or Peak Automation) the tables will contain automatically calculated values which are used.

RELATED SLOTS

☞ **PLANT POWER TABLE**

Type:	Table Slot
UNITS:	LENGTH VS. FLOW VS. POWER
Description:	Table defining the relationship between Operating Head, Turbine Release and Power at selected points of operation. In building the Plant Power table, the following

approach might be taken. For a given Operating Head create n + 2 rows of data, where n is the number of units comprising the Plant. The first row will reflect no flow through the turbines. The next row will reflect the preferred level of flow through the first unit normally activated. Subsequent rows will reflect incremental flow levels as additional turbines come on line at their respective preferred levels of flow. The final row will reflect the flow of all available units running at maximum flow capacity.

Information: This table must be user input unless the Power Linearization Automation category has selected either Plant Automation or Peak Automation methods.

Defined by:

Power

Type: Agg Series Slot

Units: Power

Description: Power generated by flow through the turbines

Information:

Defined by: Numerical 3-D Approximation in terms of Operating Head and Turbine Release. Approximation is based on the Plant Power Table. The Power LP Param table contains a value for Operating Head used to index the Operating Head column of the Plant Power Table. This approximated value, therefore, reduces the Power to a function of Turbine Release. The flow values in the Power LP Param table are then used as approximation points indexing the Turbine Release column of the Plant Power Table. The Plant Power Table should have increasing values of Operating Head and Turbine Release. Power should be a concave function of Operating Head, but concavity is not strictly enforced. The preferred order of approximation is substitution, piece-wise, two-point line, tangent.

Power LP Param

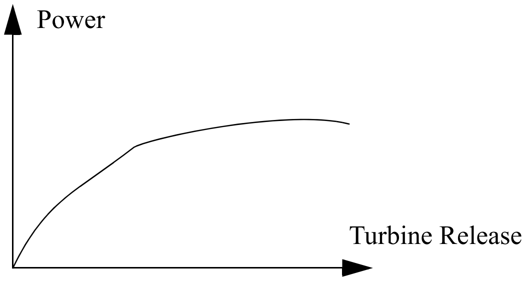
Type: Table Slot

Units: Length and Flow

Description: Specifies the Operating Head and the flow points used to take the tangent, line and piecewise approximations for Power linearization

Information: This table must be user input unless the Power Linearization Automation category has selected either Plant Automation or Peak Automation methods. The best Operating Head to choose should be close to the expected head during optimized period. Tangent approximation is generally not used, nor helpful as it often results in non-zero power for zero flow. The suggested points for a line approximation are 0 flow and best turbine flow for the entire plant. The suggested points for piecewise linearization are 0 flow, 1 unit best turbine flow, 2 units best turbine flow, . . . n units best turbine flow, and maximum turbine flow.

Defined by:



Power Turbine Release relationship. This is for a fixed operating head value. Not drawn to scale.

5.6.2.17.1 Plant Power Coefficient

SLOTS SPECIFIC TO THIS METHOD

In the RPL Optimization mode, the following slot requires special handling.

☞ **BEST TURBINE FLOW**

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow associated with the most efficient power generation for an associated Operating Head
Information:	
Defined by:	Numerical 2-D Approximation in terms of Operating Head, based upon an internal best turbine flow table. For the Plant Power Coefficient method (selected in the Power category) the internal table is the Best Turbine Q table. The Best Turbine Q table is required to have increasing values of Operating Head and Best Capacity and be a concave function of Operating Head. The preferred order of approximation is substitution, piece-wise, tangent, two-point line. The Best Turbine Flow LP Param table values are used as approximation points indexing the best turbine flow table.

If the Optimization problem uses Best Turbine Flow, either directly or indirectly, then this slot is added to the problem. Before being passed to the optimization solver, this slot is numerically approximated as a function of Operating Head (Numerical 2-D Approximation). The relationship between Operating Head and Best Turbine Flow will come from one of two places, depending on other method selection. 1) If the selected method in the Power category is Plant Efficiency Curve, then the relationship is automatically developed from the Plant Power Table. 2) For the Plant Power Coefficient method, the user-input Best Turbine Q table defines the relationship. 3) For other Plant Calculation category methods, Best Turbine Flow must be input.

Best Turbine Flow LP Param

Type: Table Slot
Units: Length
Description: Specifies the Operating Head points in the best turbine flow relationship used to take the tangent, line and piecewise approximations for Best Turbine Flow linearization
Information: The best operating head points to choose for a piecewise approximation are generally: minimum Operating Head, Operating Head at max capacity, and maximum Operating Head. These three points typically define most of the shape of the Best Turbine Flow curve.
Defined by: User-input

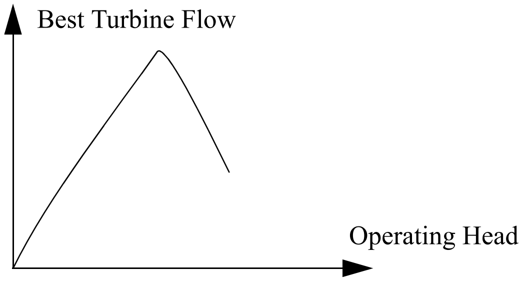
Best Turbine Q

Type: Table Slot
Units: Length vs. Flow
Description: Table defining the relationship between Operating Head and the most efficient power generation.
Information: This table is used if the Power category has the Plant Power Coefficient method selected. If the Power category has the Plant Efficiency Curve method selected, then this table is not used, but a similar relationship is automatically developed from the Plant Power Table.
Defined by: User-input

Plant Power Table

Type: Table Slot
Units: Length vs. Flow vs Power
Description: Table defining the relationship between Operating Head, Turbine Release, and Power at selected points of operation. In building the Plant Power table, the following approach might be taken. For a given Operating Head create n + 2 rows of data, where n is the number of units comprising the Plant. The first row will reflect no flow through the turbines. The next row will reflect the preferred level of flow through the first unit normally activated. Subsequent rows will reflect incremental flow levels as additional turbines come on line at their respective preferred levels of

	flow. The final row will reflect the flow of all available units running at maximum flow capacity.
Information:	This table is used to automatically develop the Best Turbine Flow relationship if Plant Efficiency Curve IS SELECTED for the Power category. If it is not, then this table is not used; the Best Turbine Q table is used.
Defined by:	User-input



Best Turbine Flow - Operating Head relationship. Not drawn to scale.

5.6.2.17.2 Plant Efficiency Curve

The Plant Efficiency Curve method approximates Best Power Flow the same as the Plant Power Calc

5.6.2.17.3 Peak Power

In RPL Optimization, this method creates a turbine release constraint:

Turbine Release <= Power Plant Capacity Fraction * Turbine Capacity * Number of Units. **(EQ 30)**

No actual power value is calculated.

5.6.2.17.4 Peak and Base

In RPL Optimization, this method creates a turbine release constraint:

Turbine Release <= Power Plant Capacity Fraction * Turbine Capacity * Number of Units. **(EQ 31)**

No actual power value is calculated.

5.6.2.17.5 Unit Power Table

The Unit Power slots and Simulation algorithm is described **HERE (Objects.pdf, Section 21.1.1.12)**. The optimization formulation is described **HERE (Section 8.2.6)**.

When the Unit Power Table method is selected, the following categories are available: Cavitation, **HERE (Section 5.6.2.25)** , Avoidance Zones, **HERE (Section 5.6.2.26)**, Startup, **HERE (Section 5.6.2.22)**, and Head Loss, **HERE (Section 5.6.2.24)**, and Frequency Regulation, **HERE (Section 5.6.2.27)**

When the Unit Power Table method is selected, the following categories are available: Cavitation, Avoidance Zones, Net Head, and Unit Regulation.

The status of a unit at a given timestep can be one of the following:

- Available
- Unavailable
- Must run

At the highest level, an entire plant may never be able to provide an ancillary service like Regulation. The user could indicate this by selecting "None" in the appropriate category. On the other hand, the following slot settings will allow users to specify unit availability / must run for individual time steps for generation and ancillary services. Without any inputs, the units will default to being available.

If the Unit Power Table method, but not the Frequency Regulation, method is selected, the user could specify through optimization RPL policy (note, setting certain slots may be possible but not preferred):

- Unit u must generate at time t:
    - Unit Is Generating [t,u] = 1, or
    - Unit Energy [t,u] = value
- Unit u is unavailable at time t:
    - Unit Is Generating [t,u] = 0, or
    - Unit Energy [t,u] = 0

If the Frequency Regulation method is selected, the user could specify through optimization policy (RPL):

- Unit u must regulate up at time t:
    - Regulation Up [t,u] = value
- Unit u must regulate down at time t:
    - Regulation Down [t,u] = value
- Unit u must regulate at time t:
    - Regulation [t,u] = value (which implies Regulation Up [t,u] = value and Regulation Down [t,u] = value)

If a unit is unavailable for some type of regulation, then a value can be 0.

5.6.2.18 Turbine Capacity

If the selected method for the Optimization Head Computation category is either None or Variable Head (not supported in RPL Optimization), then Turbine Capacity is numerically approximated before being passed to the optimization solver. The slot is approximated as a function of Operating Head (Numerical 2-D Approximation). The relationship between Operating Head and Turbine Capacity will come from one of several places, depending on the method selected in the Power category. 1) If plant-Efficiency Curve is chosen, the relationship is automatically developed from the Plant Power Table. 2) Otherwise, if Peak Power or Peak and Base is chosen, the relationship comes from the Best Generator

Flow table. 3) If none of these methods are chosen, then the relationship is contained in the user-input Max Turbine Q slot.

RELATED SLOTS

☞ TURBINE CAPACITY

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow capacity of the turbine(s)
Information:	
Defined by:	Numerical 2-D Approximation in terms of Operating Head, based upon a maximum turbine capacity table. This capacity table is determined in various ways according to the Power method:

Plant Efficiency Curve: an automatically generated maximum turbine flow table developed from the Plant Power Table

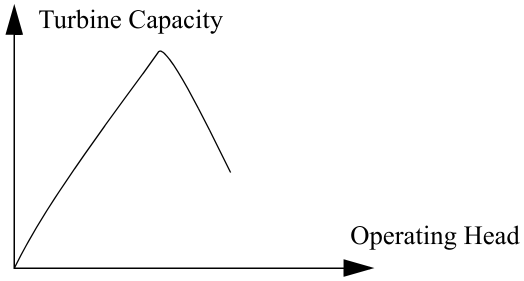
Peak Power or Peak and Base: Best Generator Flow table

All Others: Max Turbine Q

The Turbine Capacity LP Param table values are used as approximation points indexing the selected maximum turbine capacity table. The maximum turbine capacity table is required to have increasing values of Operating Head. Turbine Capacity in that table is required to be a concave function of Operating Head. The preferred order of approximation is substitution, piece-wise, tangent, two-point line.

☞ TURBINE CAPACITY LP PARAM

Type:	Table Slot
UNITS:	LENGTH VS. LENGTH VS. LENGTH
Description:	Specifies the operational head points used to take the tangent, line and piecewise approximations for Turbine Capacity linearization
Information:	For the piecewise approximation the best operating head points to chose are generally: minimum Operating Head, Operating Head at max capacity, and maximum Operating Head. These three points typically define most of the shape of the Turbine Capacity curve.
Defined by:	User-input unless the selected method in the Power Linearization Automation category is not “None”



Turbine Capacity Operating Head relationship. Not drawn to scale.

5.6.2.19 Optimization Tailwater

Click **HERE (Objects.pdf, Section 21.1.7)** for more information on the Simulation methods for Tailwater. Only a subset of available methods are supported in Optimization. These methods are: Linked or Input, Base Value Only, Base Value Plus Lookup Table, Stage Flow Lookup Table, and Coefficients Table. Selection of a method other than those will result in an error. The method selected in the Optimization Tailwater category should correspond to the one selected in the simulation Tailwater category.

Tailwater Elevation may be part of the Optimization problem and it is handled differently for each method. Below is a description of the behavior of Tailwater Elevation.

5.6.2.19.1 Opt Linked or Input

If Tailwater Elevation is not input, then Tailwater Elevation is replaced by TailwaterBaseValue: Tailwater Elevation = Tailwater Base Value.

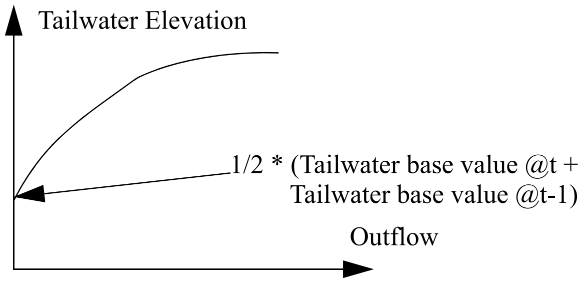
☞ **TAILWATER BASE VALUE**

Type:	Series
UNITS:	LENGTH
Description:	Described in the simulation documentation **HERE (Objects.pdf, Section 21.1.7.2)**

5.6.2.19.2 Opt Base Value Only

If Tailwater Elevation is not input, then Tailwater Elevation is replaced by (Tailwater Base Value(t) + Tailwater Base Value(t-1)) / 2

☞ **TAILWATER BASE VALUE**
Type: Series
UNITS: LENGTH
Description: Described in the simulation documentation **HERE (Objects.pdf, Section 21.1.7.3)**



Tailwater Outflow relationship. Not drawn to scale.

5.6.2.19.3 Opt Base Value Plus Lookup Table

If Tailwater Elevation is not input, this slot is replaced by a mathematical expression based on Tailwater Base Value and a numerical approximation of Tailwater Elevation as a function of Outflow (Numerical 2-D Approximation) as defined by the user in the Tailwater Table

Tailwater Elevation is constrained to be (tailwaterBaseValue(t) + tailwaterBaseValue(t-1))/2 + tempTWLookupValue, where tempTWLookupValue is a Numerical 2-D Approximation of increase in Tailwater Elevation due to flow as described in the Tailwater Table. The Tailwater Table Lookup LP Param table values for outflow are used as approximation points indexing the Tailwater Table to determine tempTWLookupValue from the Tailwater Elevation column.

☞ **TAILWATER BASE VALUE**
Type: Series
UNITS: LENGTH
Description: Described in the simulation documentation **HERE (Objects.pdf, Section 21.1.7.4)**

☞ **TAILWATER TABLE**
Type: Series Slot
UNITS: FLOW VS. LENGTH
Description: Reservoir outflow vs. either the tailwater elevation or the tailwater elevation increment
Information: If the Tailwater Base Value is non-zero, the Tailwater Table gives values of incremental increase in Tailwater Elevation over the Base value. If the Tailwater Base Value is zero, the table simply gives the Tailwater Elevation values. The Tailwater Table should have increasing values of outflow. Tailwater Elevation is required to be a convex function of outflow. The preferred order of approximation is

substitution, piece-wise, two-point line, tangent. Please see Object / Simulation documentation for further information.
I/O: User-input

☞ Tailwater Table Lookup LP Param

Type: Table Slot
Units: Flow, Flow, Flow
Description: LP Parameter approximation points for Outflow.
Information: The Tailwater Table Lookup LP Param table values for outflow are used as approximation points indexing the Tailwater Table to determine tempTWLookupValue from the Tailwater Elevation column.
I/O: User Input only
Links: Not Linkable

☞ temp Tailwater Lookup Value

Type: Series
Units: Length
Description: Numerical 2-D Approximation of increase in Tailwater Elevation due to flow as described in the Tailwater Table.

5.6.2.19.4 Opt Stage Flow Lookup Table

For more information on the Stage Flow Lookup Table method, click **HERE (Objects.pdf, Section 21.1.7.5)**. Tailwater Elevation is numerically approximated as a function of Stage and Flow as defined by the user in the Stage Flow Tailwater Table. The tables will be queried either using user-input points defined in the Tailwater Elevation LP Param table or using automatically calculated points, depending on method selection in the Tailwater Linearization Automation category. If the selected method in the Tailwater Linearization Automation category is “None”, then the user-input Tailwater LP Param table values are used. For the Range Input Automation method, the automatically developed points will be used.

Tailwater Elevation is approximated numerically (3-D approximation) as a function of Outflow and Tailwater Base Value based on the Stage Flow Tailwater Table. The Tailwater Elevation LP Param table Tailwater Base Value and flow values are used as approximation points indexing the Downstream Stage and Outflow columns of the Stage Flow Tailwater Table, respectively.

Tailwater Base Value

Type: Series
Units: Length
Description: Described in the simulation documentation **HERE (Objects.pdf, Section 21.1.7.5)**

Stage Flow Tailwater Table

Type: Table Slot
Units: Flow vs. Length vs. Length
Description: Reservoir Outflow vs. Downstream Elevation (Tailwater Base Value) vs. Tailwater Elevation
Information: Data must be entered into the table in increasing blocks of the same Outflow value for the 3-dimensional table interpolation to work correctly. For every block of same Outflows in column 1, Downstream Stages should be listed in increasing order in column 2, and the corresponding Tailwater Elevations in column 3. Tailwater elevation is required to be a concave function of Outflow. The preferred order of approximation is substitution, piece-wise, two-point line, tangent. Internally, the Stage Flow Tailwater Table is rearranged to use the Stage as the primary index (column) with outflow as the secondary index (column). This rearranged table may be labeled as “Convolved Stage Flow Tailwater Table” in output messaging. Also, because of this rearrangement, it is desirable to repeat stage values for each outflow.
Defined by: User-input

Tailwater Elevation LP Param

Type: Table Slot
Units: Length, Flow, Flow, Flow
Description: Specifies the fixed tailwater base value point and the outflow points used to take the tangent, line and piecewise approximations for tailwater elevation linearization
Information: This table is used for linearization. The best Tailwater Base Value point to choose for tangent approximation would be an outflow equal to the expected value of outflow during the run; for the line approximation, the minimum and maximum values expected during the run; for piecewise approximation, the minimum and maximum values expected during the run plus additional intermediate values to more closely fit the tailwater curve.
Defined by: User-input unless the selected method in the Tailwater Linearization Automation category is not “None”.

TAIL WATER REFERENCE ELEVATION

Type:	Table Slot
UNITS:	LENGTH
Description:	Lowest Reservoir discharge Elevation when there are no backwater effects from a downstream pool (reservoir)
Information:	Although this is part of the Stage Flow Lookup Table method (and its corresponding Opt method), **this slot does not influence optimization**. Please see Object / Simulation documentation for further information.
I/O:	User-input

5.6.2.19.5 Opt Coefficients Table

This method sets up the physical Tailwater Elevation constraints using the same equation and coefficients used in the Coefficients Table tailwater method, **HERE (Objects.pdf, Section 21.1.7.7)**:

$$
\begin{aligned}
\text{Tailwater Elevation} = {} & \text{Constant Coeff}[t] + \text{Constant Coeff}[t-1] + \\
& \text{Outflow Coeff}[t] \times \text{flow} + \text{Outflow Coeff}[t-1] \times \text{Outflow}[t-1] + \\
& \text{TW Base Val Coeff}[t] \times \text{TailwaterBaseValueTemp}[t] + \\
& \text{TW Base Val Coeff}[t-1] \times \text{Tailwater BaseValue}[t-1] + \\
& \text{TW Elev Coeff}[t-1] \times \text{Tailwater Elevation}[t-1]
\end{aligned}
$$

TAILWATER BASE VALUE

Type:	Series
UNITS:	LENGTH
Description:	See Simulation description **HERE (Objects.pdf, Section 17.1.7.7)**.

TAILWATER COEFFICIENTS

Type:	Table
UNITS:	MULTI
Description:	See Simulation description **HERE (Objects.pdf, Section 21.1.7.7)**.
Defined by:	If the Power Surface Approximation method is selected, then there are further restrictions on the coefficients that can be specified. With the Power Surface Approximation, having terms for Outflow[t-1], Tailwater BaseVal[t] and Tailwater[t-1] are not allowed.

5.6.2.20 Tailwater Linearization Automation

This method category allows the user to choose whether they want to enter the approximation points used to the linearization approximations of tailwater or have the selection done automatically by riverware. The default method is none, which requires the points be input by the user. The automation method Range Input determines the points using the expected range of outflow values.

This category is dependent on 1) Optimization Head Calculation category selection of either Variable Head (not available in RPL Optimization) or None, and 2) Optimization Power category selection of Independent Linearizations, and 3) Optimization Tailwater category selection of either Opt Base Value Plus Lookup Table or Opt Stage Flow Lookup Table.

5.6.2.20.1 None

This method requires the input of the approximation points for tailwater linearization into the Tailwater LP param table.

5.6.2.20.2 Range Input Automation

The Range Input Automation method selects the linearization points for the linear approximations of tailwater in the Tailwater Elevation LP Param table. Tailwater is linearized as a 2- or 3-dimensional function depending on the method selected in the Optimization Tailwater category. If Opt Base Value Plus Lookup Table is selected tailwater is a 2-D function of flow. If Opt Stage Flow Lookup Table is selected tailwater is a 3-D function of flow and downstream stage. The same flow points are used for both the 2- and 3-D approximations. For the 3-D approximation the fixed point is required for the downstream stage. This point must still be input by the user.

SLOTS SPECIFIC TO THIS METHOD

☞ **EXPECTED OUTFLOW RANGE**

Type:	Table Slot
UNITS:	FLOW, FLOW
Information:	
Information:	Provide the high and low expected outflow values for the run.
Defined by:	User Input

The user inputs high and low expected values into the Expected Outflow Range table. The average of the values is calculated. The average is used as the tangent approximation point. The low and high values are used as the line approximation points. The low, average and high points are used as the piecewise points. The fixed point for three dimensional approximations is still required to be user input, but the validity of the point is checked during the automation process.

5.6.2.21 Pump Power Numerical Approximation

If the Optimization problem uses Pump Power, either directly or indirectly, then this slot is added to the problem. Before being passed to the optimization solver, this slot is numerically approximated as a function of Operating Head and Pumped Flow (Numerical 3-D Approximation). The relationship between Operating Head and Pumped Flow will come from the user-input Pump Power Lin Data table. The table will be queried using the points defined in the Pump Power LP Param table. The values in the Pump Power LP Param table will be entirely user-input or may include an automatically calculated value for Operating Head. If the Operating Head is not available at the beginning of the run, then this value will be automatically assigned by RiverWare.

RELATED SLOTS

(LISTED ALPHABETICALLY)

☞ **PUMP POWER**

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Power required to operate the pumps
Information:	
Defined by:	Numerical 3-D Approximation in terms of Operating Head and Pumped Flow based on the Pump Power Lin Data table. The Pump Power LP Param table contains a value for Operating head used to index the Operating Head column of the Pump Power Lin Data table. This approximated value, therefore, reduces the Pump Power to a function of Pumped Flow at the given Operating Head. The flow values in the Pump Power LP Param table are then used as approximation points indexing the Pumped Flow column of the Pump Power Lin Data table. The Pump Power Lin Data table should have increasing values of Operating Head and Pumped Flow. Pump Power should be a convex function of Operating Head, but concavity is not strictly enforced; mild non-convex regions are permissible to allow for round-off error, etc. The preferred order of approximation is substitution, piece-wise, tangent, two-point line.

☞ **PUMP POWER LIN DATA**

Type:	Table Slot
UNITS:	LENGTH VS FLOW VS POWER
Description:	Table defining the relationship between Operating Head, Pumped Flow and power expended (Pump Power).
Information:	
Defined by:	User-input

☞ **PUMP POWER LP PARAM**

Type:	Agg Series Slot
UNITS:	LENGTH, FLOW, FLOW, FLOW
Description:	Specifies the Operating Head and the flow points used to take the tangent, line and piecewise approximations for Pump Power Lin Data linearizations.
Information:	This slot is the pumping equivalent of the Power LP Param table. The best Operating Head to choose should be close to the expected head during optimized period. Tangent approximation is generally not used, nor helpful as it often results in non-zero power for zero flow. The suggested points for a line approximation are 0 flow and the preferred pumping flow. The suggested points for piecewise linearization are 0 flow, 1 unit preferred pumping flow, 2 units preferred pumping flow, . . . n units preferred pumping flow, and maximum pumping flow.

5.6.2.22 Optimization Reserves

This category depends on selecting either Plant Power Coefficient or Plant Efficiency Curve in the Power category. The methods in this category can be used to account for power reserve requirements in the optimization policy. There is no analogous simulation method associated with this category.

5.6.2.22.1 None

This is the default, do-nothing method. No new slots will be added, and no new variables or constraints will be added to the optimization problem.

5.6.2.22.2 Constraint Based Single Timestep

This method introduces a set of duplicate variables that represent the value of standard variables assuming full deployment of either upward or downward reserves. In the RPL Optimization Goal Set, users can then write parallel constraints using the duplicate reserve variables for any constraint that cannot be violated while deploying reserves. The purpose of the new variables and their parallel constraints is so that the only reserves that are credited to a hydropower project are those that can actually be reasonably deployed. In other words, it will not “count” reserves that could not be deployed without violating specified constraints. The new method also introduces new variables for upward and downward reserves, which allow the user to write policy on the reserves themselves, for example, project minimum reserve requirements or total system reserve requirements. This method only models reserves from the generating portion of the Pumped Storage Reservoir. It does not model reserves that might be provided by pumping.

One significant assumption is made that the deployment of these reserves are temporary and only meant to cover a single timestep. It is assumed that in the event reserves are deployed in actual operations, adjustments would be made for later time periods, including reestablishing reserves. With this assumption, the consequences of deploying reserves at a reservoir are limited to that reservoir and that timestep; constraints on downstream reservoirs and later timesteps are not considered when calculating the reserves that can be credited at a given timestep.

The deployment variables are constrained similarly to the original variables with two exceptions. First, the mass balance constraints with other reservoirs or intermediate reaches are omitted (within RiverWare). Second, if there are any constraints that can be violated during deployment then these constraints can also be omitted (within the Optimization Goal Set). The deployment variables will typically use the same linear approximations as the original variables. For example, Pool Elevation with Up Reserve will be approximated using the Pool Elevation LP Param table slot.

Note that this method has no analogous simulation method. The slots introduced by this method can be referenced in the RPL optimization policy to constrain the solution, but they will not, generally, display values after the post-optimization rulebased simulation like the standard variables, such as Outflow or Pool Elevation. The user does have the option of writing customized post-optimization rules to set values in these slots based on user-specified logic or using the OptValue or OptValuePiecewise predefiend functions.

SLOTS SPECIFIC TO THIS METHOD

This method will add the following series slots, all of which will be available as variables for the RPL optimization policy.

☞ UP RESERVE

Type:	Series Slot
UNITS:	POWER
Description:	The upward reserves available; the amount the reservoir could increase generation on the given timestep
Information:	Introducing this variable in the optimization policy forces a two-point line approximation to be used for both Power and Power with Up Reserve if Independent Linearizations is selected in the Optimization Power category. This may introduce additional approximation error.
Defined by:	$\text{Up Reserve} = \text{Power with Up Reserve} - \text{Power}$

☞ DOWN RESERVE

Type:	Series Slot
UNITS:	POWER
Description:	The downward reserves available; the amount the reservoir could decrease generation on the given timestep
Information:	Introducing this variable forces a two-point line approximation to be used for both Power and Power with Down Reserve if Independent Linearizations is selected in the Optimization Power category. This may introduce additional approximation error.
Defined by:	$\text{Down Reserve} = \text{Power} - \text{Power with Down Reserve}$

☞ POWER WITH UP RESERVE

Type:	Series Slot
UNITS:	POWER
Description:	The average Power generated if full upward reserves were deployed
Information:	This variable is defined by the Power approximation as a function of Turbine Release with Up Reserve and a combination of other variables. The set of other variables depends on which Power approximation is being applied. The Power approximation that is applied is dependent on the selected method in the Optimization Power category.
Defined by:	$\text{Power with Up Reserve} = f(\text{Turbine Release with Up Reserve}, \ldots)$

☞ POWER WITH DOWN RESERVE

Type:	Series Slot
UNITS:	POWER
Description:	The average Power generated if full downward reserves were deployed
Information:	This variable is defined by the Power linear approximation as a function of Turbine Release with Down Reserve and a combination of other variables. The set of other variables depends on which Power approximation is being applied. The Power approximation that is applied is dependent on the selected method in the Optimization Power category.
Defined by:	$\text{Power with Down Reserve} = f(\text{Turbine Release with Down Reserve}, \ldots)$

☞ TURBINE RELEASE WITH UP RESERVE

Type: Series Slot
UNITS: FLOW
Description: The Turbine Release at full deployment of upward reserves
Information: This variable is constrained to be less than or equal to Turbine Capacity with Up Reserve.
Defined by: $\text{Turbine Release with Up Reserve} = \text{Turbine Release} + \text{Turbine Change with Up Reserve}$

and

$$\text{Turbine Release with Up Reserve} \le \text{Turbine Capacity with Up Reserve}$$

☞ TURBINE CHANGE WITH UP RESERVE

Type: Series Slot
UNITS: FLOW
Description: The amount Turbine Release would change to fully deploy all upward reserves
Information: This variable is expected to always be non-negative, and thus the slot configuration lower bound should be set to 0.
Defined by: $\text{Turbine Release with Up Reserve} = \text{Turbine Release} + \text{Turbine Change with Up Reserve}$

☞ TURBINE CAPACITY WITH UP RESERVE

Type: Series Slot
UNITS: FLOW
Description: The maximum Turbine Release possible given the full deployment of upward reserves
Information: This variable is defined by the Turbine Capacity approximation as a function of Operating Head with Up Reserve.
Defined by: $\text{Turbine Capacity with Up Reserve} = f(\text{Operating Head with Up Reserve})$

☞ OPERATING HEAD WITH UP RESERVE

Type: Series Slot
UNITS: FLOW
Description: The Operating Head at full deployment of upward reserves.
Information:
Defined by:

$$\text{Operating Head with Up Reserve} = \frac{\text{Pool Elevation with Up Reserve} + \text{Pool Elevation}[t-1]}{2} - \text{Tailwater Elevation with Up Reserve}$$

Spill with Up Reserve

Type: Series Slot
Units: Flow
Description: The amount of total Spill at full deployment of upward reserves
Information: This could be equal to Spill, or it could be less than Spill if flow is allowed to be shifted from Spill to Turbine Release during deployment. This depends on how the RPL policy is formulated.
Defined by: Spill with Up Reserve =
Unregulated Spill with Up Reserve + Regulated Spill with Up Reserve + Bypass with Up Reserve

Unregulated Spill with Up Reserve

Type: Series Slot
Units: Flow
Description: The amount of Unregulated Spill at full deployment of upward reserves
Information: This quantity is defined by the Unregulated Spill linear approximation as a function of Storage with Up Reserve. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Unregulated Spill, then the value of this variable will automatically be set to zero.
Defined by: Unregulated Spill with Up Reserve = f(Storage with Up Reserve)

Regulated Spill with Up Reserve

Type: Series Slot
Units: Flow
Description: The amount of Regulated Spill at full deployment of upward reserves
Information: This could be equal to Regulated Spill, or it could be less than Regulated Spill if flow is allowed to be shifted from Spill to Turbine Release during deployment, or it could be greater than Regulated Spill if the reservoir has a minimum spill requirement as a percentage of total outflow. This depends on how the RPL policy is formulated. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Regulated Spill, then the value of this variable will automatically be set to zero, and the slot should not be referenced in the RPL optimization policy.
Defined by: Regulated Spill with Up Reserve = Regulated Spill + Regulated Spill Change with Up Reserve

and

Regulated Spill with Up Reserve ≤ Regulated Spill Capacity with Up Reserve

Regulated Spill Change with Up Reserve

Type: Series Slot
Units: Flow
Description: The amount Regulated Spill would change to fully deploy all upward reserves
Information: Often this variable will be negative (or 0, and thus the slot Upper Bound will often be set to 0); however it could be positive if the reservoir has a minimum spill requirement as a percentage of total outflow. It could be constrained to be 0, directly or indirectly, by RPL policy, or it could be allowed to be greater than or less than

zero. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Regulated Spill, then the slot should not be referenced in the RPL optimization policy.

Defined by: Regulated Spill with Up Reserve = Regulated Spill + Regulated Spill Change with Up Reserve

Regulated Spill Capacity with Up Reserve

Type: Series Slot
Units: Flow
Description: The maximum Regulated Spill given full deployment of upward reserves
Information: This variable is defined by the Regulated Spill Capacity approximation as a function of Storage with Up Reserve. This slot will get added regardless of which Spill method is selected. Users will not typically need to do anything with this slot.
Defined by: Regulated Spill Capacity with Up Reserve = f(Storage with Up Reserve)

Bypass with Up Reserve

Type: Series Slot
Units: Flow
Description: The amount of Bypass at full deployment of upward reserves
Information: This could be equal to Bypass, or it could be less than Bypass if flow is allowed to be shifted from Bypass to Turbine Release during deployment, or it could be greater than Bypass if the reservoir has a minimum bypass requirement as a percentage of total outflow. This depends on how the RPL policy is formulated. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Bypass, then the value of this variable will automatically be set to zero, and the slot should not be referenced in the RPL optimization policy.
Defined by: Bypass with Up Reserve = Bypass + Bypass Change with Up Reserve

and

Bypass with Up Reserve ≤ Bypass Capacity with Up Reserve

Bypass Change with Up Reserve

Type: Series Slot
Units: Flow
Description: The amount Bypass would change to fully deploy all upward reserves
Information: Often this variable will be negative (or 0, and thus the slot Upper Bound will often be set to 0); however it could be positive if the reservoir has a minimum bypass requirement as a percentage of total outflow. It could be constrained to be 0, directly or indirectly, by RPL policy, or it could be allowed to be greater than or less than zero. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Bypass, then the slot should not be referenced in the RPL optimization policy.
Defined by: Bypass with Up Reserve = Bypass + Bypass Change with Up Reserve

☞ BYPASS CAPACITY WITH UP RESERVE

Type: Series Slot
UNITS: FLOW
Description: The maximum Bypass given full deployment of upward reserves
Information: This variable is defined by the Bypass Capacity approximation as a function of Storage with Up Reserve. This slot will get added regardless of which Spill method is selected. Users will not typically need to do anything with this slot.
Defined by: $\text{Bypass Capacity with Up Reserve} = f(\text{Storage with Up Reserve})$

☞ OUTFLOW WITH UP RESERVE

Type: Series Slot
UNITS: FLOW
Description: The amount of total Outflow at full deployment of upward reserves
Information:
Defined by: $\text{Outflow with Up Reserve} = \text{Turbine Release with Up Reserve} + \text{Spill with Up Reserve}$

☞ STORAGE WITH UP RESERVE

Type: Series Slot
UNITS: VOLUME
Description: The end of timestep Storage if full upward reserves were deployed
Information: This variable definition differs from the standard Storage variable only in the Outflow term. The previous Storage and the Net Inflow are the same. Net Inflow represents all gains and losses, including Inflow, Hydrologic Inflow, Precipitation, Evaporation, etc.
Defined by: $\text{Storage with Up Reserve} = \text{Storage}[t-1] + (\text{NetInflow} - \text{Outflow with Up Reserve}) \times \text{Timestep}$

☞ POOL ELEVATION WITH UP RESERVE

Type: Series Slot
UNITS: LENGTH
Description: The end of timestep Pool Elevation if full upward reserves were deployed
Information: This variable is defined by the Pool Elevation linear approximation as a function of Storage with Up Reserve.
Defined by: $\text{Pool Elevation with Up Reserve} = f(\text{Storage with Up Reserve})$

☞ TAILWATER ELEVATION WITH UP RESERVE

Type: Series Slot
UNITS: LENGTH
Description: The average Tailwater Elevation if full upward reserves were deployed
Information: This variable is defined by the Tailwater Elevation linear approximation as a function of Outflow with Up Reserve and the Tailwater Base Value (if applicable based on the selected Optimization Tailwater method). Tailwater Base Value is unaffected by the Reserves modeling and will be the same for both Tailwater Elevation and Tailwater Elevation with Up Reserves. The Tailwater Elevation linear

approximation that is applied is dependent on the selected method in the Optimization Tailwater category.

Defined by: Tailwater Elevation with Up Reserve = f(Outflow with Up Reserve, Tailwater Base Value)

Temp Tailwater Lookup with Up Reserve

Type: Series Slot
Units: Length
Description: This slot only applies when the Base Value Plus Lookup Table method is selected in the Tailwater category. It represents the lookup value from the Tailwater Table as a function of Outflow with Up Reserve using the Tailwater linear approximation.
Information: This slot will get added regardless of which Tailwater method is selected but will not get used unless the Base Value Plus Lookup method is selected. Users should not need to do anything with this slot.
Defined by: Temp Tailwater Lookup with Up Reserve = f(Outflow with Up Reserve)

PSA Head Factor with Up Reserve

Type: Series Slot
Units: Flow
Description: This slot only applies when the Power Surface Approximation method is selected in the Optimization Power category. It represents the component of Operating Head that is dependent on Storage with Up Reserve and Spill with Up Reserve. It is calculated using the same coefficients used for the Standard PSA Head Factor variable.
Information: This slot will get added regardless of which Optimization Power method is selected but will not get used unless the Power Surface Approximation method is selected. Users should not need to do anything with this slot.
Defined by: PSA Head Adjustment with Up Reserve = f(Storage with Up Reserve, Spill with Up Reserve)

Turbine Release with Down Reserve

Type: Series Slot
Units: Flow
Description: The Turbine Release at full deployment of downward reserves
Information:
Defined by: Turbine Release with Down Reserve = Turbine Release + Turbine Change with Down Reserve

and

Turbine Release with Down Reserve ≤ Turbine Capacity with Down Reserve

TURBINE CHANGE WITH DOWN RESERVE

Type: Series Slot
Units: Flow
Description: The amount Turbine Release would change to fully deploy all downward reserves
Information: This variable is expected to always be negative or zero, and thus the slot configuration Upper Bound should be set to 0.
Defined by:

$$\text{Turbine Release with Down Reserve} = \text{Turbine Release} + \text{Turbine Change with Down Reserve}$$

TURBINE CAPACITY WITH DOWN RESERVE

Type: Series Slot
Units: Flow
Description: The maximum Turbine Release possible given the full deployment of downward reserves
Information: This variable is defined by the Turbine Capacity approximation as a function of Operating Head with Down Reserve.
Defined by:

$$\text{Turbine Capacity with Down Reserve} = f(\text{Operating Head with Down Reserve})$$

OPERATING HEAD WITH DOWN RESERVE

Type: Series Slot
Units: Flow
Description: The Operating Head at full deployment of downward reserves.
Information:
Defined by:

$$\text{Operating Head with Down Reserve} = \frac{\text{Pool Elevation with Down Reserve} + \text{Pool Elevation}[t-1]}{2} - \text{Tailwater Elevation with Down Reserve}$$

SPILL WITH DOWN RESERVE

Type: Series Slot
Units: Flow
Description: The amount of total Spill at full deployment of downward reserves
Information: This could be equal to Spill, or it could be greater than Spill if flow is allowed to be shifted from Turbine Release to Spill during deployment. This depends on how the RPL policy is formulated.
Defined by:

$$\text{Spill with Down Reserve} = \text{Unregulated Spill with Down Reserve} + \text{Regulated Spill with Down Reserve} + \text{Bypass with Down Reserve}$$

UNREGULATED SPILL WITH DOWN RESERVE

Type: Series Slot
Units: Flow
Description: The amount of Unregulated Spill at full deployment of downward reserves
Information: This quantity is defined by the Unregulated Spill linear approximation as a function of Storage with Down Reserve. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Unregulated Spill, then the value of this variable will automatically be set to zero.
Defined by:

$$\text{Unregulated Spill with Down Reserve} = f(\text{Storage with Down Reserve})$$

☞ REGULATED SPILL WITH DOWN RESERVE

Type: Series Slot
UNITS: FLOW
Description: The amount of Regulated Spill at full deployment of downward reserves
Information: This could be equal to Regulated Spill, or it could be greater than Regulated Spill if flow is allowed to be shifted from Turbine Release to Spill during deployment. This depends on how the RPL policy is formulated. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Regulated Spill, then the value of this variable will automatically be set to zero, and the slot should not be referenced in the RPL optimization policy.
Defined by: Regulated Spill with Down Reserve = Regulated Spill + Regulated Spill Change with Down Reserve

and

Regulated Spill with Down Reserve ≤ Reguluated Spill Capacity with Down Reserve

☞ REGULATED SPILL CHANGE WITH DOWN RESERVE

Type: Series Slot
UNITS: FLOW
Description: The amount Regulated Spill would change to fully deploy all downward reserves
Information: This variable will typically by non-negative. It could be constrained to be 0, directly or indirectly, by RPL policy, or it could be allowed to be greater than zero. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Regulated Spill, then the slot should not be referenced in the RPL optimization policy.
Defined by: Regulated Spill with Down Reserve = Regulated Spill + Regulated Spill Change with Down Reserve

☞ REGULATED SPILL CAPACITY WITH DOWN RESERVE

Type: Series Slot
UNITS: FLOW
Description: The maximum Regulated Spill given full deployment of downward reserves
Information: This variable is defined by the Regulated Spill Capacity approximation as a function of Storage with Down Reserve. This slot will get added regardless of which Spill method is selected. Users will not typically need to do anything with this slot
Defined by: Regulated Spill Capacity with Down Reserve = f(Storage with Down Reserve)

☞ BYPASS WITH DOWN RESERVE

Type: Series Slot
UNITS: FLOW
Description: The amount of Bypass at full deployment of downward reserves
Information: This could be equal to Bypass, or it could be greater than Bypass if flow is allowed to be shifted from Turbine Release to Bypass during deployment. This depends on how the RPL policy is formulated. his slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Bypass, then the

value of this variable will automatically be set to zero, and the slot should not be referenced in the RPL optimization policy.

Defined by: $\text{Bypass with Down Reserve} = \text{Bypass} + \text{Bypass Change with Down Reserve}$

and

$\text{Bypass with Down Reserve} \leq \text{Bypass Capacity with Down Reserve}$

Bypass Change with Down Reserve

Type: Series Slot
Units: Flow
Description: The amount Bypass would change to fully deploy all downward reserves
Information: This variable will typically be non-negative. It could be constrained to be 0, directly or indirectly, by RPL policy, or it could be allowed to be greater than zero. This slot will get added regardless of which Spill method is selected. If the selected Spill method does not include Bypass, then the slot should not be referenced in the RPL optimization policy.
Defined by: $\text{Bypass with Down Reserve} = \text{Bypass} + \text{Bypass Change with Down Reserve}$

Bypass Capacity with Down Reserve

Type: Series Slot
Units: Flow
Description: The maximum Bypass given full deployment of downward reserves
Information: This variable is defined by the Bypass Capacity approximation as a function of Storage with Down Reserve. This slot will get added regardless of which Spill method is selected. Users will not typically need to do anything with this slot
Defined by: $\text{Bypass Capacity with Down Reserve} = f(\text{Storage with Down Reserve})$

Outflow with Down Reserve

Type: Series Slot
Units: Flow
Description: The amount of total Outflow at full deployment of downward reserves
Information:
Defined by: $\text{Outflow with Down Reserve} = \text{Turbine Release with Down Reserve} + \text{Spill with Down Reserve}$

Storage with Down Reserve

Type: Series Slot
Units: Volume
Description: The end of timestep Storage if full downward reserves were deployed
Information: This variable definition differs from the standard Storage variable only in the Outflow term. The previous Storage and the Net Inflow are the same. Net Inflow represents all gains and losses, including Inflow, Hydrologic Inflow, Precipitation, Evaporation, etc.
Defined by:

$$\text{Storage with Down Reserve} = \text{Storage}[t-1] + (\text{NetInflow} - \text{Outflow with Down Reserve}) \times \text{Timestep}$$

☞ Pool Elevation with Down Reserve

Type: Series Slot
UNITS: LENGTH
Description: The end of timestep Pool Elevation if full downward reserves were deployed
Information: This variable is defined by the Pool Elevation linear approximation as a function of Storage with Down Reserve.
Defined by: Pool Elevation with Down Reserve = f(Storage with Down Reserve)

☞ Tailwater Elevation with Down Reserve

Type: Series Slot
UNITS: LENGTH
Description: The average Tailwater Elevation if full downward reserves were deployed
Information: This variable is defined by the Tailwater Elevation linear approximation as a function of Outflow with Down Reserve and the Tailwater Base Value (if applicable based on the selected Optimization Tailwater method). Tailwater Base Value is unaffected by the Reserves modeling and will be the same for both Tailwater Elevation and Tailwater Elevation with Up Reserves. The Tailwater Elevation linear approximation that is applied is dependent on the selected method in the Optimization Tailwater category.
Defined by: Tailwater Elevation with Down Reserve = f(Outflow with Down Reserve, Tailwater Base Value)

☞ Temp Tailwater Lookup with Down Reserve

Type: Series Slot
UNITS: LENGTH
Description: This slot only applies when the Base Value Plus Lookup Table method is selected in the Tailwater category. It represents the lookup value from the Tailwater Table as a function of Outflow with Down Reserve using the Tailwater linear approximation.
Information: This slot will get added regardless of which Tailwater method is selected but will not get used unless the Base Value Plus Lookup method is selected. Users should not need to do anything with this slot.
Defined by: Temp Tailwater Lookup with Down Reserve = f(Outflow with Down Reserve)

☞ PSA Head Factor with Down Reserve

Type: Series Slot
UNITS: FLOW
Description: This slot only applies when the Power Surface Approximation method is selected in the Optimization Power category. It represents the component of Operating Head that is dependent on Storage with Down Reserve and Spill with Up Reserve. It is calculated using the same coefficients used for the Standard PSA Head Factor variable.
Information: This slot will get added regardless of which Optimization Power method is selected but will not get used unless the Power Surface Approximation method is selected. Users should not need to do anything with this slot.
Defined by: PSA Head Adjustment with Down Reserve =
f(Storage with Down Reserve, Spill with Down Reserve)

Sample RPL Goal Set Implementation:

Within a RPL Goal set, any constraint that should not be violated when deploying reserves should include a constraint on the standard variable(s) as well as a duplicated constraint on the corresponding reserves variable(s). For example, assume that a reservoir has a maximum pool elevation that cannot be violated when reserves are deployed, and that maximum pool elevation is stored in a series slot on a data object called Res_Data.PE_Max. The RPL goal might look like:

```
REPEATED MAXIMIN
    FOR(DATETIME t IN @”Start Timestep” TO @”Finish Timestep”) DO
        ADD CONSTRAINT Res.Pool Elevation[t] <= Res_Data.PE_Max[t]
        ADD CONSTRAINT Res.Pool Elevation with Down Reserve[t] <= Res_Data.PE_Max[t]
    END FOR
END REPEATED MAXIMIN
```

The reservoir might also have a target elevation constraint that is a lower priority and can be violated by the deployment of reserves. In this case, the target goal would only include a constraint on the Pool Elevation slot but not a constraint on the Pool Elevation with Down Reserve slot.

It is important to note that referencing any of the reserve slots in a RPL policy statement will typically draw a large number of additional variables and constraints into the optimization problem. For performance reasons, the user will probably not want these additional constraints to be added unless they are necessary. Assume, for example, that it is known prior to a run that a particular reservoir is not available to provide reserves during select hours. This may be indicated by setting an input value of 0 MW for those hours in a data object series slot called Res_Data.Down_Reserve_Max (similarly for Up Reserve). It would not be necessary to model the reserve versions of constraints for those time steps because it is already known that reserves will be zero. In this case the goal with the Pool Elevation constraints shown previously might be written as:

```
REPEATED MAXIMIN
    FOR(DATETIME t IN @”Start Timestep” TO @”Finish Timestep”) DO
        ADD CONSTRAINT Res.Pool Elevation[t] <= Res_Data.PE_Max[t]
        IF(Res_Data.Down_Reserve_Max[t] != 0 MW)
            ADD CONSTRAINT Res.Pool Elevation with Down Reserve[t] <=
                                                        Res_Data.PE_Max[t]
        END IF
    END FOR
END REPEATED MAXIMIN
```

In this way, the constraint on Pool Elevation with Down Reserve would only be added when the reservoir is available to provide reserves (the max reserves are not constrained to zero).

Users could then write policy on the reserves to be carried at individual projects. Assume that the data object series slots Res_Data.Up_Reserve_Min and Res_Data.Up_Reserve_Max store the minimum upward reserves that the reservoir must carry and the maximum upward reserves the project can be credited respectively. A goal for the reservoir might be:

```
REPEATED MAXIMIN
      FOR(DATETIME t IN @”Start Timestep” TO @”Finish Timestep”) DO
          IF(Res_Data.Up_Reserve_Max[t] != 0 MW)
                ADD CONSTRAINT Res.Up Reserve[t] >= Res_Data.Up_Reserve_Min[t]
                ADD CONSTRAINT Res.Up Reserve[t] <= Res_Data.Up_Reserve_Max[t]
          END IF
      END FOR
END REPEATED MAXIMIN
```

These two constraints may not necessarily be included in the same goal. Note that the first constraint based on the minimum required reserves will constrain the physical operations of the reservoir in order to meet the required reserve obligation. The second constraint, based on the maximum credited reserves will not directly constraint the physical operations, rather it will only limit the amount of reserves that can be counted. In other words, the solution may operate the reservoir such that, according to physical limits and other specified constraints, the reservoir may have a larger amount of reserves available, but this constraint allows the user to say that only a limited amount will be counted.

Users may then write goals which sum reserves across multiple projects to meet a system reserve requirement.

```
REPEATED MAXIMIN
      FOR(DATETIME t IN @”Start Timestep” TO @”Finish Timestep”) DO
          ADD CONSTRAINT (FOR(OBJECT res IN ProjectsAvailableForUpReserve(t)) SUM
                                        res.Up Reserve[t]
                                END FOR)
                                          >= System_Data.Up_Reserve_Min[t]
      END FOR
END REPEATED MAXIMIN
```

In this goal, ProjectsAvailableForUpReserve represents a user-defined function that returns a list of reservoirs that can carry reserves, again to prevent adding numerous constraints on the reserves version of variables unnecessarily. The details of this function would be model-specific.

5.6.2.23 Startup

This category depends on selecting the Unit Power Table method, **HERE (Section 5.6.2.17.5)**, and describes how the monetary cost associated with starting up or shutting down a unit (turbine) will be modeled. There are two methods in this category, one which does not model these costs (effectively assigning them a value of 0) and one which uses a table describing the combined costs for starting up and shutting down a unit.

5.6.2.23.1 None

This is the default, do-nothing method.

5.6.2.23.2 Unit Lumped Cost Method

For each unit, this method lumps the cost of startup and shutdown into one value. Slots are described **HERE (Objects.pdf, Section 21.1.32.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.6.2.24 Head Loss

This category depends on the Unit Power Table method, **HERE (Section 5.6.2.17.5)**, and contains methods for modeling additional head loss that occurs. This head loss may come from the configuration of the penstocks for bringing water to the turbines.

5.6.2.24.1 None

In this method, there is no additional head loss to be used in the power calculation. In terms of penstock head loss, this method should be selected if the penstocks for the units are independent and the penstock losses are typically incorporated in the power data. Thus the power data is specified in terms of operating head.

5.6.2.24.2 Shared Penstock Head Loss method

In this method, there is additional head loss that results because units share a common penstock. The operating head losses in the penstock depend on the total turbine release and are shared for all units. The net head is calculated by subtracting penstock losses from the operating head. The unit data and power must be specified in terms of unit Net Heads instead of Operating Head. Slots are described **HERE (Objects.pdf, Section 21.1.33.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.6.2.25 Cavitation

This category depends on selecting the Unit Power Table method, **HERE (Section 5.6.2.17.5)**, and contains methods for dealing with the problem of cavitation on turbines. Cavitation is the sudden formation and collapse of low-pressure bubbles in liquids by means of mechanical forces and this process can cause damage to turbines under certain operating conditions.

5.6.2.25.1 None

This is the default, do-nothing method.

5.6.2.25.2 Unit Head and Tailwater Based Regions

This method allows the user to specify the regions of operation in which cavitation does NOT occurs, so that these regions can be avoided. These regions can be dependent on both operating head and tailwater. Slots are described **HERE (Objects.pdf, Section 21.1.34.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.6.2.26 Avoidance Zones

This category depends on selecting the Unit Power Table method, **HERE (Section 5.6.2.17.5)**, and contains methods for modeling the existence of undesirable regions of operation for turbines. There are two methods in this category, one which does not model avoidance zones at all, and one which

5.6.2.26.1 None

This the default, do-nothing method; avoidance zones are not considered.

5.6.2.26.2 Unit Head Based Avoidance Zones

This method allows the user to specify a table that defines the conditions in which the turbines should not be operated. Slots are described **HERE (Objects.pdf, Section 21.1.35.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.6.2.27 Frequency Regulation

This category depends on selecting the Unit Power Table method, **HERE (Section 5.6.2.17.5)**, although in the future it might be enabled for other power methods. The frequency regulation methods model the provision of the frequency regulation ancillary service, that is, how the reservoir can be made available to flexibly follow a load demand within a specified range during a certain period in order to affect the frequency of the generated power.

5.6.2.27.1 None

This is the default, do-nothing method; no regulation is modeled.

5.6.2.27.2 Unit Frequency Regulation

NOTE: THIS METHOD IS NOT YET IMPLEMENTED

When frequency regulation is scheduled, it allows the unit to follow the real time load. Exactly what will happen in real time is unknowable. This results in two sets of values at scheduling time, nominally scheduled power and turbine release. It is uncertain if the real time operators will actually use the service. At present, we distinguish between the nominal "scheduled" power (and turbine release) that the regulation is allowed to depart from and the "expected" power generation (and turbine release) that will take place when regulation is allowed. Both are important. The scheduled power sets the baseline for regulation and should be communicated to the power dispatchers. The expected power and release are more useful for coordinating a plant with the rest of the system. Slots are described **HERE (Objects.pdf, Section 21.1.36.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.6.3 Modeling a Pumped Storage Reservoir

Steps to follow in setting up a pumped power reservoir for optimization

1. Set up a running simulation model, **using methods that are compatible with optimization for Power, Tailwater, Spill, etc.**

2. Chose an Optimization Power method.

3. Choose an Optimization Head Computation Method. In Optimization, None must be chosen.

4. Select the Optimization Spill method corresponding to the method selected in the Spill category.

5. Fill in the LP Param tables related to Power, Turbine Capacity and Best Turbine Flow (if plant power is used). Alternatively, the Power Linearization Automation Method can also be chosen to automatically determine the values for the LP Param tables.

6. Select the Optimization Tailwater method corresponding to the method selected in Tailwater category. If Opt Base Value Plus Lookup Table or Opt Stage Flow Lookup Table is selected, fill in the Tailwater Elevation LP Param table. Alternatively, select a Tailwater Linearization Automation method (Range Input Automation is currently available).

7. If future value is needed, select a method in the Future Value category, followed by the Optimization Future Value category, and if desired, Cumul Stor Val Linearization Automation.

8. Selection of the remaining methods and linearizations can be done in any order. Pool elevation linearizations, Pool Elevation Linearization Automation, Evaporation and Precipitation, Optimize Evaporation Computation, Evaporation Linearization Automations, Bank Storage, Hydrologic Inflow, Energy In Storage, and Diversion from Reservoir. The appropriate data must be entered for each method.

5.7 Reach

In RPL Optimization, a constraint is always created to ensure mass balance. This constraint takes the following form with actual values or expressions comprising the variables shown, being dependent on the configuration of the reach:

$$0 = \Sigma \text{Inflows} - \text{Outflow} + \text{Return Flow} - \text{Diversion} \qquad \text{(EQ 32)}$$

Inflows may include Inflow, Local Inflow, time lagging effects, and GainLoss consideration.

Return Flow and Diversion may be excluded if model configuration does not require the terms' usage.

5.7.1 General Slots

General slots are always present on the object, regardless of selected methods. The following slots are provided on the Reach when the Optimization Controller is selected. They are listed alphabetically. Additional information on general simulation slots, including these slots, can be found in the Reach documentation **HERE (Objects.pdf, Section 22)**.

☞ **INFLOW**

Type:	Agg Series
UNITS:	FLOW
Description:	flow at entrance to the object
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint

☞ **OUTFLOW**

Type:	Agg Series
UNITS:	FLOW
Description:	flow at exit from the object
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint

5.7.2 User Methods in Optimization

5.7.2.1 Diversion from Reach

This category is dependent on selection of one of the following methods in the Routing category: No Routing, Variable Time Lag, Impulse Response, Muskingum, Muskingum Cunge, Muskingum Cunge Improved, Kinematic, Kinematic Improved, or MacCormack. **Please note that of these methods, only "No Routing" is supported by RPL Optimization.**

Of the available methods, "None" and "Available Flow Based Diversion" are supported; selection of any other method will result in an error during run initialization. Please click **HERE (Objects.pdf, Sec-**

tion 22.1.17) for additional information.

5.7.2.1.1 None

This method is the default for this category and should be selected if no diversion is desired.

5.7.2.1.2 Available Flow Based Diversion

In RPL Optimization, this method adds a constraint to the optimization problem. The constraint simply specifies that the value for the Available For Diversion slot is the Upper Bound on Available For Diversion (on the slot configuration) or the Inflow minus any Minimum Diversion Bypass.

SLOTS SPECIFIC TO THIS METHOD

☞ **AVAILABLE FOR DIVERSION**

Type:	Agg Series Slot
UNITS:	FLOW
Description:	the divertable flow in the reach
Information:	
Defined by:	a user-input constant or a function of inflow, depending on model configuration. If a maximum value has been assigned to the Available For Diversion slot, then the Available For Diversion slot is set to that maximum value. Otherwise, if the Min Diversion Bypass category specifies a method other than “None”, the slot is set equal to Inflow - Minimum Diversion Bypass. Otherwise, if a minimum value has been assigned to the Outflow slot, the Available For Diversion slot is set to Inflow - that minimum value. Otherwise, the Available For Diversion slot is set to Inflow.

5.7.2.2 Gain Loss

Of the available methods, “None” and “Constant Gain Loss” are supported; selection of any other method could result in an error during begin run. Please click **HERE (Objects.pdf, Section 22.1.10)** for additional information.

5.7.2.2.1 None

This method is the default for this category and should be used if gains and losses are not required in the model.

5.7.2.2.2 Constant Gain Loss

This method allows the user to specify constant-value parameters for GainLoss. As such, the parameters are used directly in the RPL Optimization problem. No special consideration of slot values is necessary in RPL Optimization. Please click **HERE (Objects.pdf, Section 22.1.10.2)** for additional information.

In defining the Inflows expression described at the beginning of this section, the following formula applies:

Inflows = (Preliminary Inflows * GainLoss Coeff) + GainLoss

Preliminary Inflows may equal Inflow, Inflow + Local Inflow, or appropriate time lagged inflow. (See Routing **HERE (Section 5.7.2.5)** for further detail on the lagged inflow.)

5.7.2.3 Local Inflow and Solution Direction

The reach object considers Local Inflow in its mass balance calculation if specified. In RPL Optimization Local Inflow is included as one of the Inflows if specified. Please click **HERE (Objects.pdf, Section 22.1.3)** for additional information.

5.7.2.4 Min Diversion Bypass

Under certain configurations, the Available Flow Based Diversion method in the Diversion From Reach category may look for the Minimum Diversion Bypass value to calculate Available For Diversion flow. (See Diversion From Reach, above.) If it is desirable that a Minimum Diversion Bypass value be specified, the Input Min Bypass method should be selected. Please click **HERE (Objects.pdf, Section 22.1.18)** for additional information.

If a Min Diversion Bypass method is selected and a value is specified, the Available For Diversion constraint **HERE (Section 5.7.2.1.2)** will include the Minimum Diversion Bypass.

5.7.2.5 Routing

Of the available methods, only No Routing, Time Lag and Impulse Response are supported in RPL optimization; selection of any other method will result in an error during begin run. Please click **HERE (Objects.pdf, Section 22.1.1)** for additional information.

5.7.2.5.1 No Routing

This method provides no routing functionality, but adds consideration of Return Flow to the mass balance.

SLOTS SPECIFIC TO THIS METHOD

☞ **RETURN FLOW**

Type:	Multi Slot
UNITS:	FLOW
Description:	Return flow into the reach
Information:	Enters at the bottom of the reach and so is not lagged by the reach nor is it subject to gains or losses. Return Flow must be defined, either by input, a link or a constraint. It will not default to 0 if undefined.
Defined by:	Explicit Optimization variable in the mass balance constraint:

$$0 = \text{InflowExpression} - \text{Outflow} + \text{ReturnFlow} - \text{Diversion}$$

where Inflow Expression depends on model configuration and may include Inflow, Local Inflow, and GainLoss consideration. Inflow may be linked to a slot on another object.

5.7.2.5.2 Time Lag

In RPL Optimization, Time Lag routing results in a modified mass balance constraint that includes the lagging effect. Selection of this method excludes usage of Return Flow and Diversion values in the optimization problem. Routing occurs prior to any gain and loss consideration.

$$\sum \text{Inflows} = (\text{Inflow}(\text{lagINT} - 1) + \text{LocalInflow}(\text{lagINT} - 1)) \times \text{flowFraction1}$$
$$+ (\text{Inflow}(\text{lagINT}) + \text{LocalInflow}(\text{LagINT})) \times \text{flowFraction2}$$

where:

lagINT refers to the timestep on or after the point in time equal to the current timestep minus LagTime.

lagINT - 1 refers to the timestep prior to LagINT.

flowFraction1 equals the fraction of a timestep used to weigh the inflow associated with lagINT - 1; flowFraction1 equals the fractional remainder of the LagTime / timestep (e.g. 0.7 if LagTime = 1.7 timesteps and 0 if LagTime is a multiple of the timestep).

flowFraction2 equals the fraction of a timestep used to weigh the inflow associated with lagINT; flowFraction2 equals 1 - flowFraction1. (e.g. 0.3, if LagTime = 1.7 timesteps and 1 if LagTime is a multiple of the timestep).

SLOTS SPECIFIC TO THIS METHOD

☞ **LAGTIME**

Type: Table Slot
UNITS: TIME
Description: lag time or travel time of flow through the reach
Information: a single value
Defined by: user-input

5.7.2.5.3 Impulse Response

Impulse Response routing results in a modified mass balance constraint where the reach Outflow is the weighted sum of the current and previous timesteps' Inflows. Selection of this method excludes usage of Return Flow and Diversion values in the optimization problem. Any gains or losses are added to or subtracted from the routed flow.

$$\text{Outflow} = C_0\text{Inflow}_t + C_1\text{Inflow}_{t-1} +$$
$$\ldots + C_{\text{ncoeff}-2}\text{Inflow}_{t-\text{ncoeff}-2} + C_{\text{ncoeff}-1}\text{Inflow}_{t-(\text{ncoeff}-1)} + \text{TotalGainLoss}$$

SLOTS SPECIFIC TO THIS METHOD

☞ **LAG COEFF**
Type: Table Slot
UNITS: NO UNITS
Description: impulse response coefficients
Information: There must be the same number of values in the Lag Coeff table as the value given in Num of Coeff
Defined by: user-input

☞ **NUM. OF COEFF**
Type: Scalar Slot
UNITS: NO UNITS
Description: number of impulse response coefficients
Information: There must be the same number of values in the Lag Coeff table as the value given in Num of Coeff
Defined by: user-input

5.7.3 Numerical Approximation Approaches

No Reach slots require Numerical Approximation.

5.7.4 Modeling a Reach

To use a Reach object in optimization, set up a running simulation model using methods that are compatible with Optimization for the following categories: Routing, Local Inflow and Solution Direction, Gain Loss, Min Diversion Bypass and Diversion from Reach. Input the required data for each method.

5.8 Slope Power Reservoir

The Sloped Power Reservoir is similar to the Level Power Reservoir but with added functionality to model the “wedge” storage or backwater storage effects of a sloped water surface. Additional information can be found **HERE (Objects.pdf, Section 23)**.

5.8.1 General Slots

General slots are always present on the object, regardless of selected methods. The following slots are provided on the reservoir when the optimization controller is selected.

☞ BACKWATER ELEVATION

Type:	Agg Series
UNITS:	LENGTH
Description:	Elevation of the reservoir at the top of the dam

☞ CANAL FLOW

Type:	Agg Series
UNITS:	FLOW
Description:	Flow into (out of) the reservoir from (to) a canal
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint

☞ DIVERSION

Type:	Series Slot
UNITS:	FLOW
Description:	Flow from the reservoir to a diverting object
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint

☞ ENERGY

Type:	Agg Series Slot
UNITS:	ENERGY
Description:	Product of the power generated by flow through the turbines and the length of the timestep.
Information:	
Defined by:	Replaced by Power * Timestep Length

☞ FLOW FROM PUMPED STORAGE

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow into the reservoir from a pumped storage reservoir
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint. This slot should be linked to Outflow on a Pumped Storage object. The Pumped Storage object constrains its Outflow.

☞ FLOW TO PUMPED STORAGE

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow out of the reservoir into a pumped storage reservoir
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint. This slot should be linked to Pumped Flow on a Pumped Storage object.

☞ INFLOW

Type:	MultiSlot
UNITS:	FLOW
Description:	Inflow into the reservoir from upstream
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint

☞ INFLOW 2

Type:	Agg Series
UNITS:	FLOW
Description:	An additional inflow into the reservoir

☞ MAX TURBINE Q

Type:	Table Slot
UNITS:	LENGTH VS FLOW
Description:	Table relating Operating Head to maximum Turbine Capacity
Information:	See power methods for more information

☞ OPERATING HEAD

Type:	Agg Series Slot
UNITS:	LENGTH
Description:	Elevation difference between the average Pool Elevation and the average Tailwater Elevation during a timestep
Information:	
Defined by:	Replacement by (Pool Elevation(t) + Pool Elevation(t-1)) / 2 - Tailwater Elevation

☞ OUTFLOW

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Outflow from the reservoir
Information:	
Defined by:	Explicit Optimization variable as Outflow = Turbine Release + Spill

☞ Plant Power Table

Type:	Table
Units:	Length vs Flow vs Power
Description:	3D Table relating Operating Head, Turbine Release, and Power
Information:	See power methods for more information

☞ Pool Elevation

Type:	Agg Series Slot
Units:	Length
Description:	Elevation of the water surface of the Reservoir at the dam
Information:	When Pool Elevation is a part of the optimization problem, as it is in all conceivable

☞ Power

Type:	Agg Series Slot
Units:	Power
Description:	Power generated by flow through the turbines
Information:	
Defined by:	Numerical 3-D Approximation in terms of Operating Head and Turbine Release. Approximation is based on the Plant Power Table. The Power LP Param table contains a value for Operating Head used to index the Operating Head column of the Plant Power Table. This approximated value, therefore, reduces the Power to a function of Turbine Release at the given Operating Head. The flow values in the Power LP Param table are then used as approximation points indexing the Turbine Release column of the Plant Power Table. The Plant Power Table should have increasing values of Operating Head and Turbine Release. Power should be a concave function of Operating Head, but concavity is not strictly enforced; mild non-concave regions are permissible to allow for round-off error, etc. The preferred order of approximation is substitution, piece-wise, two-point line, tangent.

☞ Return Flow

Type:	MultiSlot
Units:	Flow
Description:	Flow returning from a diversion object
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint (see Storage)

☞ Spill

Type:	Agg Series Slot
Units:	Flow
Description:	Sum of the Bypass, Regulated Spill and Unregulated Spill
Information:	
Defined by:	Explicit Optimization variable as Spill = Bypass + Regulated Spill + Unregulated Spill

☞ **STORAGE**

Type:	Agg Series Slot
UNITS:	VOLUME
Description:	Volume of water stored in the reservoir
Information:	
Defined by:	Explicit Optimization variable as Storage = Storage(t-1) + Precipitation Volume - Evaporation - Change in Bank Storage + timestep * (Inflow + Inflow2 Canal Flow + Flow TO Pumped Storage + Hydrologic Inflow Net + Return Flow - (Outflow + Diversion + Flow FROM Pumped Storage))

☞ **TAILWATER BASE VALUE**

Type:	Agg Series Slot
UNITS:	LENGTH
Description:	Elevation of tailwater or base elevation used to compute elevation of tailwater
Information:	
Defined by:	Explicit Optimization variable should be input or linked. Related constraints can be found **HERE (Optimization.pdf, Section 5.8.2.22)**Tailwater Elevation Numerical Approximation discussion, or on other objects to which the slot is linked.

☞ **TAILWATER ELEVATION**

Type:	Agg Series Slot
UNITS:	LENGTH
Description:	Water surface elevation on the downstream side of the dam
Information:	
Defined by:	Various approaches dependent on the method selected in the Optimization Tailwater category.

☞ **TURBINE CAPACITY LP PARAM**

Type:	Table
UNITS:	LENGTH, LENGTH, LENGTH
Description:	LP Param table for turbine capacity
Information:	see power methods for more information

☞ **TURBINE RELEASE**

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow through the turbines of a power reservoir
Information:	
Defined by:	Explicit Optimization variable as Turbine Release <= Power Plant Cap Fraction * Turbine Capacity

5.8.2 User Methods in Optimization

The following categories and methods are available for use in Optimization. Because of dependency relationships, you may not be able to see them in your model until you change other methods. In the discussion below, you will see the other methods that need to be selected to enable a given method to be used in your model. When building a model, you will wish to review this to ensure that required dependencies are satisfied so that the desired methods are available for use.

5.8.2.1 Bank Storage

Not all methods are functional in RPL optimization. Of the available methods, "None" and "CRSS Bank Storage" are supported.

5.8.2.1.1 None

Click **HERE (Objects.pdf, Section 23.1.26.1)** for more information.

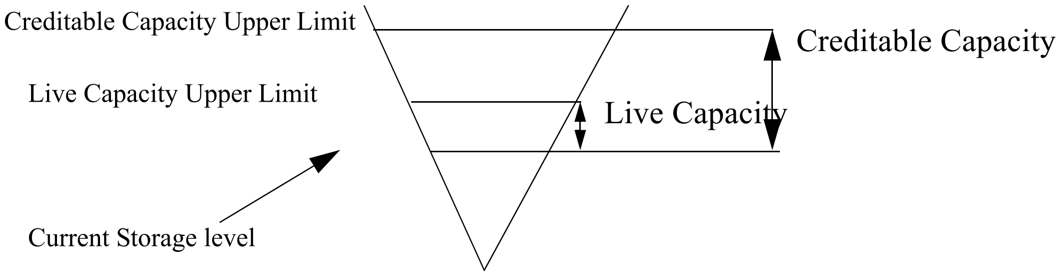
Input Bank Storage

5.8.2.1.2 CRSS Bank Storage

Click **HERE (Objects.pdf, Section 23.1.26.3)** for more information.

5.8.2.2 Creditable Capacity Available

The creditable capacity category conceptually represents the total amount of available storage space above the current storage in the reservoir. As the pool storage in the reservoir increases, the creditable capacity decreases.



5.8.2.2.1 None

If this method is selected creditable capacity is not calculated for the reservoir.

5.8.2.2.2 Constraint and Variable

If this method is selected an additional decision variable and physical constraint are added to the optimization problem:

SLOTS SPECIFIC TO THIS METHOD:

☞ **CREDITABLE CAPACITY**

Type:	Gassers
UNITS:	VOLUME
Description:	The creditable capacity is the total amount of storage space available above the current storage
Information:	
Defined by:	Explicit Optimization Variable included in the Optimization Problem as part of the constraint

Max Storage >= Storage + Creditable Capacity. **(EQ 33)**

Max Storage for the reservoir is entered in the Upper Bound field of the Storage slot configuration dialog box.

5.8.2.3 Cumul Stor Val Linearization Automation

Appearance of this category is dependent on selecting the Opt Cumulative Storage Value Table method for the Optimization Future Value category (which in turn is dependent on the Cumulative Storage Value Table method being selected for the Future Value category).

This category allows the optimization to automate the creation of the Cumulative Storage Value Table, and the selection of linearization points and linearization method for the Cumulative Storage Value slot. Actual usage of these values occurs in the **HERE (Section 5.8.2.12)** Optimization Future Value category's Opt Cumulative Storage Value Table method.

5.8.2.3.1 None

If this method is selected, no automation will be performed.

5.8.2.3.2 Marginal Value to Table and Lin

This method uses information from the simulation slot Marginal Storage Value Table to generate the Cumul Stor Val Table, select linearization points, and choose a linearization method. The cumulative storage value in the Cumul Stor Val Table can be thought of as the summation of the marginal storage values from a storage of 0 to the current storage.

As an illustration of the automation procedure, consider the following Marginal Storage Value Table:

Marginal Storage Value Table:

Storage	Marginal Value
20	30
60	26
100	24

To parameterize the Cumul Stor Val Table, the automation proceeds as follows:

1) The first row receives a Storage value of 0.

Cumul Stor Val Table

Storage	Cumulative Value
0	

2) For each row i in the Marginal Storage Table, not including the last row, the average Storage (i.e. the midpoint) of row i and row i + 1 is assigned as Storage in each successive row of the Cumul Stor Val Table. For example, row 2 Storage equals the average of 20 and 60; row 3 Storage equals the average of 60 and 100.

Cumul Stor Val Table

Storage	Cumulative Value
0	
40	
80	

3) The last row of the Cumul Stor Val Table receive a Storage value equal to the maximum storage associated with the Storage slot’s configuration.

Cumul Stor Val Table

Storage	Cumulative Value
0	
40	
80	
140	

4) Returning to the first row of the Cumul Storage Val Table, a Cumulative Value of 0 is assigned.

Cumul Stor Val Table

Storage	Cumulative Value
0	0
40	

Storage	Cumulative Value
80	
140	

5) For each successive row j = 2, 3, 4 in the Cumul Stor Val Table and corresponding row i = 1, 2, 3 in the Marginal Storage Value Table, the Cumulative Value equals the Storage from row j - 1 + (the change in Storage of row j from row j - 1) * the Marginal Value from row i. For example, row 2 Cumulative Value equals 1200 calculated from 0 + (40 - 0) * 30; row 3 Cumulative Value equals 2240 calculated from 1200 + (80 - 40) * 26; row 4 Cumulative Value equals 3680 calculated from 2240 + (140 - 80) * 24.

Cumul Stor Val Table:

Storage	Cumulative Value
0	0
40	1200
80	2240
140	3680

The Cumul Stor Val LP Param table is then built using Storage values from the Cumul Stor Val Table:

Cumul Stor Val LP Param

Tangent	Line	piecewise
	0	0
	140	40
		80
		140

SLOTS SPECIFIC TO THIS METHOD

There are no slots specific to this method as it uses the slots in the Opt Cumulative Storage Value Table method. However, for clarity in the discussion above, the slots are reshown here.

☞ CUMUL STOR VAL TABLE

Type:	Table Slot
UNITS:	VOLUME VS. VALUE
Description:	The estimated total economic value of water stored in the reservoir for discrete storage values.
Information:	
Defined by:	Automated procedure described above.

☞ **CUMUL STOR VAL LP PARAM**

Type:	Table Slot
UNITS:	VOLUME, VOLUME, VOLUME
Description:	Specifies the storage points used to take the tangent, line and piecewise approximations for Cumul Stor Val Table linearization
Information:	
Defined by:	Automated procedure described above.

☞ **MARGINAL STORAGE VALUE TABLE**

Type:	Table Slot
UNITS:	STORAGE VS. $ VALUE
Description:	Anticipated Storage versus worth of Cumulative Storage per unit energy
Information:	This table should be increasing in storage, and usually decreasing in marginal value
Defined by:	Required input

5.8.2.4 Diversion from Reservoir

Not all methods in this category are supported in RPL optimization. Of the available methods, “None” and “Available Flow Based Diversion” are supported; selection of any other method will result in an error during begin run.

5.8.2.4.1 None

Click **HERE (Objects.pdf, Section 21.1.26.1)** for more information.

5.8.2.4.2 Available Flow Based Diversion

Click **HERE (Objects.pdf, Section 21.1.26.2)** for more information.

5.8.2.5 Energy in Storage

In Optimization, currently “None” and “EIS Table Lookup” are supported.

5.8.2.5.1 None

No Energy in storage is considered.

5.8.2.5.2 EIS Table Lookup

With this method selected, Energy in Storage is considered as a function of Pool Elevation. Click **HERE (Objects.pdf, Section 23.1.6.2)** for more information on the simulation method. If the optimization problem uses Energy In Storage, either directly or indirectly, then this slot is added to the problem. Before being passed to the optimization solver, this slot is numerically approximated as a function of Pool Elevation (Numerical 2-D Approximation). The relationship between Pool Elevation and Energy In Storage will come from the user-input Energy In Storage Table. The table will be queried either using user-input points defined in the Energy In Storage LP Param table or using automatically calculated points, depending on method selection in the Pool Elevation Linearization Automation category. If the selected method in the Pool Elevation Linearization Automation is “none”, then the user-input Energy In Storage LP Param table values are used. For other Pool Elevation Linearization Automation methods (Initial Target Input, Min Difference or Range Input, Min Difference) the same points automatically developed for the Pool Elevation linearization will be used.

RELATED SLOTS

☞ ENERGY IN STORAGE

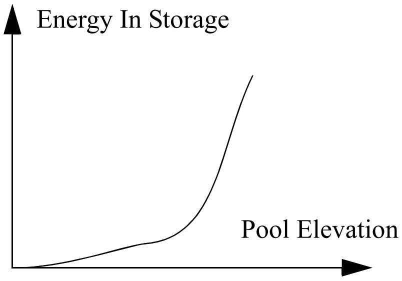
Type:	Agg Series Slot
UNITS:	ENERGY VS. POWER
Description:	Energy in Storage in the reservoir
Information:	
Defined by:	Numerical 2-D Approximation in terms of Pool Elevation, based upon the Energy In Storage Table. The Energy In Storage LP Param table values are used as approximation points indexing the Energy In Storage Table. The Energy In Storage Table should have increasing values of Pool Elevation and Energy In Storage. Energy In Storage is required to be a convex function of Pool Elevation. The preferred order of approximation is substitution, piece-wise, tangent, two-point line.

☞ ENERGY IN STORAGE LP PARAM

Type:	Table Slot
UNITS:	LENGTH
Description:	Specifies the Pool Elevation points used to take the tangent, line and piecewise approximations for Energy In Storage linearization.
Information:	This table is used for linearization unless the Pool Elevation Linearization Automation category has a method selected other than “none”.
Defined by:	User-input

☞ **ENERGY IN STORAGE TABLE**

Type:	Table Slot
UNITS:	LENGTH VS. ENERGY
Description:	Table defining the relationship between Energy In Storage and Pool Elevation
Information:	
Defined by:	User-input



Energy in Storage pool elevation relationship. Not drawn to scale.

5.8.2.6 Slope Storage

This category is used to specify how the wedge is to be modeled. Click **HERE (Objects.pdf, Section 23.1.18)** for more information.

5.8.2.7 Slope Storage Coefficients

This category is used to specify whether to use Weighting Coefficients or Impulse Response coefficients to model the wedge. Both methods are supported in RPL optimization. Click **HERE (Objects.pdf, Section 23.1.19)** for more information.

5.8.2.8 Optimization Backwater

5.8.2.8.1 Lambda Method

This is the default method and is the only one supported in RPL optimization. The documentation is under development

5.8.2.8.2 Independent Linearizations

This method is not supported in RPL optimization

5.8.2.9 Optimization Evaporation

This category can be used to model evaporation and precipitation.

5.8.2.9.1 None

The Optimization Evaporation method “None” is the default method for this category. It does no calculations and requires that “None” be selected for the Evaporation and Precipitation category.

5.8.2.9.2 Opt Input Evaporation

This method is analogous to the Input Evaporation method in simulation and requires the Input Evaporation method to be selected for the Evaporation and Precipitation category. Evaporation Rate and Precipitation Rate are entered as a time series. Evaporation is calculated as a product of Evaporation Rate, Average Surface Area over the timestep and Timestep length. Similarly Precipitation Volume is calculated as the product of Precipitation Rate, Average Surface Area and Timestep length.

Evaporation and Precipitation are calculated based on a static Volume-Surface Area relationship. In reality the Volume-Surface Area relationship may not be static for a slope storage reservoir, so results from the Opt Input Evaporation method should be treated with caution when used on a slope power reservoir object.

Note: The linearization of the Surface Area variable can result in a small approximation error in optimization for Evaporation and Precipitation Volume. This means there can be a small difference between the mass balances in the optimization solution and the post-optimization rulebased simulation when using this method. It is important to use care when setting the approximation points for Surface Area in order to reduce this approximation error. Refer to the information on the Surface Area LP Param slot below. Also caution should be used if applying this method at a monthly timestep. All rates in the optimization mass balance are converted to monthly volumes based on a 30-day month, regardless of the month. This will also produce a difference between the mass balances in the optimization solution and the post-optimization rulebased simulation for a *monthly* timestep.

SLOTS SPECIFIC TO THIS METHOD

☞ ELEVATION AREA TABLE

Type:	Table Slot
UNITS:	LENGTH VS. AREA
Description:	Represents the Elevation-Surface Area relationship
Information:	This table must be input. It is used to derive the Volume Area Table.
Defined by:	User-input

☞ EVAPORATION

Type:	Series Slot
UNITS:	VOLUME
Description:	The volume of water lost to evaporation over the timestep
Information:	If this slot contains user input, it is added directly to the mass balance constraint, otherwise it is defined by the expression below.
Defined by:	Either user-input or the following constraint:

$$\text{Evaporation} = \text{EvaporationRate} \times \frac{\text{SurfaceArea}(t-1) + \text{SurfaceArea}}{2} \times \text{Timestep}$$

☞ EVAPORATION RATE

Type:	Series Slot
UNITS:	VELOCITY
Description:	The rate, in length per time, at which water is lost to evaporation at each timestep
Information:	This slot can be set as user input. If it is not set as an input, and if Evaporation is not an input, this slot defaults to zero. If Evaporation is an input, this slot is not used.
Defined by:	User-input or defaults to zero

☞ PRECIPITATION RATE

Type:	Series Slot
UNITS:	VELOCITY
Description:	The rate, in length per time, at which water is gained from precipitation at each timestep
Information:	This slot can be set as user input. If it is not set as an input, it defaults to zero.
Defined by:	User-input or defaults to zero

☞ PRECIPITATION VOLUME

Type:	Series Slot
UNITS:	VOLUME
Description:	The volume of water gained from precipitation over the timestep
Information:	The Input Evaporation method will not allow this slot to be set as an input.
Defined by:	Explicit constraint:

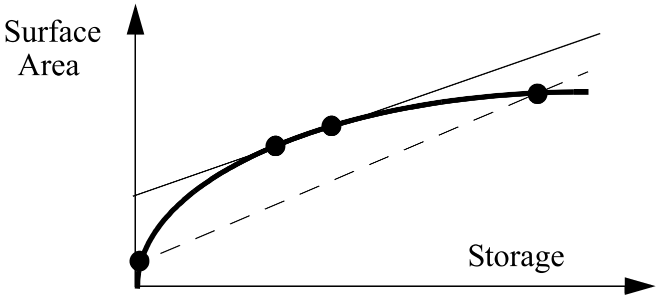
$$\text{Precipitation Volume} = \text{PrecipitationRate} \times \frac{\text{SurfaceArea}(t-1) + \text{SurfaceArea}}{2} \times \text{Timestep}$$

☞ SURFACE AREA

Type:	Series Slot
UNITS:	AREA
Description:	The area of the water surface at the end of the timestep
Information:	This slot is numerically approximated as a function of Storage (Numerical 2-D Approximation). The relationship between Pool Elevation and Storage comes from the automatically generated Volume Area Table. The table is queried using the user-input points defined in the Surface Area LP Param table.
Defined by:	Numerical 2-D Approximation in terms of Storage, based upon the Volume Area Table. The Surface Area LP Param table values are used as approximation points indexing the Volume Area Table. The preferred order of approximation is substitution, piecewise, two-point line, tangent. Most often the two-point line (secant) approximation will be used.

☞ SURFACE AREA LP PARAM

Type:	Table Slot
UNITS:	VOLUME
Description:	Specifies the Storage points used to take the tangent, line and piecewise approximations for Surface Area linearization
Information:	The best Storage point to choose for tangent approximation would be the expected storage during the run; for line approximation, the expected maximum and minimum Storage; for piecewise approximation, use points that cover the full range of expected Storage during the run with intermediate points such that a piecewise linear curve reasonably approximates the actual curve.In most cases, the line (secant) approximation will be used. It is important to set these points carefully to minimize approximation error.
Defined by:	User-input



Surface Area vs. Storage relationship with two alternative line approximations.
The figure is not drawn to scale

The figure above represents two alternative selections of points for the line approximation in the Surface Area LP Param table. The linear approximation represented by the dashed line corresponds to the selection of points near the extremes of the Volume Area table. This approximation will tend to result in an under-estimation of Surface Area, and thus an under-estimation of Evaporation and Precipitation Volume. Evaporation losses in the post-optimization rulebased simulation would be greater than the losses approximated in the optimization solution. The linear approximation represented by the solid line corresponds to the selection of points closer together in the Volume Area Table, and is more similar to the Tangent approximation. This approximation would tend to result in an over estimation of Surface Area, and the losses due to Evaporation in the post-optimization rulebased simulation would be less than the approximation in the optimization solution.

☞ VOLUME AREA TABLE

Type:	Table Slot
UNITS:	VOLUME VS. AREA
Description:	Represents the Storage Volume-Surface Area relationship
Information:	This table is read-only and is automatically generated at the start of the run from the Elevation Area Table and the Elevation Volume table. The method starts by copying

	the Elevation Area Table and then replaces the Pool Elevation Values with the corresponding Storage values. The Storage values are linearly interpolated based on the Elevation Volume Table
Defined by:	Automatically generated

5.8.2.10 Evaporation Linearization Automation Category

This category allows the approximation points for surface area to be automatically generated. This category requires a method other than "None" to be selected for the Optimization Evaporation category.

5.8.2.10.1 None

This is the default method for this category. There is no approximation point automation. The user will be required to enter approximation points in the Surface Area LP Param slot manually.

5.8.2.10.2 Use Elevation Approximation Points

This method automates the approximation points in the Surface Area LP Param table slot. It copies the same storage values used in the Pool Elevation LP Param slot. If this method is selected it will overwrite any values that have been entered manually in the Surface Area LP Param slot. This method may provide a good starting point for establishing the Surface Area approximation points; however in some cases, it may be important for the user to adjust these points manually (see details **HERE (Section 5.8.2.9.2)** under Surface Area LP Param).

5.8.2.11 Future Value

This category and its methods are not dependent on other method selections.

5.8.2.11.1 None

Click **HERE (Objects.pdf, Section 23.1.12.1)** for more information.

5.8.2.11.2 Cumulative Storage Value Table

In RPL-Optimization, the Cumulative Storage Value is Numerically Approximated as described below.

Click **HERE (Objects.pdf, Section 23.1.12.2)** for more information.

SLOTS SPECIFIC TO THIS METHOD

☞ CUMULATIVE STORAGE VALUE

Type: Agg Series Slot
UNITS: $
Description: Represents the future energy value of the current storage
Information:
Defined by: 2-D approximation in terms of Anticipated Storage, based upon the Cumul Stor Val Table. The Cumul Stor Val LP Param table values (Storage) are used as approximation points indexing the Cumul Stor Val Table. The Cumul Stor Val Table should have increasing values of Storage and Cumulative Value. Cumulative Storage Value is required to be a concave function of Anticipated Storage. The preferred order of approximation is substitution, piece-wise, tangent, two-point line. The Cumul Stor Val Linearization Automation category's Marginal Value to Table and Lin method can automate creation of the Cumul Stor Val LP Param table and the Cumul Stor Val Table.

☞ ANTICIPATED STORAGE

Type: Agg Series Slot
UNITS: VOLUME
Description: The combination of the actual storage plus water that would be expected to enter the reservoir after the Current Timestep but has not yet, due to lagging.
Information:
Defined by:

☞ CUMUL STOR VAL TABLE

Type: Table Slot
UNITS: VOLUME VS. VALUE
Description: The estimated total economic value of water stored in the reservoir for discrete storage values.
Information:
Defined by: User-input or by automated procedure if Cumul Stor Val Linearization Automation category has selected the Marginal Value to Table and Lin method.

☞ MARGINAL STORAGE VALUE TABLE

Type: Table Slot
UNITS: STORAGE VS. $ VALUE
Description: Anticipated Storage versus Cumulative Storage Value per unit energy
Information: This table should be increasing in storage, and logically decreasing in marginal value
Defined by: Required input

☞ **SPILL COST**
Type: Agg Series Slot
UNITS: $
Description:
Information:
Defined by:

5.8.2.12 Optimization Future Value

This category allows the optimization to provide slots relating to the future value of water. Its appearance is dependent on selecting the Cumulative Storage Value Table method for the Future Value category.

5.8.2.12.1 None

If this method is selected, the future value slots will not be visible, and no linearization will be attempted.

5.8.2.12.2 Opt Cumulative Storage Value Table

If this method is selected, the following slots will be visible, and linearization will be allowed.

SLOTS SPECIFIC TO THIS METHOD

☞ **CUMUL STOR VAL LP PARAM**
Type: Table Slot
UNITS: VOLUME, VOLUME, VOLUME
Description: Specifies the storage points used to take the tangent, line and piecewise approximations for Cumul Stor Val Table linearization
Information:
Defined by: User-input or by automated procedure if Cumul Stor Val Linearization Automation category has selected the Marginal Value to Table and Lin method.

5.8.2.13 Hydrologic Inflow

If any method in this category is selected, hydrologic inflow is included in the reservoir mass balance. Optimization assumes hydrologic inflow to be known (data) and it is not solved for by the reservoir regardless of the method selected. Click **HERE (Objects.pdf, Section 23.1.15)** for more information.

5.8.2.14 Live Capacity Available

The live capacity category conceptually represents the amount of available storage space between the current storage in the reservoir and a user defined maximum. As the pool storage in the reservoir increases, the live capacity decreases.

5.8.2.14.1 None

If this method is selected live capacity is not calculated for the reservoir.

5.8.2.14.2 Constraint and Variable

If this method is selected an additional decision variable and physical constraint are added to the optimization problem. See the figure in the Creditable Capacity Category.

☞ **LIVE CAPACITY**

Type:	Agg Series Slot
UNITS:	VOLUME
Description:	The amount of storage space available between the current storage and a user defined upper limit. This upper limit is entered in the Upper Bound field of the Live Capacity slot configuration dialogue box.
Defined by:	Explicit constraint

$$\text{Max Live Capacity} >= \text{Storage} + \text{Live Capacity} \qquad \text{(EQ 34)}$$

5.8.2.15 Optimization Spill

The Optimization Spill methods determine how spill is calculated for the reservoir and generates physical constraints that correspond to the selected methods.

This category is dependent on selection of the Independent Linearizations method for the Optimization Power category. However, the method selected in the Optimization Spill category must match with the corresponding non-optimization method in the Spill category.

Spill is an Optimization decision variable. The following constraint is always generated for a reservoir:

$$\text{Outflow} = \text{Turbine release (or Release)} + \text{Spill} \qquad \text{(EQ 35)}$$

The spill methods generate values applicable to an additional constraint:

$$\text{Spill} = \text{Regulated Spill} + \text{Unregulated Spill} + \text{Bypass}, \qquad \text{(EQ 36)}$$

where some of these terms may be omitted if they do not apply to the selected spill method.

Depending on the method selected, some of the following slots will be added. Other slots will be used from the non-optimization method selected. Please click **HERE (Objects.pdf, Section 23.1.8)** for details on non-optimization Spill methods.

As applicable, the following constraints are also added:

$$\text{Bypass} <= \text{Bypass Capacity} \qquad \text{(EQ 37)}$$

$$\text{Regulated Spill} <= \text{Regulated Spill Capacity} \qquad \text{(EQ 38)}$$

$$\text{Unregulated Spill} = \text{Unregulated Spill Capacity} \qquad \text{(EQ 39)}$$

☞ BYPASS CAPACITY

Type:	Series Slot
UNITS:	FLOW
Description:	Bypass capacity
Information:	
Defined by:	Numerical 2-D Approximation in terms of storage, based upon the Bypass Capacity Table.

☞ BYPASS CAPACITY TABLE

Type:	Table Slot
UNITS:	VOLUME VS. FLOW
Description:	Storage vs. corresponding maximum bypass spill values
Information:	
Defined by:	Internally developed based on Bypass Table and Elevation Volume Table relationships. For each pool elevation in the Bypass Table, the Bypass Capacity Table has a row relating Storage to Bypass Capacity. The Pool Elevations are converted to Storage using the Elevation Volume Table.

☞ REGULATED SPILL CAPACITY

Type:	Series Slot
UNITS:	FLOW
Description:	Regulated spill capacity
Information:	
Defined by:	Numerical 2-D Approximation in terms of storage, based upon the Regulated Spill Capacity Table.

☞ REGULATED SPILL CAPACITY TABLE

Type:	Table Slot
UNITS:	VOLUME VS. FLOW
Description:	Storage vs. corresponding maximum regulated spill values
Information:	
Defined by:	Internally developed based on Regulated Spill Table and Elevation Volume Table relationships. For each pool elevation in the Regulated Spill Table, the Regulated Spill Capacity Table has a row relating Storage to Regulated Spill Capacity. The Pool Elevations are converted to Storage using the Elevation Volume Table.

☞ REGULATED SPILL OR BYPASS LP PARAM

Type:	Table Slot
UNITS:	VOLUME
Description:	Specifies the Storage points use to take the tangent, line and piecewise approximations for Regulated Spill Capacity linearization and Bypass Spill Capacity linearization.
Information:	
Defined by:	User-input

☞ UNREGULATED SPILL LINEARIZATION TABLE

Type:	Table Slot
UNITS:	VOLUME VS. FLOW
Description:	Storage vs. corresponding unregulated spill values
Information:	
Defined by:	Internally developed based on Unregulated Spill Table and Elevation Volume Table relationships. For each pool elevation in the Unregulated Spill Table, the Unregulated Spill Capacity Table has a row relating Storage to Unregulated Spill Capacity. The Pool Elevations are converted to Storage using the Elevation Volume Table.

☞ UNREGULATED SPILL LP PARAM

Type:	Table Slot
UNITS:	VOLUME
Description:	Specifies the Storage points use to take the tangent, line and piecewise approximations for Unregulated Spill Linearization Table linearization
Information:	
Defined by:	User-input

5.8.2.15.1 None

If this method is selected the slot bounds in the slot configuration dialog are set to zero. No additional constraints are generated.

5.8.2.15.2 Opt Monthly Spill

This method is not functional in RPL Optimization.

This method sets the lower and upper bounds on spill. The lower bound is set to zero and the default upper bound is set to a very big number (9,999,999 cms). The default upper bound can be revised in the Spill slot configuration, Upper Bound parameter. No additional constraints are generated beyond these bounds.

5.8.2.15.3 Opt Unregulated

If this method is selected only unregulated spill is considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Unregulated Spill} \tag{EQ 40}$$

5.8.2.15.4 Opt Regulated

When this method is selected only regulated spill is considered. The lower bound on spill is set to zero and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} \tag{EQ 41}$$

5.8.2.15.5 Opt Regulated and Unregulated

When this method is selected unregulated and regulated spill are considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} + \text{Unregulated Spill} \tag{EQ 42}$$

5.8.2.15.6 Opt Regulated and Bypass

When this method is selected only regulated and bypass spill are considered and the lower bound on spill is set to zero and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} + \text{Bypass} \tag{EQ 43}$$

5.8.2.15.7 Opt Regulated, Bypass and Unregulated

When this method is selected unregulated, regulated and bypass spill are considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} + \text{Unregulated Spill} + \text{Bypass} \tag{EQ 44}$$

5.8.2.15.8 Opt Bypass, Regulated and Unregulated

When this method is selected unregulated, regulated and bypass spill are considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Bypass} + \text{Regulated Spill} + \text{Unregulated Spill} \tag{EQ 45}$$

5.8.2.16 Pool Elevation Linearization Automation

This category is no longer supported in RPL optimization.

5.8.2.17 Optimization Head Computation

The methods in this category are no longer supported in RPL optimization. Appearance of this category is dependent on selection of the Independent Linearizations method for the Optimization Power category.

5.8.2.18 Optimization Power

The Optimization Power category allows the user to select which type of linearizations will be used for linearizing various slots. For Independent Linearizations all slots are linearized separately according to the user specified methods. This category has no dependencies.

5.8.2.18.1 None

This is the default method. It is an error to have this method selected for Optimization. No slots are added for this method.

5.8.2.18.2 Independent Linearizations

The Independent Linearizations method linearizes all the variables separately according to the user selected methods. The variables that need to be linearized vary greatly depending on the other methods selected. See the various Categories and Linearization Approaches for details.

5.8.2.18.3 Power Coefficient

The Power Coefficient method models Power at each time step as Turbine Release multiplied by a Power Coefficient Estimate.

SLOTS SPECIFIC TO THIS METHOD

☞ **POWER COEFFICENT ESTIMATE**

Type:	Series Slot
UNITS:	POWERPERFLOW
Description:	This represents the estimated Power Coefficient that gets used in the Optimization definition of Power.
Information:	The Power Coefficient Estimate is a required input for the run. It can either be input directly (manually or by DMI), or it could be set by an initialization rule. For example, an initialization rule could set the Power Coefficient Estimate based on a “Seed” (Slot Cache) value. If the Power Coefficient method is selected, and Power Coefficient Estimate is not an input for all time steps, then the run will abort with an error message.
Defined by:	Input

Power at each time step is then defined as:

$$\text{Power} = \text{Turbine Release} \times \text{Power Coefficient Estimate}$$

5.8.2.18.4 Power Surface Approximation

This method allows for the modeling of dynamic Operating Head to be incorporated into Optimization Power modeling. It provides a significant improvement in the Power approximation error over the linearization using the Power LP Param table, which assumes a constant Operating Head over the entire run period. The improvement is especially significant for projects that exhibit a wide range of Operating Head over the course of the run, particularly if the optimization policy is trying to maximize Power or set Power greater than or equal to a value.

The method introduces a new variable, PSA Head Factor, which represents the weighted contributions of Storage, Spill and Tailwater Base Value to Operating Head. The method generates a set of planes, the Power Surface, to constrain power as a function of Turbine Release and the PSA Head Factor.

The Power Surface can be thought of as pavement over a warped bridge that is level on one end and sloped and higher on the other end. The bridge ends represent zero Turbine Release and maximum Turbine Release respectively. In addition, the bridge is bowed concave across the center line everywhere except the level (zero Turbine Release) end. This bowing reflects Power as a function of the Head Factor for a fixed value of Turbine Release. The edges of the bridge represent Power as a function of Turbine Release for the low and high values of the Head Factor. Additional points defining a grid on the surface correspond to intermediate values of Turbine Release and the Head Factor. The user can define dimensions and points used for this grid. A piecewise linear surface is composed of cutting planes with each plane defining one triangle on the surface. A surface underneath the bridge composed of two planes defines a lower bound on Power given Turbine Release and Head Factor values.

Because only two planes can be defined for the lower bounds on power, additional approximation error can be introduced if the optimization policy has an incentive to minimize power. In these cases the optimization solution can have an incentive to under-approximate Power for a given Turbine Release, and thus the Post-optimization Rulebased Simulation will calculate a larger Power value than the optimization value for Power.

SLOTS SPECIFIC TO THIS METHOD

The user enters data into three slots:

PSA Sample Input,

PSA Head Factor Grid Lines

PSA Turbine Release Grid Lines

The remaining table slots associated with the method are automatically populated by RiverWare. A description of all of the slots associated with this method is given below.

☞ PSA SAMPLE INPUT

Type:	Table Slot
UNITS:	FLOW, VOLUME, FLOW, LENGTH
Description:	This table contains user-specified sample values for Turbine Release, Storage, Total Spill (if applicable) and Tailwater Base Value (if applicable) that are used by RiverWare to compute the PSA Head Factor Weights. The four parameters in this table represent all of the parameters that contribute to Operating Head. In other words, if a value is known for each of these parameters, it is possible to calculate the corresponding Operating Head.
Information:	The values in this table should span the full possible range for each parameter for the given run. For example, the Turbine Release column should contain a value of 0, the highest possible turbine release from the reservoir, and any other intermediate values specified by the user. Specifying tighter upper and lower bounds for each parameter will improve the approximation, but it is important that all possible values for each variable within the run are spanned by the range for that parameter in the table. The user can determine how many rows to include in the table. It is expected that typically, 3-4 values will be specified for each parameter. An example is shown

below. The Spill column should only contain values if a method other than None has been selected in the Spill category. Otherwise the Spill column should be left empty (will display “NaN”). The Tailwater Base Value column should only contain values if the reservoir’s Tailwater Base Value slot is linked to the Pool Elevation slot of a downstream reservoir (i.e. if the Downstream Reservoir Pool Elevation contributes to the calculation of Operating Head). Otherwise the Tailwater Base Value column should be left empty.

Defined by: User Input

Turbine Release	Storage	Spill (Total)	Tailwater Base Value
0	0	0	500
100	1000	50	510
200	2000	100	520

PSA HEAD FACTOR GRID LINES

Type: Scalar Slot

UNITS: NONE

Description: The number of PSA Head Factor values for each Turbine Release value used when calculating the PSA Grid Points

Information: This must be an integer greater than or equal to 2. The value should typically range from 2 to 4. A larger number will improve the approximation but will increase run time.

Defined by: User Input

PSA TURBINE RELEASE GRID LINES

Type: Scalar Slot

UNITS: NONE

Description: The number of Turbine Release values for each PSA Head Factor value used when calculating the PSA Grid Points

Information: This must be an integer greater than or equal to 2. The value should typically range from 2 to 4. A larger number will improve the approximation but will increase run time.

Defined by: User Input

PSA Sample Output

Type: Table Slot
Units: Flow, Volume, Flow, Volume, Power
Description: A table containing all permutations of the values entered in the PSA Sample Input table slot along with the Power calculated for each combination of Sample Input values. This sample output is used to calculate the PSA Head Factor Weights.
Information: RiverWare will populate this table at the beginning of the run. For the PSA Sample Input table shown above, with three values in each of the four columns, this table would contain 81 rows, one for each permutation. For a given combination of Turbine Release, Storage, Spill and Tailwater Base Value, RiverWare can calculate the corresponding Power. The third column will contain the Downstream Storage values corresponding to the Tailwater Base Value entries in the PSA Sample Input slot.
Defined by: Populated by RiverWare at the beginning of the run

For each row *i* in the table:

$$\text{SamplePower}_i = f(\text{SampleTurbineRelease}_i, \text{SampleStorage}_i, \text{SampleSpill}_i, \text{SampleTailwaterBaseValue}_i)$$

The calculation of SamplePower depends on the selected Power and Tailwater methods.

PSA Head Factor Weights

Type: Table Slot
Units: PerTime, None, PerTime
Description: The weights for Storage, Spill and Downstream Storage used when calculating PSA Head Factor
Information: This is a 1 x 3 table with a single weight for Storage, Spill and Downstream Storage. The weights are calculated by RiverWare by performing a regression on the values in the PSA Sample Outputs table slot. If one of the parameters is not used (contains NaN for all rows in the PSA Sample Input table), the corresponding weight in this table slot will be zero.
Defined by: Calculated by regression at the beginning of the run

PSA Head Factor

Type: Series Slot
Units: Flow
Description: This slot is a variable in the optimization solution and represents the weighted contribution of Storage, Spill and Downstream Storage to Operating Head and thus power. It can be thought of as a storage analog of Operating Head.
Information: This variable is used in the automatically generated constraints that define Power. It will not, generally display any values, but the user can write a post-optimization rule that calculates the value of this variable at each time step using the equation shown below and the values in the PSA Head Factor Weights table slot.
Defined by: $\text{PSA Head Factor} = \alpha \cdot \text{Storage} + \beta \cdot \text{Spill} + \gamma \cdot \text{Downstream Storage}$

The coefficients α, β and γ come from the PSA Head Factor Weights table slot.

PSA GRID POINTS

Type: Table Slot
UNITS: FLOW, VOLUME, FLOW, VOLUME, POWER, FLOW
Description: Points used to calculate the planes for the PSA Max Constraints and PSA Min Constraints, one row for each point with columns for Turbine Release, Storage, Spill, Downstream Storage, Power and PSA Head Factor.
Information: The number of rows (points) in this table equals the product of the number of lines in the PSA Head Factor Grid Lines and PSA Turbine Release Grid Lines scalar slots. For each grid point, RiverWare calculates the value of Power and the PSA Head Factor corresponding to the parameter values in that row.
Defined by: Populated by RiverWare at the beginning of the run

Turbine Release Column: Smallest and largest values from PSA Sample Input and evenly spaced intermediate values corresponding to the number of lines in the PSA Turbine Release Grid Lines scalar slot

Storage, Spill and Downstream Storage Columns: Smallest and largest values from PSA Sample Input and evenly spaced intermediate values corresponding to the number of lines in the PSA Head Factor Grid Lines scalar slot

Power Column: Determined by selected Power method, for each row *i* in the table

$$\text{Power}_i = f(\text{TurbineRelease}_i, \text{Storage}_i, \text{Spill}_i, \text{DownstreamStorage}_i)$$

Head Factor Column: $\text{HeadFactor}_i = \alpha \cdot \text{Storage}_i + \beta \cdot \text{Spill}_i + \gamma \cdot \text{DownstreamStorage}_i$

The coefficients α, β and γ come from the PSA Head Factor Weights table slot.

PSA MAX CONSTRAINTS

Type: Table Slot
UNITS: FLOW, FLOW, FLOW, FLOW, FLOW, FLOW, POWERPERFLOW, POWERPERFLOW, POWER
Description: This table contains the planes that define the upper bounds for power as a function of Turbine Release and PSA Head Factor, one row for each plane. Power is constrained to be less than or equal to the planes defined in this table. The first six columns contain three pairs of Turbine Release and PSA Head Factor values that define three points on the plane. The remaining three columns contain the corresponding Turbine Release Coefficient, Head Factor Coefficient and Constant Term that define the same plane and are used in the actual constraint expressions.
Information: The points used to define the planes come from the Turbine Release and Head Factor points in the PSA Grid Points table slot. RiverWare will not use all of the possible combinations of Turbine Release and Head Factor points but rather will only use the combinations necessary to define the Power Surface.
Defined by: Populated by RiverWare at the beginning of the run

For each row *i* in the PSA Max Constraints table slot a constraint is added to the optimization problem:

$$\text{Power} \le u_i \cdot \text{Turbine Release} + v_i \cdot \text{PSA Head Factor} + w_i$$

The coefficients u_i, v_i and w_i are the Turbine Release Coefficient, Head Factor Coefficient and Constant Term from row *i* of the PSA Max Constraints table. Turbine Release and PSA Head Factor are the actual variables in the optimization problem.

☞ PSA MIN CONSTRAINTS

Type:	Table Slot
UNITS:	FLOW, FLOW, FLOW, FLOW, FLOW, FLOW, POWERPERFLOW, POWERPERFLOW, POWER
Description:	This table contains the planes that define the lower bounds for power as a function of Turbine Release and PSA Head Factor, one row for each plane. Power is constrained to be greater than or equal to the planes defined in this table. The first six columns contain three pairs of Turbine Release and PSA Head Factor values that define three points on the plane. The remaining three columns contain the corresponding Turbine Release Coefficient, Head Factor Coefficient and Constant Term that define the same plane and are used in the actual constraint expressions
Information:	This table will always contain two rows. Only two planes can be defined for the lower bounds on power.
Defined by:	Populated by RiverWare at the beginning of the run

For each row *i* in the PSA Min Constraints table slot a constraint is added to the optimization problem:

$$\text{Power} \geq u_i \cdot \text{Turbine Release} + v_i \cdot \text{PSA Head Factor} + w_i$$

The coefficients u_i, v_i and w_i are the Turbine Release Coefficient, Head Factor Coefficient and Constant Term from row *i* of the PSA Min Constraints table. Turbine Release and PSA Head Factor are the actual variables in the optimization problem.

METHOD SUMMARY

Three slots require input from the user. Descriptions of these slots are provided above.

PSA Sample Input

PSA Turbine Release Grid Points

PSA Head Factor Grid Points

Then at the start of the run RiverWare carries out the following steps:

- Populate the PSA Sample Output Table with all permutations of parameters in the PSA Sample Input table slot
- Calculate the PSA Head Factor Weights using a regression based on values in the PSA Sample Output table slot
- Add defining constraints for PSA Head Factor using the PSA Head Factor Weights
- Populate the PSA Grid Points table slot based on the number of grid lines specified by the user
- Generate upper bound planes using the PSA Grid Points and add the corresponding constraints on Power to the Optimization problem

- Generate lower bound planes using the PSA Grid Points and add the corresponding constraints on Power to the Optimization problem

METHOD DETAILS

This section provides details of the calculations carried out by RiverWare to generate the constraints that define the Power Surface. This material is included for reference only.

1. RiverWare populates the PSA Sample Output table slot. All permutations of values in the PSA Sample Input table slot are determined, and a row for each is added to the PSA Sample Output table slot. RiverWare may make some adjustments to prevent infeasible combinations. For example, the highest Turbine Release value may not be possible for all combinations of Storage, Spill and Tailwater Base Value. RiverWare will adjust the Turbine Release to use the maximum turbine release for the resulting Operating Head. The table is organized in blocks of constant Turbine Release, with the lowest Turbine Release block first and then increasing Turbine Release. The values in the remaining columns are ordered corresponding to increasing Operating Head. Storage is increasing. Spill and Tailwater Base Value (or Downstream Storage) are decreasing. For each combination of values (each row), RiverWare calculates the corresponding Operating Head and Power. The Power value is added to each row in the PSA Sample Output table. The details of these calculations are shown here for a given row *i* in the PSA Sample Output table. Note that Power_i is the only calculated quantity displayed in the table. Values from the intermediate calculations shown below are not displayed in the table.

$$\text{PoolElevation}_i = f(\text{Storage}_i)$$

Pool Elevation is calculated by interpolation on the reservoir Elevation Volume Table.

If Tailwater Base Value is linked to Pool Elevation of a downstream reservoir:

$$\text{TailwaterBaseValue}_i = \text{DownstreamPoolElevation}_i = f(\text{DownstreamStorage}_i)$$

Tailwater Base Value is calculated by interpolation on the downstream reservoir Elevation Volume Table. Tailwater Base Value is only calculated from the Downstream Storage if either Base Value Plus Lookup Table, Stage Flow Lookup Table or Coefficients Table is the selected Tailwater method *and* Tailwater Base Value is linked to the Pool Elevation of the downstream reservoir. Otherwise Tailwater Base Value is unused, and thus Operating Head and Power are not dependent on Downstream Storage.

$$\text{Outflow}_i = \text{TurbineRelease}_i + \text{Spill}_i$$

Spill will only be non-zero if a method other than None is selected in the Spill category.

$$\text{TailwaterElevation}_i = f(\text{Outflow}_i, \text{TailwaterBaseValue}_i)$$

The tailwater calculation is dependent on the selected method in the Tailwater category.

$$\text{OperatingHead}_i = \text{PoolElevation}_i - \text{TailwaterElevation}_i$$

$$\text{Power}_i = f(\text{TurbineRelease}_i, \text{OperatingHead}_i)$$

The Power calculation is dependent on the selected method in the Power category.

2. RiverWare calculates the PSA Head Factor Weights using a combination of linear regression and averages.

FOR EACH Turbine Release value *i*

FOR EACH combination of Spill and Tailwater Base Value (or Downstream Storage), *k*, perform a linear regression to calculate a temporary Storage coefficient g_{ik}:

$$g_{ik} = \frac{n\sum(\mathrm{Power}_j \times \mathrm{Storage}_j) - \left(\left(\sum \mathrm{Power}_j\right)\left(\sum \mathrm{Storage}_j\right)\right)}{n\left(\sum \mathrm{Storage}_j^2\right) - \left(\sum \mathrm{Storage}_j\right)^2}$$

where *n* is the number of Storage values.

Calculate the temporary Storage coefficient for the given Turbine Release, g_i:

$$g_i = \frac{\sum g_{ik}}{n}$$

where *n* is the number of Spill and Downstream Storage) combinations.

FOR EACH combination of Storage and Downstream Storage, *k*, perform a linear regression to calculate a temporary Spill coefficient e_{ik}:

$$e_{ik} = \frac{n\sum(\mathrm{Power}_j \times \mathrm{Spill}_j) - \left(\left(\sum \mathrm{Power}_j\right)\left(\sum \mathrm{Spill}_j\right)\right)}{n\left(\sum \mathrm{Spill}_j^2\right) - \left(\sum \mathrm{Spill}_j\right)^2}$$

where *n* is the number of Spill values.

Calculate the temporary Spill coefficient for the given Turbine Release, e_i:

$$e_i = \frac{\sum e_{ik}}{n}$$

where *n* is the number of Storage and Downstream Storage combinations.

FOR EACH combination of Storage and Spill, *k*, perform a linear regression to calculate a temporary Downstream Storage coefficient f_{ik}:

$$f_{ik} = \frac{n\sum(\mathrm{Power}_j \times \mathrm{DownstreamStorage}_j) - \left(\left(\sum \mathrm{Power}_j\right)\left(\sum \mathrm{DownstreamStorage}_j\right)\right)}{n\left(\sum \mathrm{DownstreamStorage}_j^2\right) - \left(\sum \mathrm{DownstreamStorage}_j\right)^2}$$

where *n* is the number of Downstream Storage values.

Calculate the temporary Downstream Storage coefficient for the given Turbine Release, f_i:

$$e_i = \frac{\sum e_{ik}}{n}$$

where *n* is the number of Storage and Spill combinations.

Calculate average ratios over all Turbine Release values:

$$c = \frac{\sum (e_i / g_i)}{n}$$

$$d = \frac{\sum (f_i / g_i)}{n}$$

where *n* is the number of Turbine Release values.

Calculate the final Head Factor Weights:

$$\text{Storage Weight} = \frac{1}{c}$$

$$\text{Spill Weight} = -1$$

If Tailwater Base Value is linked to the downstream Pool Elevation:

$$\text{Downstream Storage Weight} = \frac{d}{c}$$

These three weight values are displayed in the three columns of the PSA Head Factor Weights table slot.

In the optimization solution the PSA Head Factor is defined at a given time step by

If Tailwater Base Value is linked to the downstream Pool Elevation:

$$\text{PSA Head Factor} = \\ \text{StorageWeight} \times \text{Storage} + \text{SpillWeight} \times \text{Spill} + \text{DownstreamStorageWeight} \times \text{Downstream Storage}$$

If Tailwater Base Value is not linked:

$$\text{PSA Head Factor} = \\ \text{StorageWeight} \times \text{Storage} + \text{SpillWeight} \times \text{Spill}$$

3. RiverWare calculates the grid points for the PSA Grid Points table slot. The number of rows (points) in this table equals the product of the number of lines in the PSA Head Factor Grid Lines and PSA Turbine Release Grid Lines scalar slots. The smallest and largest values in the PSA Sample Input slot lead to the smallest and largest values for each of the parameters that affect power (Turbine Release, Storage, Spill and Downstream Storage. One row in this table will correspond to the smallest Turbine Release, smallest Storage, largest Spill and largest Downstream Storage, and thus smallest power. Another row will correspond to the largest Turbine Release, largest Storage, smallest Spill and smallest Downstream Storage, and thus the largest Power. RiverWare generates evenly spaced values between the extremes for each variable.Within the PSA Grid Points table, for a given Turbine Release value, Storage is increasing and Spill and Downstream Storage are decreasing. For each grid point, RiverWare calculates the value of Power and the PSA Head Factor corresponding to the parameter values in each row and adds these to the final two columns in the table.

For each row *i* in the table:

$$\text{Power}_i = f(\text{TurbineRelease}_i, \text{Storage}_i, \text{Spill}_i, \text{DownstreamStorage}_i)$$

The Power calculation depends on the method selected in the Power category.

$$\text{HeadFactor}_i = \alpha \cdot \text{Storage}_i + \beta \cdot \text{Spill}_i + \gamma \cdot \text{DownstreamStorage}_i$$

The coefficients α, β and γ come from the PSA Head Factor Weights table slot.

4. RiverWare calculates the upper bound planes. These are entered in the PSA Max Constraints table slot, one row for each plane, where the planes represent Power as a function of Turbine Release and PSA Head Factor. For each combination of three points in the PSA Grid Points Table, RiverWare calculates the corresponding plane. For any three points, the plane is defined by three linear equations, which can be represented in matrix form.

$$\bar{P} = \mathbf{A}\bar{u}$$

$$\bar{P} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} QT_1 & HF_1 & 1 \\ QT_2 & HF_2 & 1 \\ QT_3 & HF_3 & 1 \end{bmatrix}, \qquad \bar{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

The coefficients that define the plane are solved for by

$$\bar{u} = \mathbf{A}^{-1}\bar{P}$$

RiverWare only uses the planes that are necessary to define the Power Surface. Each plane is checked against all remaining points in the PSA Grid Points table. If the Power value for any of the remaining points lies above the given plane, then that plane is rejected (i.e. it would over-constrain Power, resulting in an under-estimation of Power). For each plane *i* that is used, RiverWare adds the following constraint to the optimization problem for each time step:

$$\text{Power} \le u_i \cdot \text{Turbine Release} + v_i \cdot \text{PSA Head Factor} + w_i$$

5. RiverWare calculates the lower bound planes. These are entered in the PSA Min Constraints table slot, one row for each plane. Only two planes can be calculated for the lower bounds. The points used for the two planes are as follows:

Plane 1:

Point A1 - Min Turbine Release, Max Head Factor

Point B1 - Max Turbine Release, Max Head Factor

Point C1 - Min Turbine Release, Min Head Factor

Plane 2:

Point A2 - Min Turbine Release, Max Head Factor

Point B2 - Max Turbine Release, Max Head Factor

Point C2 - Max Turbine Release, Max Head Factor

The coefficients defining the planes are calculated using the same matrix calculation described for the upper bound planes. For each lower bound plane *i*, RiverWare adds the following constraint to the optimization problem for each time step:

$$\text{Power} \geq u_i \cdot \text{Turbine Release} + v_i \cdot \text{PSA Head Factor} + w_i$$

A note on the lower bound approximation:

Because only two planes can be defined for the lower bounds on power, additional approximation error can be introduced if the optimization policy has an incentive to minimize power. This can occur if Power is included in a Minimize objective. It can also occur if the optimization problem contains constraints that require Power to be less than or equal to value. Note that these types of constraints can be introduced through multiple forms of RPL optimization goals. The first is basic less than or equal to constraint.

ADD CONSTRAINT $\text{Res.Power}[t] \leq \text{Value}$

Policy that requires Power to be equal to a value also adds a less than or equal to constraint. The following RPL statement:

ADD CONSTRAINT $\text{Res.Power}[t] = \text{Value}$

actually adds the following two constraints to the optimization problem:

$$\text{Res.Power}[t] \leq \text{Value}$$

$$\text{Res.Power}[t] \geq \text{Value}$$

A constraint to require the sum of Power from multiple reservoirs to be less than or equal to a value will have a similar effect. A final manner in which this can occur is through policy that minimizes Power indirectly through user-defined variables. For example, assume a user-defined variable is defined by the following RPL constraint:

ADD CONSTRAINT $\text{Res.Power}[t] + \text{UserVariable}[t] = \text{Value1}$

Then later the variable is constrained by

ADD CONSTRAINT $\text{UserVariable}[t] \geq \text{Value2}$

An equivalent constraint would be

ADD CONSTRAINT $\text{Res.Power}[t] \leq \text{Value1} - \text{Value2}$

It is possible to reduce the approximation error in these cases by adding constraints to the RPL Optimization Goal Set that essentially create more restrictive planes for the lower bound. These types of constraints are model-specific. Contact CADSWES if assistance is required to add more restrictive lower bounds to the Power Surface Approximation.

5.8.2.19 Power Linearization Automation

The Power Linearization Automation category allows the selection of the approximation points used for linearizing the power related slots to be done automatically or entered by the user.

It is dependent on selection of Independent Linearizations for the Optimization Power category and selection of None or Variable Head for the Optimization Head Computation category. (The Variable Head method is not currently functional in RPL Optimization.)

5.8.2.19.1 None

The method requires the user to input the approximation points for the power, best turbine flow (depending on the power method selected) and turbine capacity into the appropriate Power LP Param, Best Turbine Flow LP Param and Turbine Capacity LP Param tables, respectively.

5.8.2.19.2 Plant Automation

This method's availability for user selection is dependent on selection of Plant Power Coefficient in the Power category.

This method proceeds as follows, beginning with the Turbine Capacity LP Param table:

1) In the Max Turbine Q table, get the head values in the first row and the second to last row of the table, calling them FIRSTHEAD and LASTHEAD respectively.

2) Calculate AVEHEAD as the average of FIRSTHEAD and LASTHEAD.

3) Find the maximum value in the Turbine Capacity column (MaxTurbineCapacity) and its corresponding Operating Head (MaxTCOperatingHead) in the Max Turbine Q table. Given the nature of this table, the maximum value is not necessarily the last value. Also, if successive rows have the same Turbine Capacity, the first one (and its corresponding Operating Head) is chosen.

4) If maxTCOperatingHead > LASTHEAD - 0.5 m, then maxTCOperatingHead is set to LASTHEAD - 1 m.

The Turbine Capacity LP Param table is then defined using the variables above: The tangent approximation value is set as AVEHEAD. The line approximation points are set to FIRSTHEAD + 0.5 m and LASTHEAD - 0.5 m. The piecewise approximation points are set to FIRSTHEAD + 0.5, MaxTCOperatingHead, and LASTHEAD - 0.5.

The method then repeats the process using the Best Turbine Q table. (In this table the second column is now called Best Capacity whereas the second column in the Max Turbine Q table is called Turbine Capacity.

Next the Power LP Param table is filled in:

1) If the initial Operating Head is not known, it is calculated as initial Pool Elevation - initial Tailwater Elevation. The Elevation Volume Table and initial Storage will be used to determine initial Pool elevation, if necessary. The initial Operating Head is called MIDHEAD.

2) The Maximum Turbine Q table is queried using MIDHEAD to determine the corresponding

maximum turbine flow, here called FLOWMAX.

3) The Best Turbine Q table is queried using MIDHEAD to determine the corresponding best turbine flow. FLOWBEST is set to this best turbine flow - 0.01 cms. If FLOWBEST is greater than or equal to FLOWMAX, then reset FLOWBEST to FLOWMAX - 0.01.

The Power LP Param table is then defined using the variables above: The Operating Head value is set to MIDHEAD. The tangent approximation value is set as (FLOWBEST + FLOWMAX) / 2. The line approximation points are set to 0 and (FLOWBEST + FLOWMAX) / 2. The piecewise approximation points are set to 0, FLOWBEST, and FLOWMAX.

Now the method creates the Plant Power Table (defining power as a function of operating head and turbine release). The method adds the best and max power generation (power and flow) points for each operating head by combining the data in the Best Turbine Q, Best Power Coefficient, Max Turbine Q, and Max Power Coefficient tables. The details of the algorithm are as follows. For each row (each Operating Head) in the Max Turbine Q table:

1. Determine the Operating Head (OPHEAD) and Turbine Capacity (MAXQ) values.

2. Query the Best Turbine Q table using OPHEAD to determine Best Turbine Flow (BESTQ).

3. Query the Best Power Coefficient table using the Operating head to determine the Best Power Coefficient.

4. Query the Max Power Coefficient table using the Operating head to determine the Max Power Coefficient.

5. Query the Best Power Coefficient table using the Operating head to determine the Best Power Coefficient.

6. Calculate the MAXPOWER as MAXQ * Max Power Coefficient.

7. Calculate the BESTPOWER as BESTQ * Best Power Coefficient.

8. Create a row in the table using OPHEAD, 0, 0 (Operating Head, Turbine Release, and Power columns, respectively).

9. If BESTQ > 0, then create a row in the table using OPHEAD, BESTQ, BESTPOWER.

10. If MAXQ does not equal BESTQ, then create a row in the table using OPHEAD, MAXQ, MAXPOWER.

11. If MAXQ does not equal MaxTurbineCapacity (retained from the Turbine Capacity LP Param table work above), then create a row in the table using OPHEAD, MaxTurbineCapacity, MAXPOWER.

Finally, the initial (timestep) value of the Power Coefficient slot is assigned as determined by querying the Best Power Coefficient table using MIDHEAD (retained from the Power LP Param table work above) for Operating Head.

5.8.2.19.3 Peak Automation

This method's availability for user selection is dependent on selection of Peak Power in the Power category.

This method proceeds as follows, beginning with the Turbine Capacity LP Param table:

1. The Number of Units table is queried. Its value is herein called NUMUNITS.

2. Get the head values in the first row and the second to last row of the Best Generator Flow table, calling them FIRSTHEAD and LASTHEAD respectively.

3. Calculate AVEHEAD as the average of FIRSTHEAD and LASTHEAD.

4. Find the maximum value of flow in the Type #1 Flow column (MaxFlow) and its corresponding Operating Head (HEADMAXQ) in the Best Generator Flow table. Given the nature of this table, the maximum flow value is not necessarily the last value. Also, if successive rows have the same Type #1 Flow, the first one (and its corresponding Operating Head) is chosen.

5. If HEADMAXQ > LASTHEAD - 0.5 m, then HEADMAXQ is set to LASTHEAD - 1.0 m.

6. Calculate ALLMAXFLOW as MaxFlow * NUMUNITS.

7. The Turbine Capacity LP Param table is then defined using the variables above: The tangent approximation value is set as AVEHEAD. The line approximation points are set to FIRSTHEAD + 0.5 m and LASTHEAD - 0.5 m. The piecewise approximation points are set to FIRSTHEAD + 0.5 m, HEADMAXQ, and LASTHEAD - 0.5 m.

Next the Power LP Param table is filled in:

1. If the initial Operating Head is not known, it is calculated as initial Pool Elevation - initial Tailwater Elevation. The Elevation Volume Table and initial Storage will be used to determine initial Pool elevation, if necessary. The initial Operating Head is called MIDHEAD.

2. The Best Generator flow is queried using MIDHEAD to determine the corresponding Type #1 Flow, here called FLOWMAX.

3. Calculate ALLFLOWMAX as FLOWMAX * NUMUNITS - 0.01 cms (where NUMUNITS is retained from the Turbine Capacity LP Param table work above).

4. Calculate ALLFLOWBEST as ALLFLOWMAX - 0.01 cms.

The Power LP Param table is then defined using the variables above: The Operating Head value is set to MIDHEAD. The tangent approximation value is set as the average of ALLFLOWBEST and ALLFLOWMAX. The line approximation points are set to 0 and the average of ALLFLOWBEST and ALLFLOWMAX. The piecewise approximation points are set to 0 cms, ALLFLOWBEST and ALLFLOWMAX.

Now the method creates the Plant Power Table (defining power as a function of operating head and turbine release). For each row (each Operating Head) in the Best Generator Flow table:

1. Determine the Operating Head (OPHEAD) and Type #1 Flow (BESTQ) values.

2. Query the Best Generator Power table using OPHEAD to determine Type #1 Power (BESTPOWER).

3. Calculate ALLBESTQ = BESTQ * NUMUNITS (retained from the Turbine Capacity LP Param table work above).

4. Calculate ALLBESTPOWER = BESTPOWER * NUMUNITS (retained from the Turbine Capacity LP Param table work above).

5. Create a row in the Plant Power Table with OPHEAD, 0, 0 (Operating Head, Turbine Release, and Power columns, respectively).

6. If ALLBESTQ > 0, the create a row in the table using OPHEAD, ALLBESTQ, ALLBESTPOWER.

7. If ALLBESTQ does not equal ALLMAXFLOW (retained from the Turbine Capacity LP Param table work above), then create a row in the table using OPHEAD, ALLMAXFLOW, and ALLBESTPOWER.

Finally, the method checks that the MIDHEAD value assigned as Operating Head in the Power LP Param table is contained within the Operating Head range of the Plant Power Table. If it is not, and error occurs.

5.8.2.20 Power

For more information on the simulation methods, click **HERE (Objects.pdf, Section 23.1.1)**. Only a subset of available Simulation methods are available in Optimization. Selection of a method other than those that follow will result in an error.

If the Optimization problem uses Power, either directly or indirectly, then this slot is added to the problem. Before being passed to the optimization solver, this slot is numerically approximated as a function of Operating Head and Turbine Release (Numerical 3-D Approximation). The relationship between Operating Head and Turbine Release will come from the Plant Power Table. This table will either be user-input, or automatically parameterized depending on the method selection in the Power Linearization Automation category. The table will be queried using the points defined in the Power LP Param table. The values in the Power LP Param table also will either be user-input or automatically calculated points, depending on the method selection in the Power Linearization Automation category. If the selected method in the Power Linearization Automation is “none”, then the tables should contain user-input values which are used. For other Power Linearization Automation methods (Plant Automation or Peak Automation) the tables will contain automatically calculated values which are used.

RELATED SLOTS

☞ **PLANT POWER TABLE**

Type: Table Slot
UNITS: LENGTH VS. FLOW VS. POWER
Description: Table defining the relationship between Operating Head, Turbine Release and Power at selected points of operation. In building the Plant Power table, the following

approach might be taken. For a given Operating Head create n + 2 rows of data, where n is the number of units comprising the Plant. The first row will reflect no flow through the turbines. The next row will reflect the preferred level of flow through the first unit normally activated. Subsequent rows will reflect incremental flow levels as additional turbines come on line at their respective preferred levels of flow. The final row will reflect the flow of all available units running at maximum flow capacity.

Information: This table must be user input unless the Power Linearization Automation category has selected either Plant Automation or Peak Automation methods.

Defined by:

POWER

Type: Agg Series Slot

UNITS: POWER

Description: Power generated by flow through the turbines

Information:

Defined by: Numerical 3-D Approximation in terms of Operating Head and Turbine Release. Approximation is based on the Plant Power Table. The Power LP Param table contains a value for Operating Head used to index the Operating Head column of the Plant Power Table. This approximated value, therefore, reduces the Power to a function of Turbine Release. The flow values in the Power LP Param table are then used as approximation points indexing the Turbine Release column of the Plant Power Table. The Plant Power Table should have increasing values of Operating Head and Turbine Release. Power should be a concave function of Operating Head, but concavity is not strictly enforced. The preferred order of approximation is substitution, piece-wise, two-point line, tangent.

POWER LP PARAM

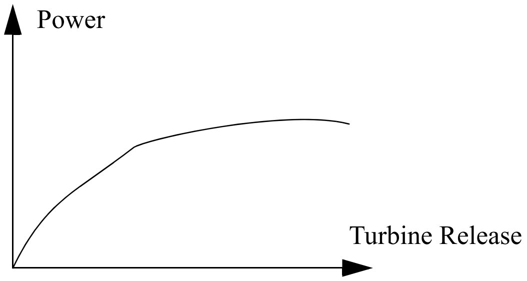
Type: Table Slot

UNITS: LENGTH AND FLOW

Description: Specifies the Operating Head and the flow points used to take the tangent, line and piecewise approximations for Power linearization

Information: This table must be user input unless the Power Linearization Automation category has selected either Plant Automation or Peak Automation methods. The best Operating Head to choose should be close to the expected head during optimized period. Tangent approximation is generally not used, nor helpful as it often results in non-zero power for zero flow. The suggested points for a line approximation are 0 flow and best turbine flow for the entire plant. The suggested points for piecewise linearization are 0 flow, 1 unit best turbine flow, 2 units best turbine flow, . . . n units best turbine flow, and maximum turbine flow.

Defined by:



Power Turbine Release relationship. This is for a fixed operating head value. Not drawn to scale.

5.8.2.20.1 Plant Power Coefficient

SLOTS SPECIFIC TO THIS METHOD

In the RPL Optimization mode, the following slot requires special handling.

☞ **BEST TURBINE FLOW**

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow associated with the most efficient power generation for an associated Operating Head
Information:	
Defined by:	Numerical 2-D Approximation in terms of Operating Head, based upon an internal best turbine flow table. For the Plant Power Coefficient method (selected in the Power category) the internal table is the Best Turbine Q table. The Best Turbine Q table is required to have increasing values of Operating Head and Best Capacity and be a concave function of Operating Head. The preferred order of approximation is substitution, piece-wise, tangent, two-point line. The Best Turbine Flow LP Param table values are used as approximation points indexing the best turbine flow table.

If the Optimization problem uses Best Turbine Flow, either directly or indirectly, then this slot is added to the problem. Before being passed to the optimization solver, this slot is numerically approximated as a function of Operating Head (Numerical 2-D Approximation). The relationship between Operating Head and Best Turbine Flow will come from one of two places, depending on other method selection. 1) If the selected method in the Power category is Plant Efficiency Curve, then the relationship is automatically developed from the Plant Power Table. 2) For the Plant Power Coefficient method, the user-input Best Turbine Q table defines the relationship. 3) For other Plant Calculation category methods, Best Turbine Flow must be input.

☞ BEST TURBINE FLOW LP PARAM

Type:	Table Slot
UNITS:	LENGTH
Description:	Specifies the Operating Head points in the best turbine flow relationship used to take the tangent, line and piecewise approximations for Best Turbine Flow linearization
Information:	The best operating head points to choose for a piecewise approximation are generally: minimum Operating Head, Operating Head at max capacity, and maximum Operating Head. These three points typically define most of the shape of the Best Turbine Flow curve.
Defined by:	User-input

☞ BEST TURBINE Q

Type:	Table Slot
UNITS:	LENGTH VS. FLOW
Description:	Table defining the relationship between Operating Head and the most efficient power generation.
Information:	This table is used if the Power category has the Plant Power Coefficient method selected. If the Power category has the Plant Efficiency Curve method selected, then this table is not used, but a similar relationship is automatically developed from the Plant Power Table.
Defined by:	User-input

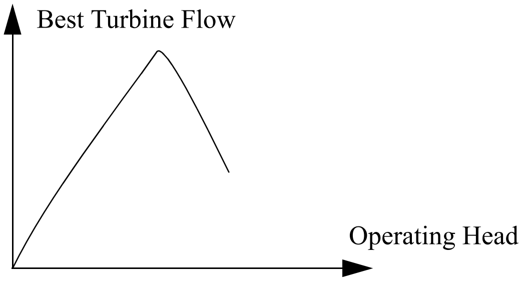
☞ PLANT POWER TABLE

Type:	Table Slot
UNITS:	LENGTH VS. FLOW VS POWER
Description:	Table defining the relationship between Operating Head, Turbine Release, and Power at selected points of operation. In building the Plant Power table, the following approach might be taken. For a given Operating Head create n + 2 rows of data, where n is the number of units comprising the Plant. The first row will reflect no flow through the turbines. The next row will reflect the preferred level of flow through the first unit normally activated. Subsequent rows will reflect incremental flow levels as additional turbines come on line at their respective preferred levels of

flow. The final row will reflect the flow of all available units running at maximum flow capacity.

Information: This table is used to automatically develop the Best Turbine Flow relationship if Plant Efficiency Curve IS SELECTED for the Power category. If it is not, then this table is not used; the Best Turbine Q table is used.

Defined by: User-input



Best Turbine Flow - Operating Head relationship. Not drawn to scale.

5.8.2.20.2 Plant Efficiency Curve

The Plant Efficiency Curve method approximates Best Power Flow the same as the Plant Power Calc

5.8.2.20.3 Peak Power

In RPL Optimization, this method creates a turbine release constraint:

$$\text{Turbine Release} <= \text{Power Plant Capacity Fraction} * \text{Turbine Capacity} * \text{Number of Units.} \quad \text{(EQ 46)}$$

No actual power value is calculated.

5.8.2.20.4 Peak and Base

In RPL Optimization, this method creates a turbine release constraint:

$$\text{Turbine Release} <= \text{Power Plant Capacity Fraction} * \text{Turbine Capacity} * \text{Number of Units.} \quad \text{(EQ 47)}$$

No actual power value is calculated.

5.8.2.20.5 Unit Power Table

The Unit Power slots and Simulation algorithm is described **HERE (Objects.pdf, Section 23.1.1.12)**. The optimization formulation is described **HERE (Section 8.2.6)**.

When the Unit Power Table method is selected, the following categories are available: Cavitation, **HERE (Section 5.8.2.27)** , Avoidance Zones, **HERE (Section 5.8.2.28)**, Startup, **HERE (Section 5.8.2.25)**, and Head Loss, **HERE (Section 5.8.2.26)**, and Frequency Regulation, **HERE (Section 5.8.2.29)**.

When the Unit Power Table method is selected, the following categories are available: Cavitation, Avoidance Zones, Net Head, and Unit Regulation.

The status of a unit at a given timestep can be one of the following:

- Available
- Unavailable
- Must run

At the highest level, an entire plant may never be able to provide an ancillary service like Regulation. The user could indicate this by selecting “None” in the appropriate category. On the other hand, the following slot settings will allow users to specify unit availability / must run for individual time steps for generation and ancillary services. Without any inputs, the units will default to being available.

If the Unit Power Table method, but not the Frequency Regulation, method is selected, the user could specify through optimization RPL policy (note, setting certain slots may be possible but not preferred):

- Unit u must generate at time t:
    - Unit Is Generating [t,u] = 1, or
    - Unit Energy [t,u] = value
- Unit u is unavailable at time t:
    - Unit Is Generating [t,u] = 0, or
    - Unit Energy [t,u] = 0

If the Frequency Regulation method is selected, the user could specify through optimization policy (RPL):

- Unit u must regulate up at time t:
    - Regulation Up [t,u] = value
- Unit u must regulate down at time t:
    - Regulation Down [t,u] = value
- Unit u must regulate at time t:
    - Regulation [t,u] = value (which implies Regulation Up [t,u] = value and Regulation Down [t,u] = value)

If a unit is unavailable for some type of regulation, then a value can be 0.

5.8.2.21 Turbine Capacity

If the selected method for the Optimization Head Computation category is either None or Variable Head (not supported in RPL Optimization), then Turbine Capacity is numerically approximated before being passed to the optimization solver. The slot is approximated as a function of Operating Head (Numerical 2-D Approximation). The relationship between Operating Head and Turbine Capacity will come from one of several places, depending on the method selected in the Power category. 1) If plant-Efficiency Curve is chosen, the relationship is automatically developed from the Plant Power Table. 2) Otherwise, if Peak Power or Peak and Base is chosen, the relationship comes from the Best Generator

Flow table. 3) If none of these methods are chosen, then the relationship is contained in the user-input Max Turbine Q slot.

RELATED SLOTS

☞ TURBINE CAPACITY

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow capacity of the turbine(s)
Information:	
Defined by:	Numerical 2-D Approximation in terms of Operating Head, based upon a maximum turbine capacity table. This capacity table is determined in various ways according to the Power method:

Plant Efficiency Curve: an automatically generated maximum turbine flow table developed from the Plant Power Table

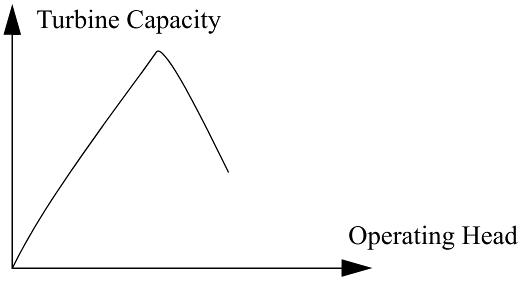
Peak Power or Peak and Base: Best Generator Flow table

All Others: Max Turbine Q

The Turbine Capacity LP Param table values are used as approximation points indexing the selected maximum turbine capacity table. The maximum turbine capacity table is required to have increasing values of Operating Head. Turbine Capacity in that table is required to be a concave function of Operating Head. The preferred order of approximation is substitution, piece-wise, tangent, two-point line.

☞ TURBINE CAPACITY LP PARAM

Type:	Table Slot
UNITS:	LENGTH VS. LENGTH VS. LENGTH
Description:	Specifies the operational head points used to take the tangent, line and piecewise approximations for Turbine Capacity linearization
Information:	For the piecewise approximation the best operating head points to chose are generally: minimum Operating Head, Operating Head at max capacity, and maximum Operating Head. These three points typically define most of the shape of the Turbine Capacity curve.
Defined by:	User-input unless the selected method in the Power Linearization Automation category is not “None”



Turbine Capacity Operating Head relationship. Not drawn to scale.

5.8.2.22 Optimization Tailwater

Click **HERE (Objects.pdf, Section 23.1.7)** for more information on the Simulation methods for Tailwater. Only a subset of available methods are supported in Optimization. These methods are: Linked or Input, Base Value Only, Base Value Plus Lookup Table, Stage Flow Lookup Table, and Coefficients Table. Selection of a method other than those will result in an error. The method selected in the Optimization Tailwater category should correspond to the one selected in the simulation Tailwater category.

Tailwater Elevation may be part of the Optimization problem and it is handled differently for each method. Below is a description of the behavior of Tailwater Elevation.

5.8.2.22.1 Opt Linked or Input

If Tailwater Elevation is not input, then Tailwater Elevation is replaced by TailwaterBaseValue: Tailwater Elevation = Tailwater Base Value.

☞ **TAILWATER BASE VALUE**

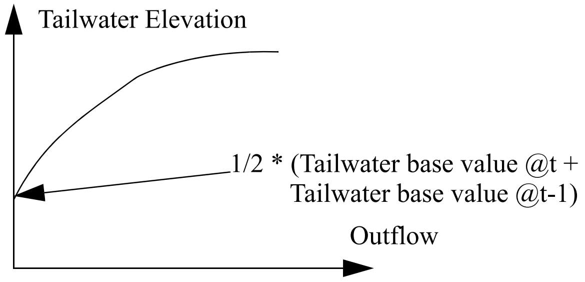
Type:	Series
UNITS:	LENGTH
Description:	Described in the simulation documentation **HERE (Objects.pdf, Section 23.1.7.2)**

5.8.2.22.2 Opt Base Value Only

If Tailwater Elevation is not input, then Tailwater Elevation is replaced by (Tailwater Base Value(t) + Tailwater Base Value(t-1)) / 2

☞ **TAILWATER BASE VALUE**

Type:	Series
UNITS:	LENGTH
Description:	Described in the simulation documentation **HERE (Objects.pdf, Section 23.1.7.3)**



Tailwater Outflow relationship. Not drawn to scale.

5.8.2.22.3 Opt Base Value Plus Lookup Table

If Tailwater Elevation is not input, this slot is replaced by a mathematical expression based on Tailwater Base Value and a numerical approximation of Tailwater Elevation as a function of Outflow (Numerical 2-D Approximation) as defined by the user in the Tailwater Table

Tailwater Elevation is constrained to be (tailwaterBaseValue(t) + tailwaterBaseValue(t-1))/2 + tempTWLookupValue, where tempTWLookupValue is a Numerical 2-D Approximation of increase in Tailwater Elevation due to flow as described in the Tailwater Table. The Tailwater Table Lookup LP Param table values for outflow are used as approximation points indexing the Tailwater Table to determine tempTWLookupValue from the Tailwater Elevation column.

☞ **TAILWATER BASE VALUE**

Type:	Series
UNITS:	LENGTH
Description:	Described in the simulation documentation **HERE (Objects.pdf, Section 23.1.7.4)**

☞ **TAILWATER TABLE**

Type:	Series Slot
UNITS:	FLOW VS. LENGTH
Description:	Reservoir outflow vs. either the tailwater elevation or the tailwater elevation increment
Information:	If the Tailwater Base Value is non-zero, the Tailwater Table gives values of incremental increase in Tailwater Elevation over the Base value. If the Tailwater Base Value is zero, the table simply gives the Tailwater Elevation values. The Tailwater Table should have increasing values of outflow. Tailwater Elevation is required to be a convex function of outflow. The preferred order of approximation is

	substitution, piece-wise, two-point line, tangent. Please see Object / Simulation documentation for further information.
I/O:	User-input

☞ Tailwater Table Lookup LP Param

Type:	Table Slot
Units:	Flow, Flow, Flow
Description:	LP Parameter approximation points for Outflow.
Information:	The Tailwater Table Lookup LP Param table values for outflow are used as approximation points indexing the Tailwater Table to determine tempTWLookupValue from the Tailwater Elevation column.
I/O:	User Input only
Links:	Not Linkable

☞ temp Tailwater Lookup Value

Type:	Series
Units:	Length
Description:	Numerical 2-D Approximation of increase in Tailwater Elevation due to flow as described in the Tailwater Table.

5.8.2.22.4 Opt Stage Flow Lookup Table

For more information on the Stage Flow Lookup Table method, click **HERE (Objects.pdf, Section 23.1.7.5)**. Tailwater Elevation is numerically approximated as a function of Stage and Flow as defined by the user in the Stage Flow Tailwater Table. The tables will be queried either using user-input points defined in the Tailwater Elevation LP Param table or using automatically calculated points, depending on method selection in the Tailwater Linearization Automation category. If the selected method in the Tailwater Linearization Automation category is “None”, then the user-input Tailwater LP Param table values are used. For the Range Input Automation method, the automatically developed points will be used.

Tailwater Elevation is approximated numerically (3-D approximation) as a function of Outflow and Tailwater Base Value based on the Stage Flow Tailwater Table. The Tailwater Elevation LP Param table Tailwater Base Value and flow values are used as approximation points indexing the Downstream Stage and Outflow columns of the Stage Flow Tailwater Table, respectively.

☞ TAILWATER BASE VALUE

Type:	Series
UNITS:	LENGTH
Description:	Described in the simulation documentation **HERE (Objects.pdf, Section 23.1.7.5)**

☞ STAGE FLOW TAILWATER TABLE

Type:	Table Slot
UNITS:	FLOW VS. LENGTH VS. LENGTH
Description:	Reservoir Outflow vs. Downstream Elevation (Tailwater Base Value) vs. Tailwater Elevation
Information:	Data must be entered into the table in increasing blocks of the same Outflow value for the 3-dimensional table interpolation to work correctly. For every block of same Outflows in column 1, Downstream Stages should be listed in increasing order in column 2, and the corresponding Tailwater Elevations in column 3. Tailwater elevation is required to be a concave function of Outflow. The preferred order of approximation is substitution, piece-wise, two-point line, tangent. Internally, the Stage Flow Tailwater Table is rearranged to use the Stage as the primary index (column) with outflow as the secondary index (column). This rearranged table may be labeled as “Convolved Stage Flow Tailwater Table” in output messaging. Also, because of this rearrangement, it is desirable to repeat stage values for each outflow.
Defined by:	User-input

☞ TAILWATER ELEVATION LP PARAM

Type:	Table Slot
UNITS:	LENGTH, FLOW, FLOW, FLOW
Description:	Specifies the fixed tailwater base value point and the outflow points used to take the tangent, line and piecewise approximations for tailwater elevation linearization
Information:	This table is used for linearization. The best Tailwater Base Value point to choose for tangent approximation would be an outflow equal to the expected value of outflow during the run; for the line approximation, the minimum and maximum values expected during the run; for piecewise approximation, the minimum and maximum values expected during the run plus additional intermediate values to more closely fit the tailwater curve.
Defined by:	User-input unless the selected method in the Tailwater Linearization Automation category is not “None”.

☞ TAIL WATER REFERENCE ELEVATION

Type: Table Slot
UNITS: LENGTH
Description: Lowest Reservoir discharge Elevation when there are no backwater effects from a downstream pool (reservoir)
Information: Although this is part of the Stage Flow Lookup Table method (and its corresponding Opt method), **this slot does not influence optimization**. Please see Object / Simulation documentation for further information.
I/O: User-input

5.8.2.22.5 Opt Coefficients Table

This method sets up the physical Tailwater Elevation constraints using the same equation and coefficients used in the Coefficients Table tailwater method, **HERE (Objects.pdf, Section 23.1.7.7)**:

$$\begin{aligned}\text{Tailwater Elevation} = {} & \text{Constant Coeff}[t] + \text{Constant Coeff}[t-1] + \\ & \text{Outflow Coeff}[t] \times \text{flow} + \text{Outflow Coeff}[t-1] \times \text{Outflow}[t-1] + \\ & \text{TW Base Val Coeff}[t] \times \text{TailwaterBaseValueTemp}[t] + \\ & \text{TW Base Val Coeff}[t-1] \times \text{Tailwater BaseValue}[t-1] + \\ & \text{TW Elev Coeff}[t-1] \times \text{Tailwater Elevation}[t-1]\end{aligned}$$

☞ TAILWATER BASE VALUE

Type: Series
UNITS: LENGTH
Description: See Simulation description **HERE (Objects.pdf, Section 23.1.7.7)**.

☞ TAILWATER COEFFICIENTS

Type: Table
UNITS: MULTI
Description: See Simulation description **HERE (Objects.pdf, Section 23.1.7.7)**.
Defined by: If the Power Surface Approximation method is selected, then there are further restrictions on the coefficients that can be specified. With the Power Surface Approximation, having terms for Outflow[t-1], Tailwater BaseVal[t] and Tailwater[t-1] are not allowed.

5.8.2.23 Tailwater Linearization Automation

This method category allows the user to choose whether they want to enter the approximation points used to the linearization approximations of tailwater or have the selection done automatically by riverware. The default method is none, which requires the points be input by the user. The automation method Range Input determines the points using the expected range of outflow values.

This category is dependent on 1) Optimization Head Calculation category selection of either Variable Head (not available in RPL Optimization) or None, and 2) Optimization Power category selection of Independent Linearizations, and 3) Optimization Tailwater category selection of either Opt Base Value Plus Lookup Table or Opt Stage Flow Lookup Table.

5.8.2.23.1 None

This method requires the input of the approximation points for tailwater linearization into the Tailwater LP param table.

5.8.2.23.2 Range Input Automation

The Range Input Automation method selects the linearization points for the linear approximations of tailwater in the Tailwater Elevation LP Param table. Tailwater is linearized as a 2- or 3-dimensional function depending on the method selected in the Optimization Tailwater category. If Opt Base Value Plus Lookup Table is selected tailwater is a 2-D function of flow. If Opt Stage Flow Lookup Table is selected tailwater is a 3-D function of flow and downstream stage. The same flow points are used for both the 2- and 3-D approximations. For the 3-D approximation the fixed point is required for the downstream stage. This point must still be input by the user.

SLOTS SPECIFIC TO THIS METHOD

☞ **EXPECTED OUTFLOW RANGE**

Type:	Table Slot
UNITS:	FLOW, FLOW
Information:	
Information:	Provide the high and low expected outflow values for the run.
Defined by:	User Input

The user inputs high and low expected values into the Expected Outflow Range table. The average of the values is calculated. The average is used as the tangent approximation point. The low and high values are used as the line approximation points. The low, average and high points are used as the piecewise points. The fixed point for three dimensional approximations is still required to be user input, but the validity of the point is checked during the automation process.

5.8.2.24 Optimization Reserves

This category depends on selecting either Plant Power Coefficient or Plant Efficiency Curve in the Power category. The methods in this category can be used to account for power reserve requirements in the optimization policy. There is no analogous simulation method associated with this category.

Note: The methods in this category have not been fully implemented for Slope Power Reservoirs. The selected method should be left as the default, None. Otherwise the run will abort with an error message stating that the method selection is not allowed for Slope Power Reservoirs.

5.8.2.24.1 None

This is the default, do-nothing method. No new slots will be added, and no new variables or constraints will be added to the optimization problem. This method should always be selected for Slope Power

Reservoirs.

5.8.2.24.2 Constraint Based Single Timestep

Although this method is visible on Slope Power Reservoirs, it is not fully implemented. Selecting this method on Slope Power Reservoir will cause the run to abort with an error.

5.8.2.25 Startup

This category depends on selecting the Unit Power Table method, **HERE (Section 5.8.2.20.5)**, and describes how the monetary cost associated with starting up or shutting down a unit (turbine) will be modeled. There are two methods in this category, one which does not model these costs (effectively assigning them a value of 0) and one which uses a table describing the combined costs for starting up and shutting down a unit.

5.8.2.25.1 None

This is the default, do-nothing method.

5.8.2.25.2 Unit Lumped Cost Method

For each unit, this method lumps the cost of startup and shutdown into one value. Slots are described **HERE (Objects.pdf, Section 23.1.34.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.8.2.26 Head Loss

This category depends on the Unit Power Table method, **HERE (Section 5.8.2.20.5)**, and contains methods for modeling additional head loss that occurs. This head loss may come from the configuration of the penstocks for bringing water to the turbines.

5.8.2.26.1 None

In this method, there is no additional head loss to be used in the power calculation. In terms of penstock head loss, this method should be selected if the penstocks for the units are independent and the penstock losses are typically incorporated in the power data. Thus the power data is specified in terms of operating head.

5.8.2.26.2 Shared Penstock Head Loss method

In this method, there is additional head loss that results because units share a common penstock. The operating head losses in the penstock depend on the total turbine release and are shared for all units. The net head is calculated by subtracting penstock losses from the operating head. The unit data and power must be specified in terms of unit Net Heads instead of Operating Head. Slots are described **HERE (Objects.pdf, Section 23.1.35.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.8.2.27 Cavitation

This category depends on selecting the Unit Power Table method, **HERE (Section 5.8.2.20.5)**, and contains methods for dealing with the problem of cavitation on turbines. Cavitation is the sudden formation and collapse of low-pressure bubbles in liquids by means of mechanical forces and this process can cause damage to turbines under certain operating conditions.

5.8.2.27.1 None

This is the default, do-nothing method.

5.8.2.27.2 Unit Head and Tailwater Based Regions

This method allows the user to specify the regions of operation in which cavitation does NOT occurs, so that these regions can be avoided. These regions can be dependent on both operating head and tailwater. Slots are described **HERE (Objects.pdf, Section 23.1.36.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.8.2.28 Avoidance Zones

This category depends on selecting the Unit Power Table method, **HERE (Section 5.8.2.20.5)**, and contains methods for modeling the existence of undesirable regions of operation for turbines. There are two methods in this category, one which does not model avoidance zones at all, and one which

5.8.2.28.1 None

This the default, do-nothing method; avoidance zones are not considered.

5.8.2.28.2 Unit Head Based Avoidance Zones

This method allows the user to specify a table that defines the conditions in which the turbines should not be operated. Slots are described **HERE (Objects.pdf, Section 23.1.37.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.8.2.29 Frequency Regulation

This category depends on selecting the Unit Power Table method, **HERE (Section 5.8.2.20.5)**, although in the future it might be enabled for other power methods. The frequency regulation methods model the provision of the frequency regulation ancillary service, that is, how the reservoir can be made available to flexibly follow a load demand within a specified range during a certain period in order to affect the frequency of the generated power.

5.8.2.29.1 None

This is the default, do-nothing method; no regulation is modeled.

5.8.2.29.2 Unit Frequency Regulation

NOTE: THIS METHOD IS NOT YET IMPLEMENTED

When frequency regulation is scheduled, it allows the unit to follow the real time load. Exactly what will happen in real time is unknowable. This results in two sets of values at scheduling time, nominally scheduled power and turbine release. It is uncertain if the real time operators will actually use the service. At present, we distinguish between the nominal “scheduled” power (and turbine release) that the regulation is allowed to depart from and the “expected” power generation (and turbine release) that will take place when regulation is allowed. Both are important. The scheduled power sets the baseline for regulation and should be communicated to the power dispatchers. The expected power and release are more useful for coordinating a plant with the rest of the system. Slots are described **HERE (Objects.pdf, Section 23.1.38.2)** and the constraints added are described **HERE (Section 8.2.7.3.5)**.

5.8.3 Modeling a Slope Power Reservoir

Steps to follow in setting up a level power reservoir for optimization

1. Set up a running simulation model, using methods that are compatible with optimization for Power, Tailwater, Spill, etc.

2. Choose a method in the Slope Storage Coefficients category.

3. Fill in the Backwater Lambda Coefficients, LP Parameters, and Restrictions table.

4. Chose an Optimization Power method.

5. Choose an Optimization Head Computation method. In Optimization, None must be chosen.

6. Select the Optimization Spill method corresponding to the method selected in the Spill category.

7. Fill in the LP Param tables related to Power, Turbine Capacity and Best Turbine Flow (if plant power is used). Alternatively, the Power Linearization Automation Method can also be chosen to automatically determine the values for the LP Param tables.

8. Select the Optimization Tailwater method corresponding to the method selected in Tailwater category. If Opt Base Value Plus Lookup Table or Opt Stage Flow Lookup Table is selected, fill in the Tailwater Elevation LP Param table. Alternatively, select a Tailwater Linearization Automation method (Range Input Automation is currently available).

9. If future value is needed, select a method in the Future Value category, followed by the Optimization Future Value category, and if desired, Cumul Stor Val Linearization Automation.

10. Selection of the remaining methods and linearizations can be done in any order. Pool elevation linearizations, Pool Elevation Linearization Automation, Evaporation and Precipitation, Optimize Evaporation Computation, Evaporation Linearization Automations, Bank Storage, Hydrologic Inflow, Energy In Storage, and Diversion from Reservoir. The appropriate data must be entered for each method.

5.9 Storage Reservoir

The Storage Reservoir is the least complicated of all the reservoirs. The only process performed by the Storage Reservoir is the storage of water. No power production facilities exist on the Storage Reservoir. Additional information can be found **HERE (Objects.pdf, Section 24)**.

5.9.1 General Slots

General slots are always present on the object, regardless of selected methods. The following slots are provided on the reservoir when the optimization controller is selected.

☞ Canal Flow

Type: Agg Series
Units: Flow
Description: Flow into (out of) the reservoir from (to) a canal
Information:
Defined by: Explicit Optimization variable in the mass balance constraint

☞ Diversion

Type: Series Slot
Units: Flow
Description: Flow from the reservoir to a diverting object
Information:
Defined by: Explicit Optimization variable in the mass balance constraint

☞ Elevation Volume Table

Type: Table
Units: Length vs Volume
Description: Table relating elevation of the reservoir to volume stored in the reservoir
Information:
I/O: Input only
Defined by: Input only

☞ Flow FROM Pumped Storage

Type: Agg Series Slot
Units: Flow
Description: Flow into the reservoir from a pumped storage reservoir
Information:
Defined by: Explicit Optimization variable in the mass balance constraint. This slot should be linked to Outflow on a Pumped Storage object. The Pumped Storage object constrains its Outflow.

☞ FLOW TO PUMPED STORAGE

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Flow out of the reservoir into a pumped storage reservoir
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint. This slot should be linked to Pumped Flow on a Pumped Storage object.

☞ INFLOW

Type:	MultiSlot
UNITS:	FLOW
Description:	Inflow into the reservoir from upstream
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint

☞ MAX RELEASE

Type:	Table Slot
UNITS:	LENGTH VS FLOW
Description:	Table relating Pool Elevation to maximum release
Information:	See power methods for more information

☞ OUTFLOW

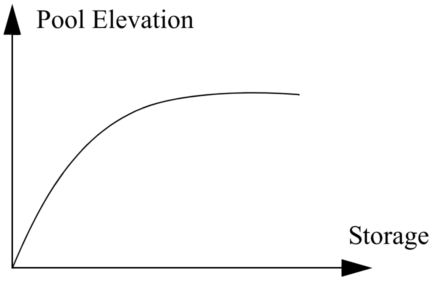
Type:	Agg Series Slot
UNITS:	FLOW
Description:	Outflow from the reservoir
Information:	
Defined by:	Explicit Optimization variable as Outflow = Turbine Release + Spill

☞ POOL ELEVATION

Type:	Agg Series Slot
UNITS:	LENGTH
Description:	Elevation of the water surface of the Reservoir
Information:	When Pool Elevation is a part of the optimization problem, as it is in all conceivable RiverWare Optimization applications, this slot is numerically approximated as a function of Storage (Numerical 2-D Approximation). The relationship between Pool Elevation and Storage will come from the user-input Elevation Volume Table. The table will be queried either using user-input points defined in the Pool Elevation LP Param table.
Defined by:	Numerical 2-D Approximation in terms of Storage, based upon the Elevation Volume Table. The Pool Elevation LP Param table values are used as approximation points indexing the Elevation Volume Table. The Elevation Volume Table should have increasing values of Pool Elevation and Storage. Storage is required to be a concave function of Pool Elevation. The preferred order of approximation is substitution, piece-wise, tangent, two-point line.

☞ POOL ELEVATION LP PARAM

Type:	Table Slot
UNITS:	VOLUME
Description:	Specifies the Storage points used to take the tangent, line and piecewise approximations for Pool Elevation linearization
Information:	This table is used for linearization unless Pool Elevation Linearization Automation category has selected Plant Automation. The best Storage point to choose for tangent approximation would be the expected storage expected during the run; for the line approximation, the expected maximum and minimum Storage; for piecewise approximation, use points that cover the full range of expected Storage during the run with intermediate points such that a piecewise linear curve reasonably approximates the actual curve.
Defined by:	User-input



Pool elevation storage relationship. The figure is not drawn to scale.

☞ RETURN FLOW

Type:	MultiSlot
UNITS:	FLOW
Description:	Flow returning from a diversion object
Information:	
Defined by:	Explicit Optimization variable in the mass balance constraint (see Storage)

☞ SPILL

Type:	Agg Series Slot
UNITS:	FLOW
Description:	Sum of the Bypass, Regulated Spill and Unregulated Spill
Information:	
Defined by:	Explicit Optimization variable as Spill = Bypass + Regulated Spill + Unregulated Spill

☞ **STORAGE**

Type: Agg Series Slot
UNITS: VOLUME
Description: Volume of water stored in the reservoir
Information:
Defined by: Explicit Optimization variable as Storage = Storage(t-1) + Precipitation Volume - Evaporation - Change in Bank Storage + timestep * (Inflow + Canal Flow + Flow TO Pumped Storage + Hydrologic Inflow Net + Return Flow - (Outflow + Diversion + Flow FROM Pumped Storage))

5.9.2 User Methods in Optimization

The following categories and methods are available for use in Optimization. Because of dependency relationships, you may not be able to see them in your model until you change other methods. In the discussion below, you will see the other methods that need to be selected to enable a given method to be used in your model. When building a model, you will wish to review this to ensure that required dependencies are satisfied so that the desired methods are available for use.

5.9.2.1 Bank Storage

Not all methods are functional in RPL optimization. Of the available methods, “None” and “CRSS Bank Storage” are supported.

5.9.2.1.1 None

Click **HERE (Objects.pdf, Section 24.1.19.1)** for more information.

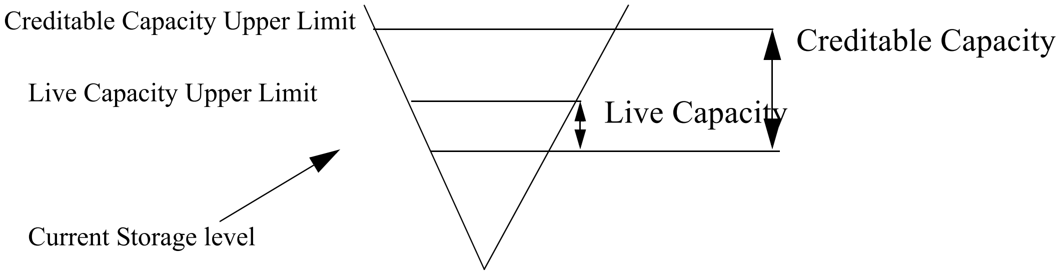
Input Bank Storage

5.9.2.1.2 CRSS Bank Storage

Click **HERE (Objects.pdf, Section 24.1.19.3)** for more information.

5.9.2.2 Creditable Capacity Available

The creditable capacity category conceptually represents the total amount of available storage space above the current storage in the reservoir. As the pool storage in the reservoir increases, the creditable capacity decreases.



5.9.2.2.1 None

If this method is selected creditable capacity is not calculated for the reservoir.

5.9.2.2.2 Constraint and Variable

If this method is selected an additional decision variable and physical constraint are added to the optimization problem:

SLOTS SPECIFIC TO THIS METHOD:

☞ **CREDITABLE CAPACITY**

Type:	Gassers
UNITS:	VOLUME
Description:	The creditable capacity is the total amount of storage space available above the current storage
Information:	
Defined by:	Explicit Optimization Variable included in the Optimization Problem as part of the constraint

$$\text{Max Storage} >= \text{Storage} + \text{Creditable Capacity.} \quad \text{(EQ 48)}$$

Max Storage for the reservoir is entered in the “max value” field of the Storage slot configuration dialog box.

5.9.2.3 Cumul Stor Val Linearization Automation

Appearance of this category is dependent on selecting the Opt Cumulative Storage Value Table method for the Optimization Future Value category (which in turn is dependent on selecting the Cumulative Storage Value Table method for the Future Value category).

This category allows the optimization to automate the creation of the Cumulative Storage Value Table, and the selection of linearization points and linearization method for the Cumulative Storage Value slot. Actual usage of these values occurs in the **HERE (Section 5.9.2.10)** Optimization Future Value category’s

Opt Cumulative Storage Value Table method.

5.9.2.3.1 None

If this method is selected, no automation will be performed.

5.9.2.3.2 Marginal Value to Table and Lin

This method uses information from the simulation slot Marginal Storage Value Table to generate the Cumul Stor Val Table, select linearization points, and choose a linearization method. The cumulative storage value in the Cumul Stor Val Table can be thought of as the summation of the marginal storage values from a storage of 0 to the current storage.

As an illustration of the automation procedure, consider the following Marginal Storage Value Table:

Marginal Storage Value Table:

Storage	Marginal Value
20	30
60	26
100	24

To parameterize the Cumul Stor Val Table, the automation proceeds as follows:

1) The first row receives a Storage value of 0.

Cumul Stor Val Table

Storage	Cumulative Value
0	

2) For each row i in the Marginal Storage Table, not including the last row, the average Storage (i.e. the midpoint) of row i and row i + 1 is assigned as Storage in each successive row of the Cumul Stor Val Table. For example, row 2 Storage equals the average of 20 and 60; row 3 Storage equals the average of 60 and 100.

Cumul Stor Val Table

Storage	Cumulative Value
0	
40	

Storage	Cumulative Value
80	

3) The last row of the Cumul Stor Val Table receive a Storage value equal to the maximum storage associated with the Storage slot’s configuration.

Cumul Stor Val Table

Storage	Cumulative Value
0	
40	
80	
140	

4) Returning to the first row of the Cumul Storage Val Table, a Cumulative Value of 0 is assigned.

Cumul Stor Val Table

Storage	Cumulative Value
0	0
40	
80	
140	

5) For each successive row j = 2, 3, 4 in the Cumul Stor Val Table and corresponding row i = 1, 2, 3 in the Marginal Storage Value Table, the Cumulative Value equals the Storage from row j - 1 + (the change in Storage of row j from row j - 1) * the Marginal Value from row i. For example, row 2 Cumulative Value equals 1200 calculated from 0 + (40 - 0) * 30; row 3 Cumulative Value equals 2240 calculated from 1200 + (80 - 40) * 26; row 4 Cumulative Value equals 3680 calculated from 2240 + (140 - 80) * 24.

Cumul Stor Val Table:

Storage	Cumulative Value
0	0
40	1200
80	2240
140	3680

The Cumul Stor Val LP Param table is then built using Storage values from the Cumul Stor Val Table:

Cumul Stor Val LP Param

Tangent	Line	piecewise
	0	0
	140	40
		80
		140

Slots Specific to this Method

There are no slots specific to this method as it uses the slots in the Opt Cumulative Storage Value Table method. However, for clarity in the discussion above, the slots are reshown here.

☞ Cumul Stor Val Table

Type: Table Slot
Units: Volume vs. Value
Description: The estimated total economic value of water stored in the reservoir for discrete storage values.
Information:
Defined by: Automated procedure described above.

☞ Cumul Stor Val LP Param

Type: Table Slot
Units: Volume, Volume, Volume
Description: Specifies the storage points used to take the tangent, line and piecewise approximations for Cumul Stor Val Table linearization
Information:
Defined by: Automated procedure described above.

☞ Marginal Storage Value Table

Type: Table Slot
Units: Storage vs. $ Value
Description: Anticipated Storage versus worth of Cumulative Storage per unit energy
Information: This table should be increasing in storage, and usually decreasing in marginal value
Defined by: Required input

5.9.2.4 Diversion from Reservoir

Not all methods in this category are supported in RPL optimization. Of the available methods, “None” and “Available Flow Based Diversion” are supported; selection of any other method will result in an error during begin run.

5.9.2.4.1 None

Click **HERE (Objects.pdf, Section 22.1.26.1)** for more information.

5.9.2.4.2 Available Flow Based Diversion

Click **HERE (Objects.pdf, Section 22.1.26.2)** for more information.

5.9.2.5 Energy in Storage

In Optimization, currently “None” and “EIS Table Lookup” are supported.

5.9.2.5.1 None

No Energy in storage is considered.

5.9.2.5.2 EIS Table Lookup

With this method selected, Energy in Storage is considered as a function of Pool Elevation. Click **HERE (Objects.pdf, Section 24.1.1.2)** for more information on the simulation method. If the optimization problem uses Energy In Storage, either directly or indirectly, then this slot is added to the problem. Before being passed to the optimization solver, this slot is numerically approximated as a function of Pool Elevation (Numerical 2-D Approximation). The relationship between Pool Elevation and Energy In Storage will come from the user-input Energy In Storage Table. The table will be queried either using user-input points defined in the Energy In Storage LP Param table or using automatically calculated points, depending on method selection in the Pool Elevation Linearization Automation category. If the selected method in the Pool Elevation Linearization Automation is “none”, then the user-input Energy In Storage LP Param table values are used. For other Pool Elevation Linearization Automation methods (Initial Target Input, Min Difference or Range Input, Min Difference) the same points automatically developed for the Pool Elevation linearization will be used.

RELATED SLOTS

☞ **ENERGY IN STORAGE**

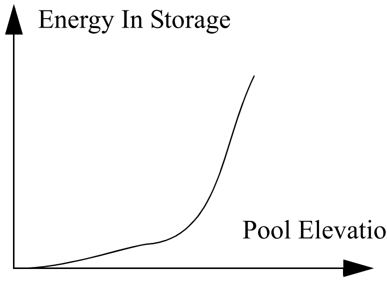
Type:	Series Slot
UNITS:	ENERGY VS. POWER
Description:	Energy in Storage in the reservoir
Information:	
Defined by:	Numerical 2-D Approximation in terms of Pool Elevation, based upon the Energy In Storage Table. The Energy In Storage LP Param table values are used as approximation points indexing the Energy In Storage Table. The Energy In Storage Table should have increasing values of Pool Elevation and Energy In Storage. Energy In Storage is required to be a convex function of Pool Elevation. The preferred order of approximation is substitution, piece-wise, tangent, two-point line.

☞ **ENERGY IN STORAGE LP PARAM**

Type:	Table Slot
UNITS:	LENGTH
Description:	Specifies the Pool Elevation points used to take the tangent, line and piecewise approximations for Energy In Storage linearization.
Information:	This table is used for linearization unless the Pool Elevation Linearization Automation category has a method selected other than “none”.
Defined by:	User-input

☞ **ENERGY IN STORAGE TABLE**

Type:	Table Slot
UNITS:	LENGTH VS. ENERGY
Description:	Table defining the relationship between Energy In Storage and Pool Elevation
Information:	
Defined by:	User-input



Energy in Storage pool elevation relationship. Not drawn to scale.

5.9.2.6 Evaporation and Precipitation

The Input Evaporation method is the only Evaporation and Precipitation method that can be modeled in conjunction with Optimization. If the Input Evaporation method is selected, the Opt Input Evaporation method must be selected for the Optimization Evaporation category. Otherwise “None” should be selected for the Evaporation and Precipitation category, and “None” should be selected for the Optimization Evaporation category. When Evaporation and Precipitation Volume exist on the reservoir, they are added as part of the mass balance constraint.

5.9.2.7 Optimization Evaporation

This category can be used to model evaporation and precipitation.

5.9.2.7.1 None

The Optimization Evaporation method “None” is the default method for this category. It does no calculations and requires that “None” be selected for the Evaporation and Precipitation category.

5.9.2.7.2 Opt Input Evaporation

This method is analogous to the Input Evaporation method in simulation and requires the Input Evaporation method to be selected for the Evaporation and Precipitation category. Evaporation Rate and Precipitation Rate are entered as a time series. Evaporation is calculated as a product of Evaporation Rate, Average Surface Area over the timestep and Timestep length. Similarly Precipitation Volume is calculated as the product of Precipitation Rate, Average Surface Area and Timestep length.

Note: The linearization of the Surface Area variable can result in a small approximation error in optimization for Evaporation and Precipitation Volume. This means there can be a small difference between the mass balances in the optimization solution and the post-optimization rulebased simulation when using this method. It is important to use care when setting the approximation points for Surface Area in order to reduce this approximation error. Refer to the information on the Surface Area LP Param slot below. Also caution should be used if applying this method at a monthly timestep. All rates in the optimization mass balance are converted to monthly volumes based on a 30-day month, regardless of the month. This will also produce a difference between the mass balances in the optimization solution and the post-optimization rulebased simulation for a *monthly* timestep.

SLOTS SPECIFIC TO THIS METHOD

☞ ELEVATION AREA TABLE

Type: Table Slot
UNITS: LENGTH VS. AREA
Description: Represents the Elevation-Surface Area relationship
Information: This table must be input. It is used to derive the Volume Area Table.
Defined by: User-input

☞ EVAPORATION

Type: Series Slot
UNITS: VOLUME
Description: The volume of water lost to evaporation over the timestep
Information: If this slot contains user input, it is added directly to the mass balance constraint, otherwise it is defined by the expression below.
Defined by: Either user-input or the following constraint:

$$\text{Evaporation} = \text{EvaporationRate} \times \frac{\text{SurfaceArea}(t-1) + \text{SurfaceArea}}{2} \times \text{Timestep}$$

☞ EVAPORATION RATE

Type: Series Slot
UNITS: VELOCITY
Description: The rate, in length per time, at which water is lost to evaporation at each timestep
Information: This slot can be set as user input. If it is not set as an input, and if Evaporation is not an input, this slot defaults to zero. If Evaporation is an input, this slot is not used.
Defined by: User-input or defaults to zero

☞ PRECIPITATION RATE

Type: Series Slot
UNITS: VELOCITY
Description: The rate, in length per time, at which water is gained from precipitation at each timestep
Information: This slot can be set as user input. If it is not set as an input, it defaults to zero.
Defined by: User-input or defaults to zero

☞ PRECIPITATION VOLUME

Type: Series Slot
UNITS: VOLUME
Description: The volume of water gained from precipitation over the timestep
Information: The Input Evaporation method will not allow this slot to be set as an input.
Defined by: Explicit constraint:

$$\text{Precipitation Volume} = \text{PrecipitationRate} \times \frac{\text{SurfaceArea}(t-1) + \text{SurfaceArea}}{2} \times \text{Timestep}$$

☞ SURFACE AREA

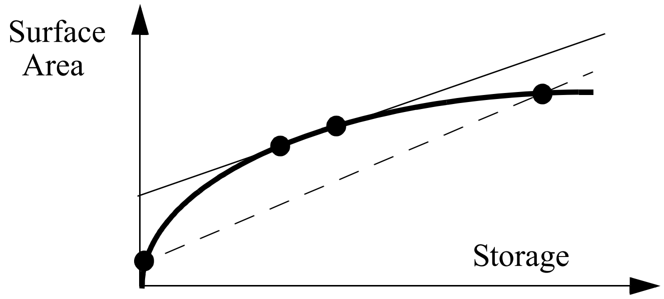
Type: Series Slot
UNITS: AREA
Description: The area of the water surface at the end of the timestep
Information: This slot is numerically approximated as a function of Storage (Numerical 2-D Approximation). The relationship between Pool Elevation and Storage comes from the automatically generated Volume Area Table. The table is queried using the user-input points defined in the Surface Area LP Param table.
Defined by: Numerical 2-D Approximation in terms of Storage, based upon the Volume Area Table. The Surface Area LP Param table values are used as approximation points indexing the Volume Area Table. The preferred order of approximation is substitution, piecewise, two-point line, tangent. Most often the two-point line (secant) approximation will be used.

☞ SURFACE AREA LP PARAM

Type: Table Slot
UNITS: VOLUME
Description: Specifies the Storage points used to take the tangent, line and piecewise approximations for Surface Area linearization
Information: The best Storage point to choose for tangent approximation would be the expected storage during the run; for line approximation, the expected maximum and minimum Storage; for piecewise approximation, use points that cover the full range of expected Storage during the run with intermediate points such that a piecewise linear curve reasonably approximates the actual curve.In most cases, the line (secant)

approximation will be used. It is important to set these points carefully to minimize approximation error.

Defined by: User-input



Surface Area vs. Storage relationship with two alternative line approximations.
The figure is not drawn to scale

The figure above represents two alternative selections of points for the line approximation in the Surface Area LP Param table. The linear approximation represented by the dashed line corresponds to the selection of points near the extremes of the Volume Area table. This approximation will tend to result in an under-estimation of Surface Area, and thus an under-estimation of Evaporation and Precipitation Volume. Evaporation losses in the post-optimization rulebased simulation would be greater than the losses approximated in the optimization solution. The linear approximation represented by the solid line corresponds to the selection of points closer together in the Volume Area Table, and is more similar to the Tangent approximation. This approximation would tend to result in an over estimation of Surface Area, and the losses due to Evaporation in the post-optimization rulebased simulation would be less than the approximation in the optimization solution.

☞ **VOLUME AREA TABLE**

Type:	Table Slot
UNITS:	VOLUME VS. AREA
Description:	Represents the Storage Volume-Surface Area relationship
Information:	This table is read-only and is automatically generated at the start of the run from the Elevation Area Table and the Elevation Volume table. The method starts by copying the Elevation Area Table and then replaces the Pool Elevation Values with the corresponding Storage values. The Storage values are linearly interpolated based on the Elevation Volume Table
Defined by:	Automatically generated

5.9.2.8 Evaporation Linearization Automation Category

This category allows the approximation points for surface area to be automatically generated. This category requires a method other than “None” to be selected for the Optimization Evaporation category.

5.9.2.8.1 None

This is the default method for this category. There is no approximation point automation. The user will be required to enter approximation points in the Surface Area LP Param slot manually.

5.9.2.8.2 Use Elevation Approximation Points

This method automates the approximation points in the Surface Area LP Param table slot. It copies the same storage values used in the Pool Elevation LP Param slot. If this method is selected it will overwrite any values that have been entered manually in the Surface Area LP Param slot. This method may provide a good starting point for establishing the Surface Area approximation points; however in some cases, it may be important for the user to adjust these points manually (see details **HERE (Section 5.9.2.7.2)** under Surface Area LP Param).

5.9.2.9 Future Value

This category and its methods are not dependent on other method selections.

5.9.2.9.1 None

Click **HERE (Objects.pdf, Section 24.1.7.1)** for more information.

5.9.2.9.2 Cumulative Storage Value Table

In RPL-Optimization, the Cumulative Storage Value is Numerically Approximated as described below.

Click **HERE (Objects.pdf, Section 24.1.7.2)** for more information.

SLOTS SPECIFIC TO THIS METHOD

☞ **CUMULATIVE STORAGE VALUE**

Type:	Agg Series Slot
UNITS:	$
Description:	Represents the future energy value of the current storage
Information:	
Defined by:	2-D approximation in terms of Anticipated Storage, based upon the Cumul Stor Val Table. The Cumul Stor Val LP Param table values (Storage) are used as approximation points indexing the Cumul Stor Val Table. The Cumul Stor Val Table should have increasing values of Storage and Cumulative Value. Cumulative Storage Value is required to be a concave function of Anticipated Storage. The preferred order of approximation is substitution, piece-wise, tangent, two-point line. The Cumul Stor Val Linearization Automation category’s Marginal Value to Table and Lin method can automate creation of the Cumul Stor Val LP Param table and the Cumul Stor Val Table.

Anticipated Storage

Type: Agg Series Slot
Units: Volume
Description: The combination of the actual storage plus water that would be expected to enter the reservoir after the Current Timestep but has not yet, due to lagging.
Information:
Defined by:

Cumul Stor Val Table

Type: Table Slot
Units: Volume vs. Value
Description: The estimated total economic value of water stored in the reservoir for discrete storage values.
Information:
Defined by: User-input or by automated procedure if Cumul Stor Val Linearization Automation category has selected the Marginal Value to Table and Lin method.

Marginal Storage Value Table

Type: Table Slot
Units: Storage vs. $ Value
Description: Anticipated Storage versus Cumulative Storage Value per unit energy
Information: This table should be increasing in storage, and logically decreasing in marginal value
Defined by: Required input

Spill Cost

Type: Agg Series Slot
Units: $
Description:
Information:
Defined by:

5.9.2.10 Optimization Future Value

This category allows the optimization to provide slots relating to the future value of water. Its appearance is dependent on selecting the Cumulative Storage Value Table method for the Future Value category.

5.9.2.10.1 None

If this method is selected, the future value slots will not be visible, and no linearization will be attempted.

5.9.2.10.2 Opt Cumulative Storage Value Table

If this method is selected, the following slots will be visible, and linearization will be allowed.

SLOTS SPECIFIC TO THIS METHOD

☞ CUMUL STOR VAL LP PARAM

Type: Table Slot
UNITS: VOLUME, VOLUME, VOLUME
Description: Specifies the storage points used to take the tangent, line and piecewise approximations for Cumul Stor Val Table linearization
Information:
Defined by: User-input or by automated procedure if Cumul Stor Val Linearization Automation category has selected the Marginal Value to Table and Lin method.

5.9.2.11 Hydrologic Inflow

If any method in this category is selected, hydrologic inflow is included in the reservoir mass balance. Optimization assumes hydrologic inflow to be known (data) and it is not solved for by the reservoir regardless of the method selected. Click **HERE (Objects.pdf, Section 24.1.9)** for more information.

5.9.2.12 Live Capacity Available

The live capacity category conceptually represents the amount of available storage space between the current storage in the reservoir and a user defined maximum. As the pool storage in the reservoir increases, the live capacity decreases.

5.9.2.12.1 None

If this method is selected live capacity is not calculated for the reservoir.

5.9.2.12.2 Constraint and Variable

If this method is selected an additional decision variable and physical constraint are added to the optimization problem. See the figure in the Creditable Capacity Category.

☞ LIVE CAPACITY

Type: Agg Series Slot
UNITS: VOLUME
Description: The amount of storage space available between the current storage and a user defined upper limit. This upper limit is entered in the “max value” field of the Live Capacity slot configuration dialogue box.
Defined by: Explicit constraint

$$\text{Max Live Capacity} >= \text{Storage} + \text{Live Capacity} \qquad \textbf{(EQ 49)}$$

5.9.2.13 Optimization Spill

The Optimization Spill methods determine how spill is calculated for the reservoir and generates physical constraints that correspond to the selected methods.

This category is dependent on selection of the Independent Linearizations method for the Optimization Power category. However, the method selected in the Optimization Spill category must match with the corresponding non-optimization method in the Spill category.

Spill is an Optimization decision variable. The following constraint is always generated for a reservoir:

$$\text{Outflow} = \text{Turbine release (or Release)} + \text{Spill} \tag{EQ 50}$$

The spill methods generate values applicable to an additional constraint:

$$\text{Spill} = \text{Regulated Spill} + \text{Unregulated Spill} + \text{Bypass}, \tag{EQ 51}$$

where some of these terms may be omitted if they do not apply to the selected spill method.

Depending on the method selected, some of the following slots will be added. Other slots will be used from the non-optimization method selected. Please click **HERE (Objects.pdf, Section 24.1.3)** for details on non-optimization Spill methods.

As applicable, the following constraints are also added:

$$\text{Bypass} <= \text{Bypass Capacity} \tag{EQ 52}$$

$$\text{Regulated Spill} <= \text{Regulated Spill Capacity} \tag{EQ 53}$$

$$\text{Unregulated Spill} = \text{Unregulated Spill Capacity} \tag{EQ 54}$$

☞ **BYPASS CAPACITY**

Type: Series Slot
UNITS: FLOW
Description: Bypass capacity
Information:
Defined by: Numerical 2-D Approximation in terms of storage, based upon the Bypass Capacity Table.

☞ **BYPASS CAPACITY TABLE**

Type: Table Slot
UNITS: VOLUME VS. FLOW
Description: Storage vs. corresponding maximum bypass spill values
Information:
Defined by: Internally developed based on Bypass Table and Elevation Volume Table relationships. For each pool elevation in the Bypass Table, the Bypass Capacity Table has a row relating Storage to Bypass Capacity. The Pool Elevations are converted to Storage using the Elevation Volume Table.

☞ REGULATED SPILL CAPACITY

Type: Series Slot
UNITS: FLOW
Description: Regulated spill capacity
Information:
Defined by: Numerical 2-D Approximation in terms of storage, based upon the Regulated Spill Capacity Table.

☞ REGULATED SPILL CAPACITY TABLE

Type: Table Slot
UNITS: VOLUME VS. FLOW
Description: Storage vs. corresponding maximum regulated spill values
Information:
Defined by: Internally developed based on Regulated Spill Table and Elevation Volume Table relationships. For each pool elevation in the Regulated Spill Table, the Regulated Spill Capacity Table has a row relating Storage to Regulated Spill Capacity. The Pool Elevations are converted to Storage using the Elevation Volume Table.

☞ REGULATED SPILL OR BYPASS LP PARAM

Type: Table Slot
UNITS: VOLUME
Description: Specifies the Storage points use to take the tangent, line and piecewise approximations for Regulated Spill Capacity linearization and Bypass Spill Capacity linearization.
Information:
Defined by: User-input

☞ UNREGULATED SPILL LINEARIZATION TABLE

Type: Table Slot
UNITS: VOLUME VS. FLOW
Description: Storage vs. corresponding unregulated spill values
Information:
Defined by: Internally developed based on Unregulated Spill Table and Elevation Volume Table relationships. For each pool elevation in the Unregulated Spill Table, the Unregulated Spill Capacity Table has a row relating Storage to Unregulated Spill Capacity. The Pool Elevations are converted to Storage using the Elevation Volume Table.

☞ UNREGULATED SPILL LP PARAM

Type: Table Slot
UNITS: VOLUME
Description: Specifies the Storage points use to take the tangent, line and piecewise approximations for Unregulated Spill Linearization Table linearization
Information:
Defined by: User-input

5.9.2.13.1 None

If this method is selected the slot bounds in the slot configuration dialog are set to zero. No additional constraints are generated.

5.9.2.13.2 Opt Monthly Spill

This method is not functional in RPL Optimization.

This method sets the lower and upper bounds on spill. The lower bound is set to zero and the default upper bound is set to a very big number (9,999,999 cms). The default upper bound can be revised in the Spill slot configuration, Max Value parameter. No additional constraints are generated beyond these bounds.

5.9.2.13.3 Opt Unregulated

If this method is selected only unregulated spill is considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Unregulated Spill} \qquad \text{(EQ 55)}$$

5.9.2.13.4 Opt Regulated

When this method is selected only regulated spill is considered. The lower bound on spill is set to zero and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} \qquad \text{(EQ 56)}$$

5.9.2.13.5 Opt Regulated and Unregulated

When this method is selected unregulated and regulated spill are considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} + \text{Unregulated Spill} \qquad \text{(EQ 57)}$$

5.9.2.13.6 Opt Regulated and Bypass

When this method is selected only regulated and bypass spill are considered and the lower bound on spill is set to zero and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} + \text{Bypass} \tag{EQ 58}$$

5.9.2.13.7 Opt Regulated, Bypass and Unregulated

When this method is selected unregulated, regulated and bypass spill are considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Regulated Spill} + \text{Unregulated Spill} + \text{Bypass} \tag{EQ 59}$$

5.9.2.13.8 Opt Bypass, Regulated and Unregulated

When this method is selected unregulated, regulated and bypass spill are considered and the following constraint is added to the LP:

$$\text{Spill} = \text{Bypass} + \text{Regulated Spill} + \text{Unregulated Spill} \tag{EQ 60}$$

5.9.2.14 Pool Elevation Linearization Automation

This category is no longer supported in RPL optimization.

5.9.2.15 Release Capacity Numerical Approximation

When Release Capacity is a part of the Optimization problem, this slot is numerically approximated as a function of Pool Elevation (Numerical 2-D Approximation). The relationship between Pool Elevation and Release Capacity will come from the user-input Max Release Table. The table will be queried using user-input points defined in the Release LP Param table.

5.9.3 Modeling a Storage Reservoir

Steps to follow in setting up a Storage reservoir for optimization

1. Set up a running simulation model, **using methods that are compatible with optimization for Spill.**

2. Select the Optimization Spill method corresponding to the method selected in the Spill category.

3. Fill in the LP Param table related to Release.

4. If future value is needed, select a method in the Future Value category, followed by the Optimization Future Value category, and if desired, Cumul Stor Val Linearization Automation.

5. Selection of the remaining methods and linearizations can be done in any order. Pool elevation linearizations, Pool Elevation Linearization Automation, Evaporation and Precipitation, Optimize Evaporation Computation, Evaporation Linearization Automations, Bank Storage, Hydrologic Inflow,

Energy In Storage, and Diversion from Reservoir. The appropriate data must be entered for each method.

5.10 Thermal

The Thermal object represents the economics of power generation. In optimization, the thermal object can influence the solution. In addition to allowing a modeler to maximize the economic value of hydropower, the thermal object allows a modeler to write constraints and objectives on the power related multislots. For example, total system hydropower generation or capacity can be maximized or constrained to meet a minimum level. Click **HERE (Objects.pdf, Section 26)** for thermal object documentation.

5.10.1 User Methods in Optimization

Following are the user selectable methods that are available in optimization.

5.10.1.1 Preferred Customer Category

A group of reservoirs may be obligated to meet the power demands of "preferred customers" before this energy is used for other purposes. For example, if a group of reservoirs is owned by another entity, the energy demand of their customers must be met before the energy is coordinated with the other reservoirs in an economic objective, and in this sense, their customers are "preferred". In such a situation, some (and perhaps all) of these reservoirs are also treated as "allocated energy" that can be flexibly used in coordination with the remaining reservoirs.

The methods in this category model the energy needs of the preferred customers. The preferred customers are identified by membership in a "Preferred Customer" subbasin. This is a predefined type of subbasin. The methods in this category are described **HERE (Objects.pdf, Section 26.1.3)**.

Although still shown in the interface, this method no longer does anything in the RPL based optimization.

5.10.1.2 Regulation Category

Regulation is one of the ancillary services that hydropower plants can supply in addition to power to increase the reliability and flexibility of the power system to adjust to fluctuations in power demand and supply. When a plant is regulating it will follow the load within some prescribed band of power rather than generating a fixed amount of power. Typically, regulation is a valuable service that can be provided efficiently by hydropower compared to alternative power sources. In locations where regulation is marketed it usually commands a solid premium above the price of the power generated. This value is partially reduced by the increased maintenance costs associated with regulation. The methods in this category are described **HERE (Objects.pdf, Section 26.1.4)**

5.10.1.3 Load

These methods determine if hourly power load data should be part of the economic valuation of hydropower. Click **HERE (Objects.pdf, Section 26.1.1)** for more information.

5.10.1.4 Modified Load

The category is dependent on selecting the Hourly Load method from the Load category.

5.10.1.4.1 None

This is the default no-action method.

5.10.1.4.2 Calculate Modified Load

The slots for this method and the simulation calculations are described **HERE (Objects.pdf, Section 26.1.2.2)**

The **Modified Load** is defined by the following constraint:

$$\begin{aligned} \text{Load Energy} \le\ & \text{Modified Load} \\ & + \text{Hydro Generation} \\ & - \text{Preferred Customer Energy} \\ & + \text{Pumped Storage Generation} \\ & - \text{Pumped Storage Pumping} \\ & + \text{Allocated Detail Energy} \end{aligned} \tag{EQ 61}$$

when all of these power sources are selected. If any of the power sources are NaN in the **Modified Load Power Sources Used** table then the terms for those power sources are omitted from the constraint.

In the policy, the following types of sample statement can be added:

```
REPEATED MAXIMIN
    FOREACH (DATETIME date IN @“Start Timestep” TO @ “Finish Timestep”
        ADD CONSTRAINT Thermal.Modified Load[date] <= 0.0
    END FOREACH
END MAXIMIN
```

This constraint will minimize the largest values of modified load.

5.10.2 Economic Valuation User Methods

These methods model the economic valuation of hydropower during an optimization run using the Optimization controller. The category doesn’t exist in the Simulation and Rulebased Simulation controllers. The values are based either explicitly or implicitly on the replacement of thermal power sources and the associated cost savings. (The descriptions of possible optimization policy in this section are not intended to exactly show the syntax of the policy statements. This is in part because the policy syntax differs under the old Optimization controller and the new Optimization controllers.)

The methods in this category exactly parallel the methods in the Thermal Replacement Value category for simulation, with the exception that no attempt is made to attribute value to particular energy sources. Thus, simulation methods that are identical except for the order of evaluation map to a single optimization method. This optimization method is dependent on one of the appropriate simulation methods being selected.

All of the methods use the slot Avoided Operating Cost to represent the explicit or implicit savings to the thermal system from hydropower generation. The methods differ in how they compute Avoided Operating Cost. It is assumed that the user will write an objective function that either directly or indirectly maximizes Avoided Operating Cost.

In this description, anywhere it says Net Avoided Cost or Avoided Operating Cost, it should have either Thermal, Block, or Linear added to the front of it, i.e. Linear Net Avoided Cost, depending on method selections.

One possible objective function is

Objective 1: *Maximize Net Avoided Cost*

Net Avoided Cost is defined as the Avoided Operating Cost minus the value of energy used for generation and energy lost to spill:

$$NetAvoidedCost = AvoidedOperatingCost - FutureValueOfUsedEnergy - SpillCost$$

Future Value of Used Energy and Spill Cost are multislots that are assumed to be linked to the individual values on power reservoirs. The individual reservoir calculations use a power coefficient and a dollar value to translate releases into these costs.

An alternative to penalizing the Future Value of Used Energy and Spill Cost is to maximize the Total Cumulative Storage Value, a multislot with the expected value of all water remaining in storage. This multislot is expected to be linked to the Cumulative Storage Value on the individual reservoirs, where it is a piecewise linear function of storage. One possible objective is

Objective 2: *Maximize Avoided Operating Cost +*

Total Cumulative Storage Value (Final timestep)

Optionally, the objective can subtract the following term, Total Cumulative Storage Value (Initial timestep).

Objective 3: *Maximize Avoided Operating Cost +*

Total Cumulative Storage Value (Final timestep) -

Total Cumulative Storage Value (Initial timestep)

Subtracting this constant term does not effect the optimal solution, but the reported objective function value will now reflect the *change* in total Cumulative Storage Value, typically a more meaningful number.

In some cases a utility may be allocating additional energy from outside the basin. If data is specified for the slots described below, the optimization will automatically create variables for each column of the Allocated Detail Energy Slot. These variables are constrained to minimum and maximum values:

$$\mathrm{AllocatedMinimum}(t) \le \mathrm{AllocatedDetailEnergy}(t) \le \mathrm{AllocatedMaximum}(t)$$

The Allocated Detail Energy is further constrained to meet periodic totals where the period is a multiple of the timesteps of the run. For example, a model with a 6-hour timestep, might have a total daily energy target. More generally, the period is specified by the timesteps with non-zero values for Allocated Total Energy. The Allocated Detail Energy is required to meet the totals. For a daily period the constraint is

$$\sum_{t \in \mathrm{day}} \mathrm{AllocatedDetailEnergy}(t) = \mathrm{AllocatedTotalEnergy}(\mathrm{day}) \quad \forall \mathrm{days}$$

The user may further constrain allocated energy by selecting the Preferred Customer method from the Preferred Customer Category

5.10.2.1 Thermal Unit Replacement Value

5.10.2.1.1 None

No evaluation is made.

5.10.2.1.2 Calculate Thermal Unit Replacement Value

The economic value of hydropower is set equal to (or replaced by) the savings from replacing generation from thermal power sources.

A constraint is added to meet the load. This load must be met (or exceeded) by a combination of the following power sources: thermal energy, power reservoir energy, allocated energy (from outside the basin), and net energy from pumped storage facilities (generation - pumping).

$$\sum_{\mathrm{units}} \mathrm{ThermalEnergy}(\mathrm{unit},t) + \mathrm{HydroGeneration}(t) + \mathrm{AllocatedDetailEnergy}(t) + \tag{EQ 62}$$

$$\mathrm{PumpStorageGeneration}(t) - \mathrm{PumpStoragePumping}(t) \ge$$

$$\mathrm{LoadEnergy}(t) + \mathrm{PreferredCustomerEnergy}(t) \quad \forall t$$

The constraint is written as an inequality rather than an equality to allow for piecewise linearization of power. The assumption is that an objective function will be written that will tend to maximize the energy generated for a given amount of turbine release. For example, an objective function minimizing the cost of thermal generation would be sufficient.

Additional terms are available for writing more complex objective functions. In this method, Avoided Operating Cost is defined as the reduction in the cost of thermal generation, the thermal costs of meeting the load with only thermal sources minus the cost of a combined hydropower and thermal solution.

$$\text{ThermalAvoidedOperatingCost}(t) = \text{ThermalOnlyCost}(t) - \sum_{\text{units}} \text{ThermalUnitCost}(\text{unit},t) \times \text{ThermalEnergy}(\text{unit},t) \quad \forall t \qquad \text{(EQ 63)}$$

Thermal Energy is further constrained by unit capacity and availability:

$$0 \le \text{ThermalEnergy}(\text{unit},t) \le \text{ThermalUnitAvailabilities}(\text{unit},t) \times \text{ThermalUnitCapacties}(\text{unit},t) \quad \forall \text{units}, t$$

SLOTS SPECIFIC TO THIS METHOD:

The slots for this method are described in the Thermal Object documentation **HERE (Objects.pdf, Section 26.1.7.2)**

5.10.2.2 Block Economic Value

5.10.2.2.1 None

No evaluation is made.

5.10.2.2.2 Calculate Block Economic Value

The economic value of hydropower is set equal to (or replaced by) a piecewise linear function of hydropower generated. The piecewise function can vary by time period. While there is no direct linkage to thermal sources of power, one interpretation of the piecewise linear function is that it represents the coordinated reduction in other power sources as a result of hydropower generation. This method does not attempt to meet an explicit load, rather the load is assumed to be included in the process that generates the piecewise linear functions.

The piecewise linear functions are represented by two slots: maximum of Hydro Block Use slot (same for all t) and Hydro Block Costs. The Hydro Block Costs are the slope of the piecewise linear segments, or alternatively the value of a unit of power. The piecewise linear function is assumed to be concave: the costs are assumed to decrease as the block number increases.

The method name indicates which hydropower resources should be evaluated by the piecewise linear function. Any power sources that precede the “Block” keyword are assumed to already be included in the piecewise linear function and are not evaluated. The order of power sources after the “Block” keyword doesn’t matter for optimization. In this case, the power sources included are

1. Allocated energy
2. Conventional hydropower

3. Pumped storage units (both pumping and generation)

Unevaluated: None.

A constraint is added to limit the Hydro Block Use credits to the net generation from all sources minus the Preferred Customer Energy demands that must first be met by the generation.

$$\sum_{\text{blocks}} \text{HydroBlockUse(block,t)} \le \text{HydroGeneration}(t) + \text{AllocatedDetailEnergy}(t) + \text{PumpStorageGeneration}(t) - \text{PumpStoragePumping}(t) - \text{PreferredCustomerEnergy}(t) \quad \forall t \tag{EQ 64}$$

The constraint is written as an inequality rather than an equality to allow for piecewise linearization of power. This constraint is triggered by writing an objective function that includes maximizing Avoided Operating Cost. In addition to this constraint, Avoided Operating Cost is set equal to (or replaced by) the piecewise linear value of power:

$$\text{BlockAvoidedOperatingCost} = \sum_{\text{blocks}} \text{HydroBlockCosts(block,t)} \times \text{HydroBlockUse(block,t)} \tag{EQ 65}$$

SLOTS SPECIFIC TO THIS METHOD:

The slots for this method are described in the Thermal Object documentation **HERE (Objects.pdf, Section 26.1.6.2)**

5.10.2.3 Linear Economic Value

5.10.2.3.1 None

No evaluation is made.

5.10.2.3.2 Calculate Linear Economic Value

The economic value of hydropower is set equal to (or replaced by) a linear function of hydropower generated. The function can vary by time period. This method, in essence, models a block valuation but with only one block.

$$\text{LinearAvoidedOperatingCost} \le \text{LinearHydroCosts}(t) \times (\text{HydroGeneration}(t) + \text{AllocatedDetailEnergy}(t) + \text{PumpStorageGeneration}(t) - \text{PumpStoragePumping}(t) - \text{PreferredCustomerEnergy}(t)) \quad \forall t \tag{EQ 66}$$

The constraint is written as an inequality rather than an equality to allow for piecewise linearization of power. This constraint is triggered by writing an objective function that includes maximizing Linear Avoided Operating Cost.

SLOTS SPECIFIC TO THIS METHOD:

The slots for this method are described in the Thermal Object documentation **HERE (Objects.pdf, Section 26.1.5.2)**

5.10.3 Modeling a Thermal Object

Steps to follow in converting a model prior to 5.1 to the approach used by 5.1:

1. Extract model information using an old executable

1.1 Load the model

1.2 Deactivate any policy referencing the Thermal Object

1.3 Save the Opt Goal set: we will load it later

1.3 Record the selected thermal method: we will later use the power source ordering

1.4 Record Avoided Operating Cost upper and lower bounds

1.5 Record the Block End Tolerance: slot is replicating and changing to scalar slot

1.6 Record the reporting method: we will reinstate it.

2. Put information into model using 5.1 executable

2.1. Change the thermal methods as desired for each Linear, Block, or Thermal category

2.1.1 Set the power source ordering for each selected method

e.g. Nan for sources that used to come before Block or Therm in the old method name

1,2, etc. for sources after Block or Therm

2.1.2 Avoided Operating Cost for each selected method, e.g. Linear Avoided Operating Cost

View Configuration

Lower Bound = 0 (or available from earlier executable unless this is a new method)

Upper Bound = 1e7 (for example) (or available from earlier executable unless this is a new method)

2.1.3 Import costs if they weren't transferred from the previous model for some reason

2.1.4 Set Hydro Block Use bounds for block method if missing

2.1.5 Set (Block or Thermal) End Tolerance

2.2. Set the reporting methods for simulation as desired (optional)

One reporting category for each selected power economic evaluation method

- Typically set the method in this category to report the same thing used in an optimization objective function

2.3 Load the goal set and replace any thermal slot references to old slots with the new method-specific versions of the slots

3. The Total Hydro Capacity slot on the Thermal object should have values by the end of simulation either by linking to Hydro Capacity on power reservoirs (plant power methods) or by setting the slot to input (Unit Power Table method). The economic value of power generation can be limited by Total Hydro Capacity. In particular, negative or invalid values for Total Hydro Capacity result are treated as zero capacity.

6. Setting up and Running an Optimization Model

This section provides a broad outline of the process of creating a RiverWare model designed for use with the Optimization controller.

6.1 Select Timestep and Run Range

The timestep and run range for Optimization are set in the **Run Control** dialog just as for Simulation models. The same timesteps are available for Optimization as for Simulation; however due to the nature of Optimization, some additional issues should be considered when setting the timestep and run range for Optimization.

During an optimization run a linear program (LP) is created which represents the physical and policy constraints for the system. The time required to solve the LP is related to the size of the model, including the number of objects, the number of timesteps in the run, and the number of constraints. Thus to permit a simulation that runs in an acceptable time, one might need to consider increasing the timestep length and/or shortening the run duration. It is important to realize that the run time for Optimization does not increase linearly with the number of timesteps. The run time tends to increase with something between the square and the cube of the number of timesteps.

Another reason for limiting the duration of an optimization run is that long optimization runs can exhibit “perfect foresight.” That is, since each LP solution optimizes over the full run duration, it can incorporate behavior that would not be plausible for operators who at any point in time do not have perfect knowledge of future events. Thus as a rule of thumb, optimization models should have a run duration no longer than the time frame for which accurate forecasts are likely to exist for the river basin operations.

Note: Although all timestep lengths are available for Optimization, it is not recommended that a monthly timestep be used with Optimization. Optimization does not have a concept of “current timestep.” Thus a 30 day month is used for all calculations and constraints that incorporate the timestep length, for example, conversions between volume and flow and between power and energy. As such, the mass balance in the Optimization solution will not match the mass balance of the Post-Optimization Rulebased Simulation for a monthly timestep.

6.2 Set Up and Test a Simulation Model

Before running a model with the Optimization controller, the model should be set up and run in Simulation mode. This allows for checking that all required input data have been provided and also allows for calibration of the physical model. A model should be validated using Simulation (or Rulebased Simulation) before trying to use it for Optimization. Also, recall that a complete Optimization run actually consists of a sequence of three runs using the Simulation, Optimization and Rulebased Simulation controllers (see **HERE (Section 3.1)**), so a working Simulation model is a requirement for running with Optimization.

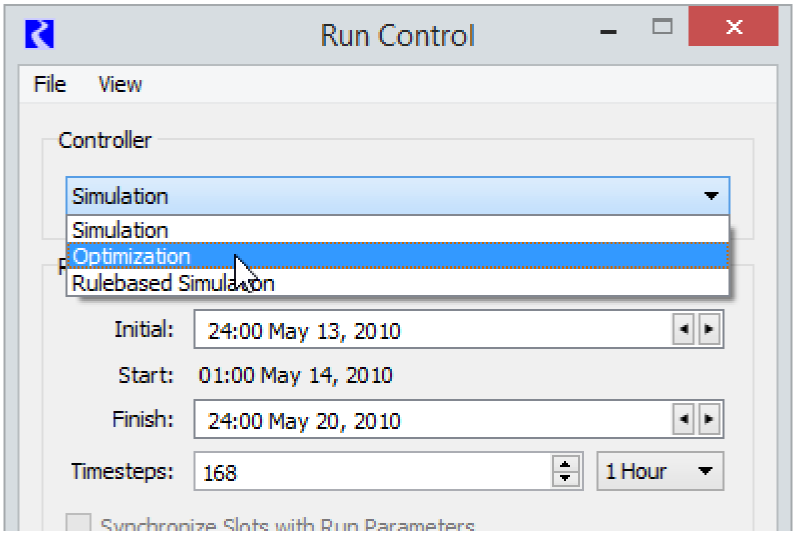
One common approach is to run the model in Simulation using historic Inflow and Outflow data as inputs and allowing the model to calculate Storage, Pool Elevation, Power, etc, which would be checked against the historic values. Then the Outflow inputs would be removed when running in Optimization to let the Optimization solution set the Outflows.

Note: When setting up a Simulation or Rulebased Simulation model that will eventually be used for Optimization, it is important to keep in mind that not all objects and Simulation methods are supported by Optimization. Thus the object and method selections in the Simulation model must be made such that they are compatible with Optimization. The objects and methods that are supported by Optimization are described in the section on Objects and Methods HERE (Section 5)

6.3 Select Optimization Methods

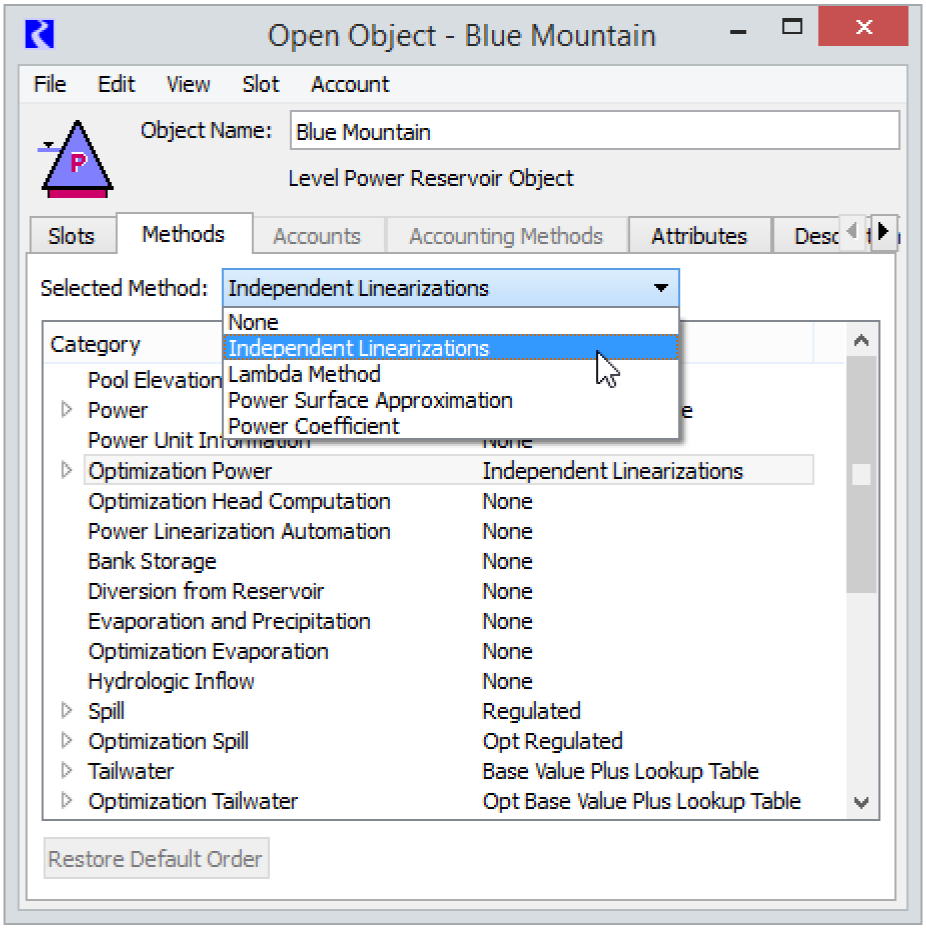
Many method categories apply only to the optimization controller, and some method selections which are appropriate for Simulation or Rulebased Simulation might not be so for an optimization model. Therefore, one must visit several categories and make appropriate selections. The optimization specific categories and methods are described **HERE (Section 5)**. It is also important to note that not all Simulation objects are supported in Optimization. Only the objects listed in **HERE (Section 5)** can be included in an Optimization model.

The Optimization method categories are only visible on simulation objects when the Optimization controller is the active controller selected on the **Run Control** dialog.



Initial: 24:00 May 13, 2010
Start: 01:00 May 14, 2010
Finish: 24:00 May 20, 2010
Timesteps: 168 1 Hour

Once the Optimization has been selected, Optimization method selections can be made in the same manner as Simulation methods on the Methods tab of each simulation object (or through the mulit-object method selector).



Object Name: Blue Mountain

Level Power Reservoir Object

Selected Method: Independent Linearizations

Category	
Pool Elevation	
Power	
Power Unit Information	None
Optimization Power	Independent Linearizations
Optimization Head Computation	None
Power Linearization Automation	None
Bank Storage	None
Diversion from Reservoir	None
Evaporation and Precipitation	None
Optimization Evaporation	None
Hydrologic Inflow	None
Spill	Regulated
Optimization Spill	Opt Regulated
Tailwater	Base Value Plus Lookup Table
Optimization Tailwater	Opt Base Value Plus Lookup Table

In some cases, a single method is selected for both Simulation and Optimization. For example, if the Time Lag routing method is selected for a reach object, then the appropriate routing constraints will be added in Optimization based on the Time Lag provided for the Simulation method. There is no need in this case to select an additional method for Optimization.

In other cases, an Optimization method must be selected that corresponds to a Simulation method. For example if the Base Value Plus Lookup Table method is selected for the Tailwater category, then Opt Base Value Plus Lookup Table should be selected in the Optimization Tailwater category. The optimization method will add slots that will be used to linearize the Tailwater variables. The linearization will use a combination of slots specific to the Optimization method as well as the Tailwater Tables from the Simulation method. More information can be found on the selection of appropriate methods for Optimization in the section on Objects and Methods **HERE (Section 5)**.

6.4 Add a Thermal Object

A single thermal object models basin-wide quantities such as the total value of storage across all reservoirs in the model. The thermal object also includes economic modeling of basin-wide power generation. A thermal object is not required for an Optimization model, but it can facilitate the modeling of common hydropower objectives. Click **HERE (Section 5.10)** for more information on the thermal object.

The thermal object acts as a summarizing object for many variables that may be included in an optimization policy. It summarizes information by linking to each of the reservoir or other objects of interest. For example, to compute the total energy produced in the model, the Thermal.Hydro Generation slot should be linked to each PowerReservoir.Energy slot. Other possible links include:

- Thermal.Hydro Generation <------> PowerReservoir.Energy
- Thermal.Total Cumulative Storage Value <------> Reservoir.Cumulative Storage Value
- Thermal.Spill Cost <------> Reservoir.Spill Cost
- Thermal.Future Value of Used Energy <------> Reservoir.Future Value of Used Energy
- Theraml.Energy in Storage <------>Reservoir.Energy in Storage
- Thermal.Hydro Capacity <------> Reservoir.Hydro Capacity

If a thermal object is used in the model, these links will need to be created.

6.5 Enter Lower and Upper Bounds

6.5.1 Bounds on Variables

Upper and lower bounds must be entered for all series slots that are decision variables. The values should be reasonable. They represent the absolute minimum and maximum for the slot. Often these minimum and maximum values are based on the actual physical minimum and maximum. For example, the Pool Elevation Lower Bound should be the bottom of the reservoir and the Upper Bound should be the top of the reservoir as found in the Elevation Volume table. Turbine Release should be limited to zero as the lower bound and turbine capacity for the upper bound. In other cases, the absolute physical minimum or maximum may not be well-defined, such as maximum Inflow (or if economic variables are included in the model).

These variable bounds remain constant and are automatically converted by RiverWare into hard constraints on the variable for every time step in the optimization solution. In these cases variables without an obviously defined minimum or maximum, the slot bounds should not unnecessarily constrain the problem. These types of variables will typically be limited, either directly or indirectly, through the constraints in the Optimization Goal Set. It is also important, however, that the slot bounds not be set arbitrarily large or small (i.e. orders of magnitude different than the realistic limits) as this can result in numerical instability due to scaling issues in the optimization problem. There is a tendency to enter data just to get the model to run. This can lead to later errors and difficulty debugging when these values inadvertently limit the solution of the problem.

It is important that slot bounds not be confused with policy constraints, nor be used to apply policy constraints. For example, a hydro plant might have a minimum generation requirement that it must meet, but the Power slot Lower Bound should still be zero. The minimum generation requirement would be set through a constraint in the optimization goal set. The slot bounds become hard constraints in the optimization problem and essentially remain constant in the model, whereas policy constraint values (usually stored in custom slots) are typically incorporated as soft constraints and may change for different runs.

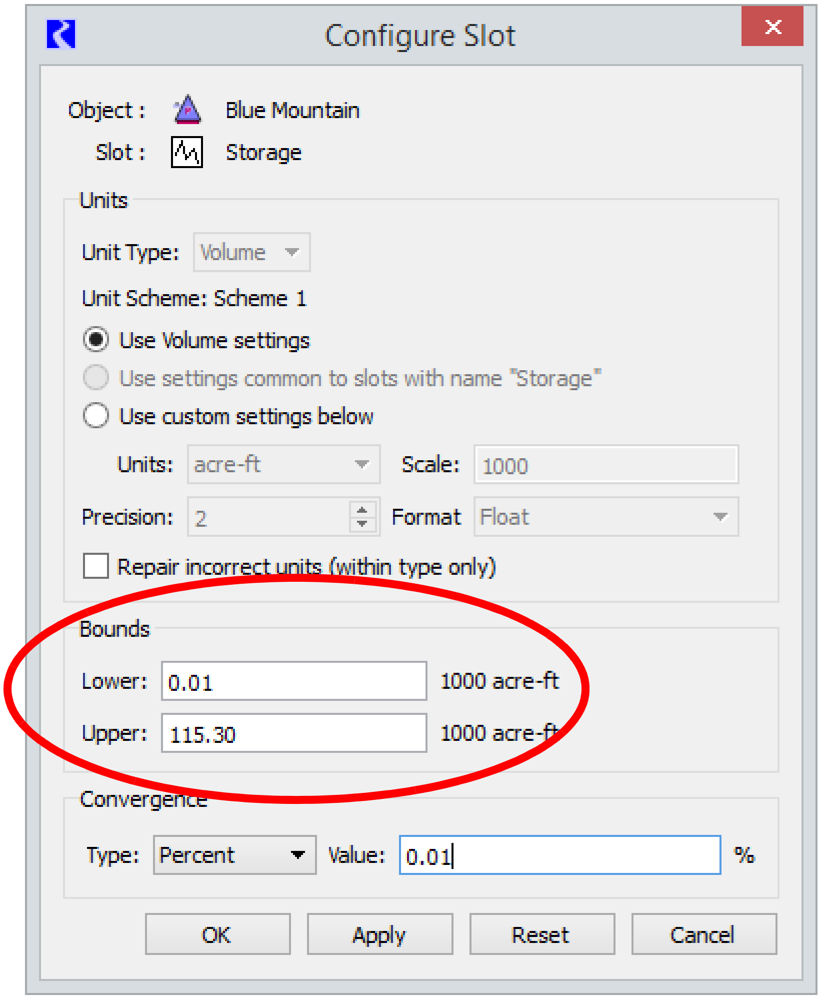
The slot bounds do not affect the Simulation solution. They are only applied in the Optimization solution. The Simulation solution is limited only by the physical characteristics described in table slots such as the Elevation Volume Table and Plant Power Table. Often it is helpful to set the slot bound slightly inside min/max values for the corresponding data table. This prevents problems of exceeding values in the data table associated with approximation error when going from Optimization to the Post-optimization RBS. For example, if the highest Pool Elevation in the Elevation Volume Table slot is 1000 ft, the Pool Elevation slot Upper Bound should be set to something like 999.95 ft. For more information on how variable bounds are used in the solution, click **HERE (Section 4.2.4)**.

Setting Bounds:

Note: On Agg Series Slots, a value must be entered for each column.

There are two ways to set the Bounds.

1. From each slot, open the Configuration dialog (**View** ➥ **Configure**) and enter the Upper and Lower Bound.



Object : Blue Mountain
Slot : Storage

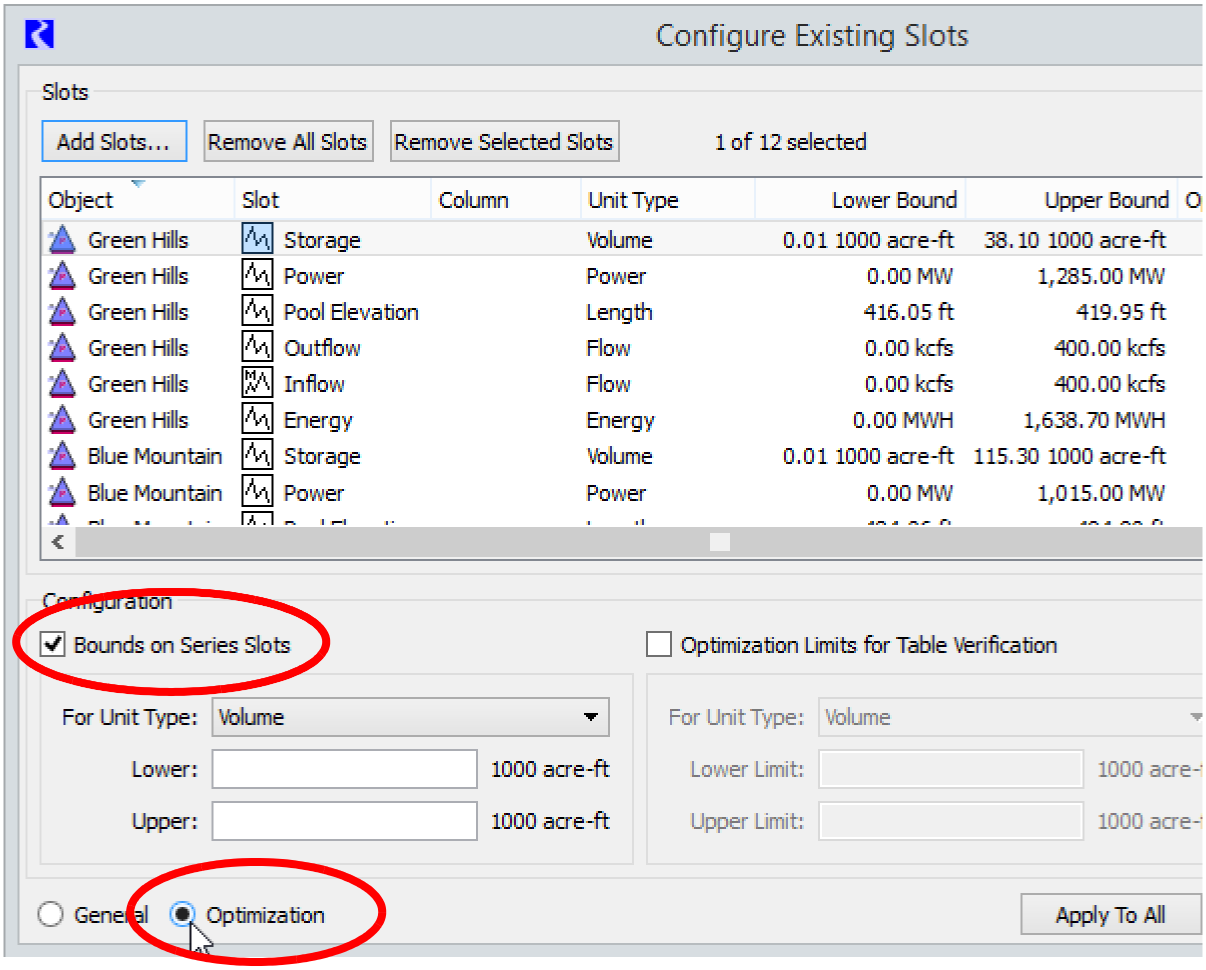
Units
Unit Type: Volume
Unit Scheme: Scheme 1
● Use Volume settings
○ Use settings common to slots with name "Storage"
○ Use custom settings below
Units: acre-ft
Scale: 1000
Precision: 2
Format Float
☐ Repair incorrect units (within type only)

Bounds
Lower: 0.01 1000 acre-ft
Upper: 115.30 1000 acre-ft

Convergence
Type: Percent
Value: 0.01 %

2. Values on multiple slots can be set at once using the global slot configuration dialog accessed from the **Workspace** ➥ **Slots** ➥ **Configure Slots** menu, described **HERE (Slots.pdf, Section 3)**. Select the

radio button for Optimization in the bottom left, and make sure that the box for Bounds on Series Slots is checked.



1 of 12 selected

Object	Slot	Column	Unit Type	Lower Bound	Upper Bound
Green Hills	Storage		Volume	0.01 1000 acre-ft	38.10 1000 acre-ft
Green Hills	Power		Power	0.00 MW	1,285.00 MW
Green Hills	Pool Elevation		Length	416.05 ft	419.95 ft
Green Hills	Outflow		Flow	0.00 kcfs	400.00 kcfs
Green Hills	Inflow		Flow	0.00 kcfs	400.00 kcfs
Green Hills	Energy		Energy	0.00 MWH	1,638.70 MWH
Blue Mountain	Storage		Volume	0.01 1000 acre-ft	115.30 1000 acre-ft
Blue Mountain	Power		Power	0.00 MW	1,015.00 MW

Configuration

☑ Bounds on Series Slots

For Unit Type: Volume

Lower: 1000 acre-ft

Upper: 1000 acre-ft

☐ Optimization Limits for Table Verification

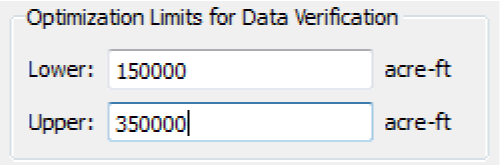
For Unit Type: Volume

Lower Limit: 1000 acre-

Upper Limit: 1000 acre-

○ General ● Optimization

6.5.2 Optimization Limits for Data Verification



Optimization Limits for Data Verification

Lower: 150000 acre-ft

Upper: 350000 acre-ft

Certain Table Slots are verified to ensure they are increasing or satisfy requirements for concavity. Some tables meet the more stringent requirements of Optimization when a reservoir is within a “normal” operating region but violate these conditions during extreme events such as a flood when optimization would not be used. Users can optionally limit data checking in Optimization to the normal region by specifying Lower and Upper Limits. This allows extreme values to be in the table so that the model can be used for extreme events, but the extreme values are not used for Optimization purposes and they are ignored in Optimization verification in order to prevent the verification from failing. The specified range is configured in the Optimization Limits for Data Verification in the Table Slot configuration (shown in the screenshot) or using the Global Slot Configuration dialog described above (check the box for Optimization Limits for Table Verification).

Some tables are automatically derived from other tables by RiverWare. For example spill tables based on elevations are automatically converted to spill tables based on storage. When the original tables have upper/lower limits, RiverWare automatically converts the original limits to their equivalent values for the derived tables, e.g. from elevation limits to storage limits.

For 3-dimensional tables, this logic may be applied both to the range of z values and to x or y within a block of constant z values.

Only select table slots have limited data checking enabled. This is the current list of those slots for Reservoirs if the slots exist based on controller and method selections:

- Bypass Table
- Elevation Volume Table
- Energy In Storage Table
- Marginal Storage Value Table
- Regulated Spill Table
- Unregulated Spill Table
- Volume Area Table

The following tables are automatically created for Reservoirs by RiverWare by converting elevation based spill tables to storage based tables.

- Bypass Capacity Table
- Regulated Spill Capacity Table
- Unregulated Spill Linearization Table

The limits on these tables are set automatically based on the original tables' limits.

In addition to these slots, Power Reservoirs have the following slots verified if the slots exist based on controller and method selections:

- Maximum Turbine Q
- Plant Power Table
- Stage Flow Tailwater Table
- Tailwater Table

The following tables are automatically created for Power Reservoirs by RiverWare by converting elevation based spill tables to storage based tables.

- Auto Max Turbine Q
- Convolved Stage Flow Tailwater Table

The limits on these tables are set automatically based on the limits in the Plant Power Table and the Stage Flow Tailwater Table respectively.

If the limits values are omitted on a table slot, then there will be no lower limit or upper limit respectively on checking that the table conforms to optimization requirements. The full range is checked.

6.6 Provide Approximation Points

When one modeled quantity (slot) is related to one or two other modeled quantities by a non-linear function, then that function must be approximated as one or more linear functions. RiverWare supports three basic ways of approximating a two dimensional function:

- Tangent - the tangent to the function at a given point
- Secant - the line passing through two given points
- Piecewise - several lines connecting points on the function

Details about each approximation technique can be found **HERE (Section 4.3.4)**.

Select Optimization methods use their own linearization method that does not depend on the tangent, secant and piecewise approximations. The information in this section does not apply for those methods. In the Optimization Power category, the Power Coefficient method **HERE (Section 5.5.2.15.3)** and the Power Surface Approximation method **HERE (Section 5.5.2.15.4)** are two such methods. For those methods, a separate set of approximation parameters are entered as described in the section on each method. In the Optimization Tailwater category, the Opt Coefficients Table method **HERE (Section 5.5.2.19.5)** uses the same table of coefficients as the corresponding Simulation method to define Tailwater Elevation (i.e. the Simulation method is already a linearized expression of Tailwater).

6.6.1 Data Table Slots

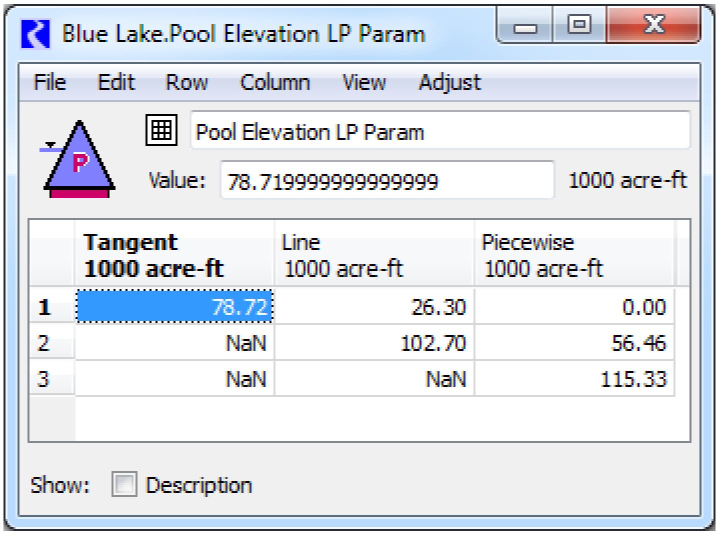
Each slot being approximated is related to another slot by a data table. Each row of the table provides a point in the curve describing the relation between the two quantities. In the case of 3-dimensional functions, multiple curves are described in the table.

- 2-D example: Elevation Volume Table, relates Elevation to Storage for a Reservoir.
- 3-D example: Plant Power Table, describes Power as a function of the Operating Head and Turbine Release. The table is organized as several curves, corresponding to Power as a function of Turbine Release for various Operating Heads.

Optimization uses the same data tables that are used in Simulation, or generates the required table automatically based on tables used in Simulation. Thus, a model that runs in Simulation should contain all of required data in data tables to run in Optimization.

6.6.2 LP Param Table Slots

Linear programming parameter tables (LP Param Tables) are specific to Optimization and are required input for an Optimization model. These slots become visible when the Controller is switched to Optimization. The LP Param table slots contain the points to be used when each approximation technique is applied. Setting the values for the LP Param tables can be one of the more challenging aspects of setting up an Optimization model. Setting poor approximation points can result in model infeasibilities that are difficult to debug. An example of a Pool Elevation LP Param table is shown below, followed by basic guidelines for setting LP Param table values..



Pool Elevation LP Param

Value: 78.719999999999999 1000 acre-ft

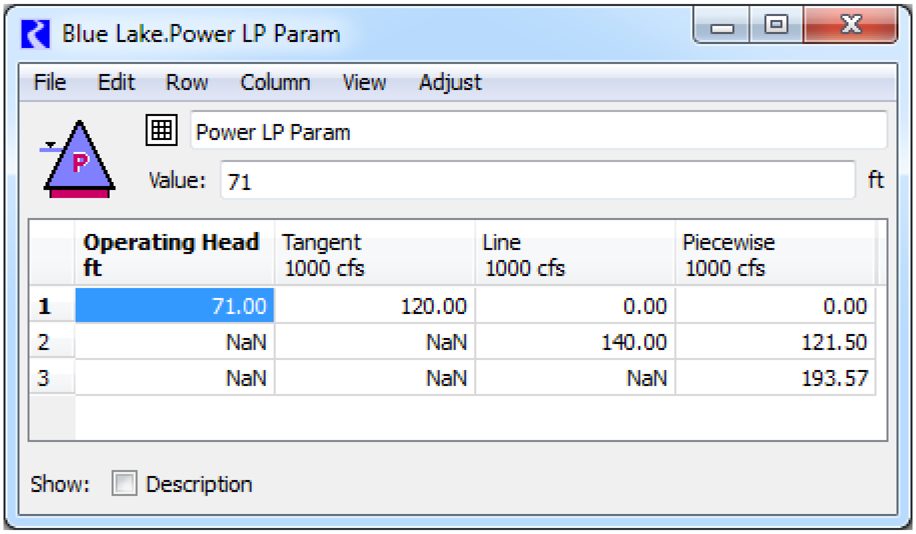
	Tangent 1000 acre-ft	Line 1000 acre-ft	Piecewise 1000 acre-ft
1	78.72	26.30	0.00
2	NaN	102.70	56.46
3	NaN	NaN	115.33

Show: ☐ Description

- The Tangent approximation requires a single point. The single approximation point for the tangent approximation should typically be set in the middle of the range of values expected to occur during the current run. The Tangent approximation will overestimate the approximated value for a concave function and underestimate the approximated value for a convex function. See **HERE (Section 4.3.4.2)**.
- The Line (Secant) approximation requires two points. They should typically be set near either end of the range of expected values. The Line approximation will overestimate a function at some points and underestimate the function at other points. See **HERE (Section 4.3.4.3)**.
- The Piecewise approximation requires two or more points. See **HERE (Section 4.3.4.4)**. For piecewise approximation, often it is sufficient to choose the three points used for the Tangent and Secant approximations. In other cases, finer granularity might be required.

Note: Points are entered into the LP Param tables in terms of the independent variable(s) not the dependent variable. For example, the values in the Pool Elevation LP Param table slot are in terms of Storage, not Pool Elevation.

For three dimensional approximations, the approximation point for the second independent variable (first column in the table) should be somewhere near the middle of the expected operating range. An example of a three-dimensional LP Param table is shown below. For the Power approximation, Turbine Release is the first independent variable and has points set for the Tangent, Line and Piecewise approximations. Operating Head is the second independent variable. A single value is set for Operating Head in the Power LP Param table, and Power is calculated based on this single Operating Head in the Optimization run.



Power LP Param

Value: 71 ft

	Operating Head ft	Tangent 1000 cfs	Line 1000 cfs	Piecewise 1000 cfs
1	71.00	120.00	0.00	0.00
2	NaN	NaN	140.00	121.50
3	NaN	NaN	NaN	193.57

6.6.3 Additional Guidelines

6.6.3.1 Turbine Capacity Approximation

The Turbine Capacity linearization will always use the Piecewise approximation. Poor Turbine Capacity approximation points can lead to runs that fail in the Post-optimization Rulebased Simulation due to flows that exceed the actual Turbine Capacity in RBS, once the approximation error is removed. When setting the Piecewise points in the Turbine Capacity LP Param table slot, four criteria should be met.

- First, the piecewise curve must be concave. The optimization solution will not behave correctly if the piecewise curve is not concave, and thus the optimization run will abort with an error if the approximation points define a piecewise curve that is not concave.
- Second, the approximation points should cover the full range of operating heads in the Plant Power Table. If the points do not cover the full range, the optimization solution will use a linear extrapolation beyond the highest and lowest approximation points. This can result in excessive approximation error for upper and lower operating heads.
- Third, the piecewise curve defined by the approximation points should never be greater than the curve defined by the Plant Power Table (Auto Max Turbine Q slot). This can result in the optimization solution allowing turbine releases that exceed the actual Turbine Capacity in the post-optimization simulation. In other words, the piecewise curve from the LP Param approximation should always be under the curve defined by the Auto Max Turbine Q table.
- Fourth, the approximation points should be set such that the difference between the piecewise curve and the Auto Max Turbine Q curve is minimized. This reduces the approximation error in the optimization solution. If the difference between the two curves is excessively large, the optimization solution may overly restrict the turbine release to a capacity significantly lower than the actual turbine capacity for the given operating head.

One strategy than can help with picking points for the Turbine Capacity approximation is to plot the Auto Max Turbine Q slot and visually identify reasonable approximation points. The Auto Max Turbine Q slot is automatically populated by RiverWare at the start of the run (even if the run fails). It might be necessary to run the model fist in order to populate the table.

Because the Turbine Capacity linearization always uses the piecewise approximation, it is not necessary to set points for the tangent and line approximations. If values are set for these other two approximations, it will not cause a problem. They will simply never be used. If in doubt, set points for all of the approximations and let RiverWare select the appropriate approximation points to use. If values are missing for an approximation that RiverWare tries to use, the run will abort with an error message notifying the user of the missing values.

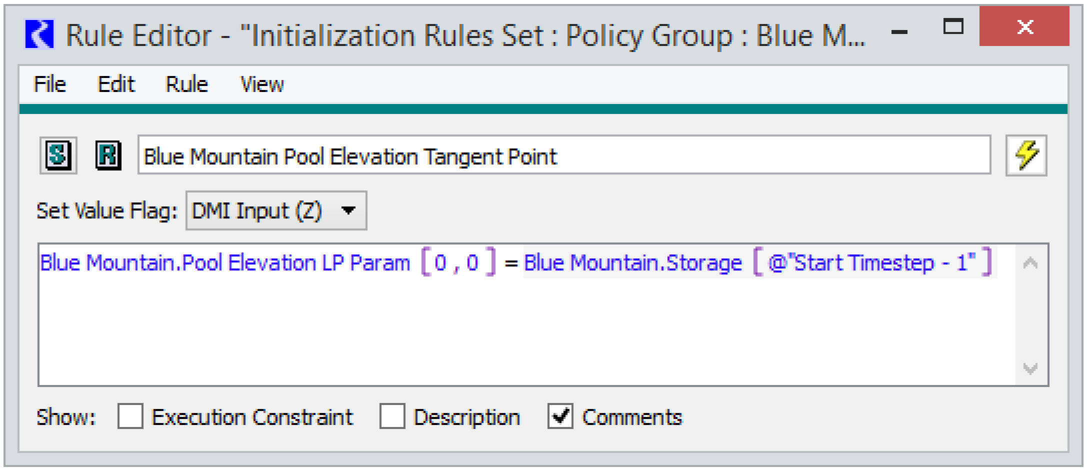
6.6.3.2 Power Approximation

The power approximation tends to be the linear approximation that introduces the most approximation error into the optimization solution. If the Independent Linearizations method is selected in the Optimization Power category, the Power LP Param table is used to define the Power approximation. The table contains an additional initial column for Operating Head in addition to the Tangent, Line and Piecewise columns with Turbine Release values. In the Optimization solution, a single piecewise linear curve for Power as a function of Turbine Release is used. This corresponds to one block of constant head in the Plant Power Table. Operating head is held constant at the value specified in the Power LP Param table. The remaining columns in the Power LP Param table set the Turbine Release for the approximation points.

Typically the Operating Head value in the Power LP Param table should be set to the average Operating Head expected in the run.However, in some cases it might be less problematic to overestimate Power than to underestimate Power. In these cases, it might make sense to set the Operating Head value in the Power LP Param table near the lower end of the expected operating range. This would cause hte Optimization solution to underestimate Power, and thus the Post-optimization Rulebased Simulation will calculate a large Power value than the Optimization.

6.6.3.3 Initialization Rules to Set Approximation Points

n some cases, it might make sense to set the approximation points based on the specific run conditions. For example, for a reservoir that has a large seasonal variation in Pool Elevation, it might not be possible to set Tangent and Line approximation points in the Pool Elevation LP Param table or an Operating Head point in the Power LP Param table that will result in sufficiently small approximation error across all conditions. In these cases, initialization rules can be used to set the approximation points such that the approximation error will be small for the expected operating range of the given run but would be large if operating in a different range in another season. For example, an initialization rule could set the Pool Elevation LP Param Tangent point equal to the initial Storage for the run. An example initialization rule is shown below.


Rule Editor - "Initialization Rules Set : Policy Group : Blue M...
File
Edit
Rule
View
S
R

Blue Mountain Pool Elevation Tangent Point

Set Value Flag: DMI Input (Z)

```
Blue Mountain.Pool Elevation LP Param [ 0 , 0 ] = Blue Mountain.Storage [ @"Start Timestep - 1" ]
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

A second rule could set the two Line points to the Tangent point plus or minus some expected delta. In this manner, the approximations will be updated automatically when the model runs, rather than needing to adjust the values manually at the start of each run.

6.7 Create an Optimization Goal Set

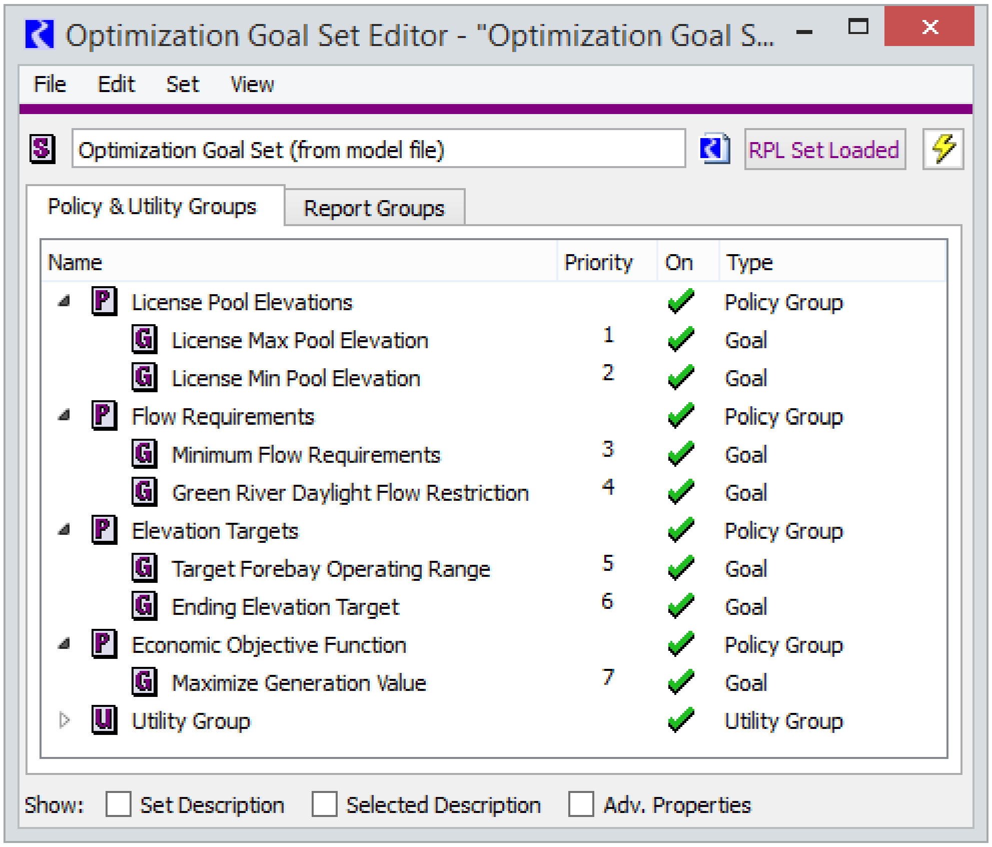
The Optimization policy must be cast into a set of prioritized goals. Like a RBS Ruleset, an Optimization Goal Set is a prioritized list of goals (written in RPL). However, whereas the rules in a RBS Ruleset are all executed at each timestep and set values, Optimization Goals are each executed once during a run from highest priority to lowest priority and have the effect of modifying or solving the current linear programming problem.

RBS rules are normally ordered to execute in an upstream to downstream order, object-by-object. Generally there is no need to order Optimization goals in an upstream to downstream order. The priorities of the goals should correspond to their “true” priorities in operations. The highest priority goals should correspond to the most important operating constraints, those that should essentially never be violated.

A single goal can contain constraints for multiple objects, but it is generally recommended that a single goal correspond to a single operating policy. For example, the highest priority goal might contain the license maximum Pool Elevation constraints for all reservoirs at all timesteps. The second goal might contain the license minimum Pool Elevation constraints for all reservoirs at all timesteps. This is generally better practice than placing the minimum and maximum elevation constraints in the same goal or placing the minimum elevation constraint in the same goal (same priority) as a minimum flow constraint. When multiple policies are combined in a single goal, it is not always obvious how the policies will trade off with one another if it is not possible to satisfy all constraints at that priority.

Medium priority goals generally contain constraints that you expect to get satisfied most of the time but that may not be possible to satisfy under some operating conditions, for example under extreme high or low flows. Lower priority goals typically contain target operating constraints, targets that you would like to meet under ideal conditions but are expected to be frequently violated due to higher priority constraints and input data (e.g. inflows). For example, you might have a low priority goal to keep flows equal across all reservoirs on each timestep, but most of the time you cannot keep flows exactly equal due to other requirements on the system. Typically the lowest priority contains an objective function to optimize with the remaining degrees of freedom given all higher priority constraints. For example, the objective might maximize the total value of hydropower generation during the run within the operational limits specified by higher priority constraints.

A simple example of an Optimization Goal Set is shown below.



Optimization Goal Set (from model file)

Name	Priority	On	Type
License Pool Elevations		✓	Policy Group
License Max Pool Elevation	1	✓	Goal
License Min Pool Elevation	2	✓	Goal
Flow Requirements		✓	Policy Group
Minimum Flow Requirements	3	✓	Goal
Green River Daylight Flow Restriction	4	✓	Goal
Elevation Targets		✓	Policy Group
Target Forebay Operating Range	5	✓	Goal
Ending Elevation Target	6	✓	Goal
Economic Objective Function		✓	Policy Group
Maximize Generation Value	7	✓	Goal
Utility Group		✓	Utility Group

Additional information about the Optimization Goal Set is given **HERE (Section 4.1)**, and details about the individual statements that make up Optimization Goals are provided **HERE (Section 4.1.3)**.

6.8 Create a Post-optimization Ruleset

RiverWare will automatically generate a Post-optimization Ruleset if there is no ruleset loaded when switching from the Optimization controller to the Rulebased Simulation controller upon completing a run with the Optimization controller. The automatically generated ruleset contains rules to set each reservoir Outflow slot to the Outflow value from the Optimization solution.

Reservoir.Outflow[] = OptValue(Reservoir.Outflow, @"t")

See details about the automatically generated ruleset **HERE (Section 3.2.1)**.

In many cases it is necessary to customize the Post-optimization Ruleset to do more than simply set reservoir Outflows to the Optimization values. Some common uses for a custom ruleset might be:

- a need to set Turbine Release and Spill independently based on their Optimization values rather than just total Outflow.
- if other water uses, such as diversions, need to be specified in the RBS based on the Optimization solution.
- to refine the Optimization solution further in the Post-optimization RBS, possibly to make small adjustments to the flows once outputs have been calculated with the approximation errors removed.
- to add post-processing calculations to the Post-optimization rules.

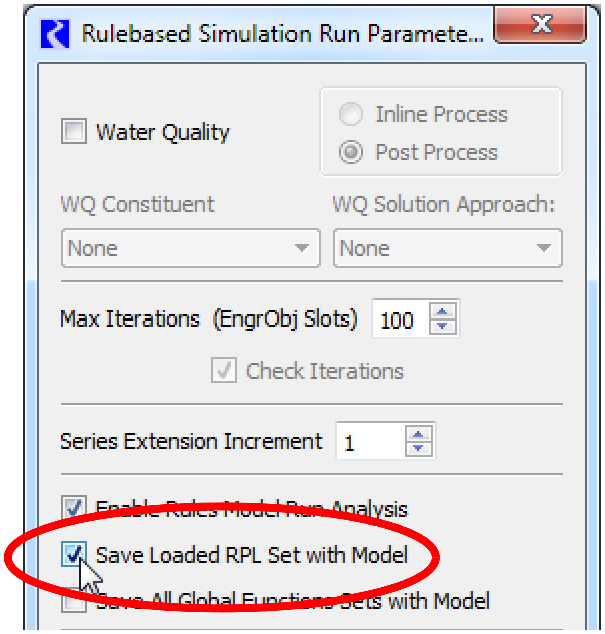
A useful approach for creating a custom Post-optimization Ruleset is to begin with the automatically generated set. Then revise the rules as needed using the basic structure of the automatically generated set.

While the Post-optimization Ruleset is a special case of a ruleset used specifically in the context of an Optimization run, from a technically perspective, it is the same as any RBS Ruleset. As such it can contain any components that might be in a standard RBS Ruleset. For example, there is nothing to technically prevent the Post-optimization Ruleset from completely overwriting the Optimization solution. It is the use of the OptValue function when setting the slot in the ruleset that ties to the Post-optimization Rulebased Simulation to the Optimization solution. The Post-optimization Ruleset can make use of as much or as little of the Optimization solution as desired as long as it adheres to all standard requirements for a RBS Ruleset.

That being said, the standard approach is to have rules set flows based on the Optimization solution and let RiverWare calculate all remaining values in Simulation. When using this standard approach, it is strongly recommend to write the rules such that they maintain the overall mass balance from the Optimization solution. This is because the feasibility of the Optimization solution is dependent on the mass balance used in the Optimization solution. In addition the satisfaction of policies such as Pool Elevation constraints in the Optimization solution is dependent on the mass balance.

As an example, Post-optimization Rules that adjust the solution by making shifts between Spill and Turbine Release but use the same total Outflow as the Optimization solution would maintain the mass balance from the Optimization solution. Whereas rules that modified the total Outflow from the Optimization solution would alter the mass balance. In the later case, it would be possible to have a run where the Storage exceeded the limits in the Elevation Volume Table in the Post-optimization Rulebased Simulation, causing the run to abort, even though it was within the required limits in the Optimization solution (or less severely violate elevation constraint limits in the Post-optimization Rulebased Simulation when they were satisfied in the Optimization solution). Of course it is possible to include logic in the rules to check for these issues, but the more this type of logic is included in the ruleset, the more the solution will tend to deviate from the Optimization solution and become more of a Rulebased Simulation solution. Depending on the intended use of the model, this may or may not be acceptable.

Once a custom Post-optimization Ruleset has been created, it can be saved as a separate file just like a standard RBS Ruleset by selecting **File ➥ Save RBS Ruleset As ...** from the Ruleset Editor dialog and/or it can be saved in the model file. Once a Ruleset has been saved with the model file, then the next time the model is opened, the Ruleset will automatically be opened and loaded. To save the Ruleset with the model file it must first be loaded. Then on the **Run Control** dialog select **View ➥ Rulebased Simulation Run Parameters...**. In the resulting dialog, check the box for **Save Loaded RPL Set with Model**. Close the dialog, and save the model with the Ruleset now included.



Rulebased Simulation Run Paramete...

☐ Water Quality
○ Inline Process
● Post Process

WQ Constituent: None
WQ Solution Approach: None

Max Iterations (EngrObj Slots) 100
☑ Check Iterations

Series Extension Increment 1

☑ Enable Rules Model Run Analysis
☑ Save Loaded RPL Set with Model
☐ Save All Global Functions Sets with Model

6.9 Run the Model

Every run should be made as follows:

1. Switch the run controller to **Simulation**, and click Start. (Click **HERE (Section 3.1.1)** for more details.)

After the Simulation run, most slots of interest will still contain NaN.

2. Switch the run controller to **Optimization**, and click Start. (Click **HERE (Section 3.1.2)** for more details.)

The Optimization run will take considerably longer than the Simulation run. During the Optimization run, the Run Status dialog will report which priority is currently executing form the Optimization Goal Set rather than a timestep as for Simulation. After the Optimization run, all slots that contained NaN after the Simulation run will continue to display NaN. The Optimization solution is only stored internally and has not been returned to the workspace.

3. Switch the run controller to **Rulebased Simulation**. If no Ruleset is loaded either accept the option to (automatically) generate and load a Post-optimization Ruleset, or alternatively load a custom ruleset, and then click Start. (Click **HERE (Section 3.1.3)** for more details.)

After running the Post-optimization Rulebased Simulation, all slots of interest should be populated with data.

Note: It is important to always run Simulation before running Optimization. Running Optimization without first running Simulation will not produce the desired results as data might be leftover from a previous run, or necessary preprocessing of data by initialization rules or expression slots will not be carried out.

An alternative to automate the three run sequence is to use a RiverWare script with actions to Set Controller and Execute Run for each of the three individual runs. Click **HERE (ScriptManagement.pdf, Section 2)** for details on how to configure and use a script.

7. Debugging and Solution Analysis

The main tools to use when debugging an optimization run are:

- The RPL debugger to investigate the policy. This is described in the RPL debugging section, **HERE (RPLDebugging.pdf, Section 2)**.
- The Priority-Oriented Optimization Solution Analysis Tool which provides information on the optimization solution, described below.

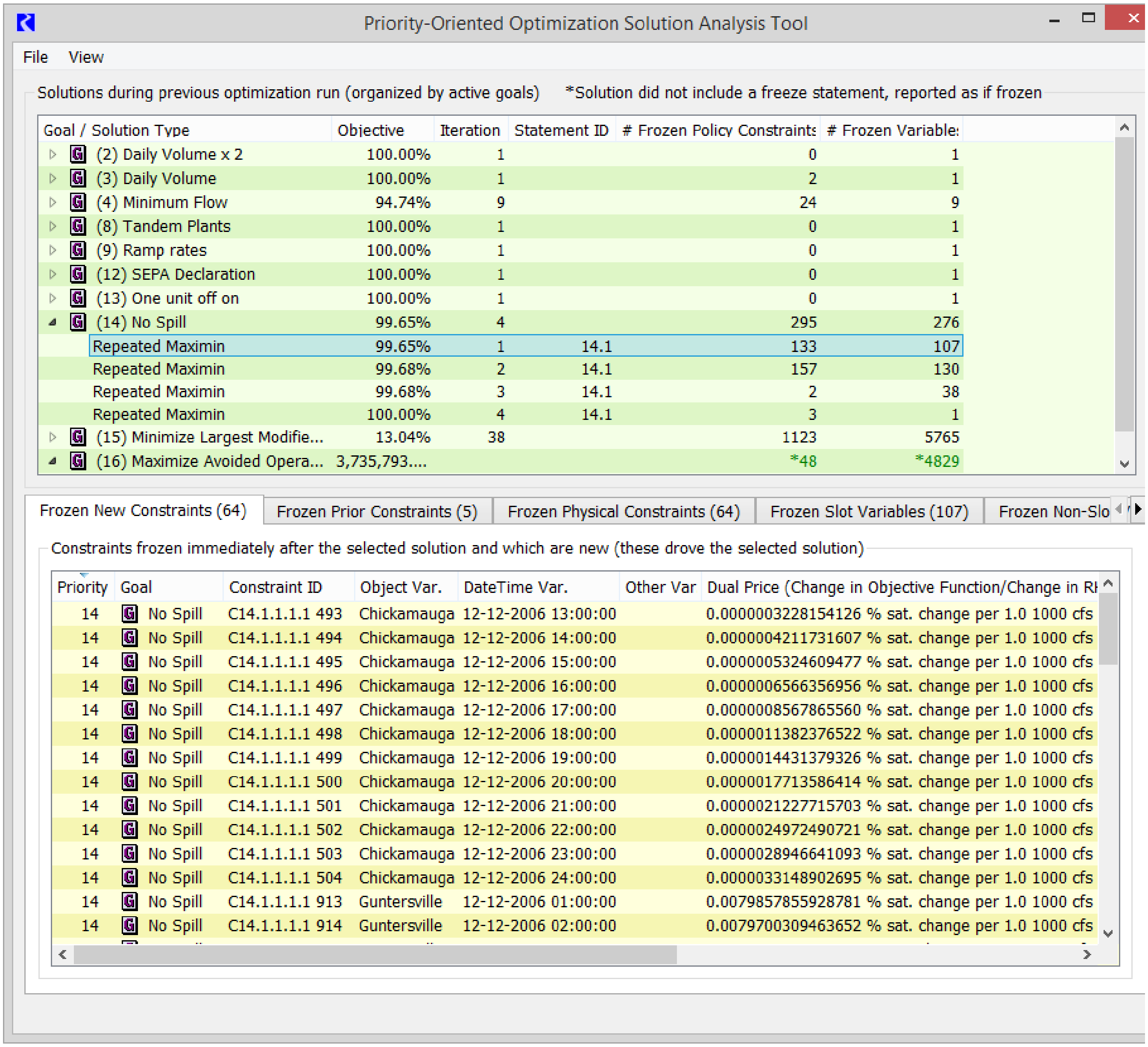
This section describes the solution analysis tool and other debugging features.

7.1 Priority-Oriented Optimization Solution Analysis Tool

The Priority-Oriented Optimization Solution Analysis Tool provides detailed information from an optimization run.This includes:

- Objective values
- Satisfaction of derived objectives (e.g. Repeated Maximin) from goals with soft constraints
- Constraints frozen with each solution
- Variables frozen with each solution
- Dual prices of frozen constraints
- Reduced costs of frozen variables

This information is valuable for understanding the optimization solution. For example, observing which constraints were frozen immediately after a given solution indicates which policies were driving the solution. Information from the tool can also be helpful for debugging problems with an optimization run. For example, constraints that were frozen when not expected can at times indicate policy in the goal set that is having unintended consequences.



Solutions during previous optimization run (organized by active goals) *Solution did not include a freeze statement, reported as if frozen

Goal / Solution Type	Objective	Iteration	Statement ID	# Frozen Policy Constraint	# Frozen Variable
(2) Daily Volume x 2	100.00%	1		0	1
(3) Daily Volume	100.00%	1		2	1
(4) Minimum Flow	94.74%	9		24	9
(8) Tandem Plants	100.00%	1		0	1
(9) Ramp rates	100.00%	1		0	1
(12) SEPA Declaration	100.00%	1		0	1
(13) One unit off on	100.00%	1		0	1
(14) No Spill	99.65%	4		295	276
Repeated Maximin	99.65%	1	14.1	133	107
Repeated Maximin	99.68%	2	14.1	157	130
Repeated Maximin	99.68%	3	14.1	2	38
Repeated Maximin	100.00%	4	14.1	3	1
(15) Minimize Largest Modifie...	13.04%	38		1123	5765
(16) Maximize Avoided Opera...	3,735,793....			*48	*4829

Frozen New Constraints (64)
Frozen Prior Constraints (5)
Frozen Physical Constraints (64)
Frozen Slot Variables (107)
Frozen Non-Slo

Constraints frozen immediately after the selected solution and which are new (these drove the selected solution)

Priority	Goal	Constraint ID	Object Var.	DateTime Var.	Other Var	Dual Price (Change in Objective Function/Change in R
14	No Spill	C14.1.1.1.1 493	Chickamauga	12-12-2006 13:00:00		0.0000003228154126 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 494	Chickamauga	12-12-2006 14:00:00		0.0000004211731607 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 495	Chickamauga	12-12-2006 15:00:00		0.0000005324609477 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 496	Chickamauga	12-12-2006 16:00:00		0.0000006566356956 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 497	Chickamauga	12-12-2006 17:00:00		0.0000008567865560 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 498	Chickamauga	12-12-2006 18:00:00		0.0000011382376522 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 499	Chickamauga	12-12-2006 19:00:00		0.0000014431379326 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 500	Chickamauga	12-12-2006 20:00:00		0.0000017713586414 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 501	Chickamauga	12-12-2006 21:00:00		0.0000021227715703 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 502	Chickamauga	12-12-2006 22:00:00		0.0000024972490721 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 503	Chickamauga	12-12-2006 23:00:00		0.0000028946641093 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 504	Chickamauga	12-12-2006 24:00:00		0.0000033148902695 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 913	Guntersville	12-12-2006 01:00:00		0.0079857855928781 % sat. change per 1.0 1000 cfs
14	No Spill	C14.1.1.1.1 914	Guntersville	12-12-2006 02:00:00		0.0079700309463652 % sat. change per 1.0 1000 cfs

The tool has two panels. The top panel, the Solution Information panel, provides an overview of the solutions that occurred during a single optimization controller run. The tabs of the lower panel provide additional details; the focus of these details is controlled by user selection within the Solution Information panel.

The information displayed by the tool is:

- Collected during every optimization run.
- Invalidated and completely cleared by deletion of objects or slots from the workspace.
- Cleared whenever a new optimization run is initiated.

- Corresponds to the current run when paused within a run; otherwise it corresponds to the previous optimization run.
- Only guaranteed to display correctly when the currently loaded goal set was the previous run's goal set.
- Partially saved in the model file. The entire set of information can be saved to an external file and reloaded for later use.

During an optimization run, the display updates when the run is paused, stopped, aborted, or ends.

7.1.1 Accessing the Analysis Tool

When information about the last optimization run is available, the Priority-Oriented Optimization Solution Analysis Tool can be accessed in the following ways:

- From the **Run Control** dialog: **View ➥ Optimization Solution Analysis...**
- From the **Model Run Analysis** dialog: **File ➥ Optimization Solution Analysis...**
- From the Workspace: **Utilities ➥ Optimization Solution Analysis...**

7.1.2 Save/Load Problem

The information associated with the Solution Information panel is automatically saved within the model file. The information associated with the remaining panels is likely to be large and so is not saved with the model, but it can optionally be saved to a file in the optimization output directory.

To automatically save the problem each time:

- Open the **Run Control** dialog
- Select the **Optimization** controller
- Select **View ➥ Optimization Run Parameters...**
- Check the box for **Save Final Optimization Problem**

When this option is selected, the constraint and variable information will be saved to a file in the optimization output directory when the model is saved.

To save the problem interactively for later use, use the **File ➥ Save Problem** operation to write the problem to a file.

To load the saved information, after opening the relevant model and loading the goal set, open the Solution Analysis Tool and select **File ➥ Load Problem...** and choose the desired file.

Note: In order for the File ➥ Load Problem utility to function correctly, the active goals and their priorities in the problem being loaded must match those of the most recent optimization problem currently displayed in the Solution Information panel (i.e. must be from a run with the same goal set or at least a set that is the same at the top level of goal names and priorities). If the loaded problem is from a different goal set, one with different goals, different goal activation and/or different prioritization than the most recent solution displayed in the

Solution Analysis Tool, then the information displayed will not be consistent with the loaded problem.

In most cases, the **Priority-Oriented Optimization Solution Analysis Tool** will be used for a run just completed and this step of saving and loading the optimization solution is not necessary.

7.1.3 Solution Information Panel

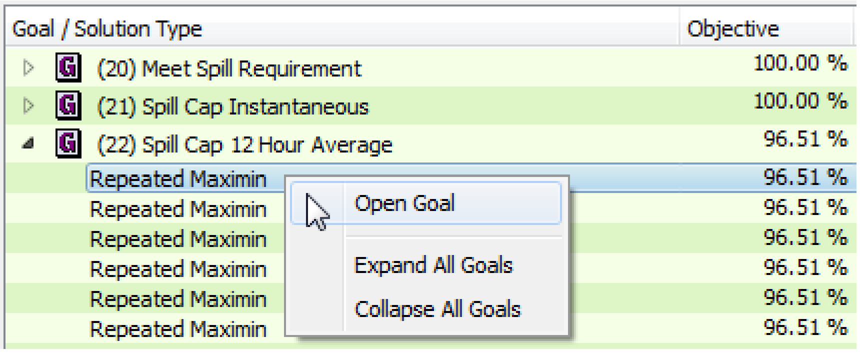
The Solution Information Panel presents an overview of the optimization solution. The top level displays the active goals in the loaded goal set, in priority order. Clicking on the arrow at the left of each these “goal rows” will expand the row to display information about each solution corresponding to that goal (a “solution row”).

The following is a description of the information in each column in the Solution Information Panel:

Column Name	Goal Row	Solution Row
Goal / Solution Type	Priority and name of the goal	Type of solution from the executed statement in the associated goal, one of: Maximize Objective, Minimize Objective, Summation, Repeated Maximin, Single Maximin, Mixed Integer
Objective	The value of the first solution row that has a value; for Maximize or Minimize objectives this is the objective value; for Repeated Maximin it is the percent satisfaction of the derived objective on the first iteration	The objective value (with units); if the associated statement was a Maximin this is the percent satisfaction of the derived objective.
Iteration	For goals with Repeated Maximin, the total number of iterations; otherwise no value is displayed	For Repeated Maximin, the solution index (iteration number); otherwise no value is displayed
Statement ID	No value displayed	The first number indicates the priority of the goal. Subsequent numbers correspond to levels within the goal. (e.g. 22.1 indicates the first statement in goal 22).

Column Name	Goal Row	Solution Row
# Frozen Policy Constraints	The number of policy constraints introduced by goals that were frozen while executing the goal (all iterations); physical constrains are not included in the count	The number of policy constraints introduced by goals that were frozen immediately after the corresponding solution; physical constrains are not included in the count. See also the following section.
# Frozen Variables	The number of variables frozen while executing the goal (all iterations)	The number of variables frozen immediately after the corresponding solution. See also the following section.

Right-clicking on a row in the Solution Information Panel will open a context menu that will allow to you open the corresponding goal.



Goal / Solution Type	Objective
(20) Meet Spill Requirement	100.00 %
(21) Spill Cap Instantaneous	100.00 %
(22) Spill Cap 12 Hour Average	96.51 %
Repeated Maximin	96.51 %
Repeated Maximin	96.51 %
Repeated Maximin	96.51 %
Repeated Maximin	96.51 %
Repeated Maximin	96.51 %
Repeated Maximin	96.51 %

7.1.3.1 Solutions without a Freeze Statement

In a typical policy, most problem solutions are immediately followed by the freezing of solution limiting variables and constraints. This freezing occurs automatically only for solutions that occur during execution of a Repeated Maximin statement; for freezing to occur after solutions caused by statements of other types, the policy must contain an explicit Freeze statement.

When a problem solution is not immediately followed by a freeze before the next solution, it is sometimes useful to know which constraints and variables would have been frozen had a Freeze Statement been executed. Thus, for these solutions the “# Frozen Constraints” and “# Frozen Variables” columns contain the number of constraints and variables that would have been frozen had the solution been immediately followed by a Freeze statement. These totals are displayed with a preceding asterisk (*) and a differently colored font. Note also that when more than two solutions occur before a freeze, a single variable or constraint might have been frozen at multiple solutions, but is reported for only the earliest solution.

7.1.4 Frozen Constraints Tabs

The lower panel presents five tabs. The first three present lists of constraints that were frozen:

- **Frozen New Constraints:** Constraints frozen immediately after the selected solution, which were introduced by the goal at this priority (these drove the selected solution)

- **Frozen Prior Constraints:** Constraints frozen immediately after the selected solution and which were introduced earlier
- **Physical Constraints:** Non-policy constraints frozen immediately after the selected solution; typically physical constraints or variable definition constraints automatically added to the optimization problem by RiverWare

Information in the Frozen Constraints tabs corresponds to the currently selected solution in the Solution Information panel. If no solution is selected, the Frozen Constraints tabs will be blank. Also, if no constraints were frozen as a result of the selected solution, the Frozen Constraints Tabs will be blank.

The following is a description of the information in each column in the Frozen Constraints tabs (note that not all columns are included in all three tabs):

Column Name	Description
Priority	Priority of the goal that added the constraint to the optimization problem
Goal	Name of the goal that added the constraint to the optimization problem
Constraint ID	The ID of the constraint, for example C4.1.2.2.1 45. The first number indicates the goal priority number and each subsequent number corresponding to a statement at each different level within the goal. The final number is the individual permutation of variables from a set of nested FOR loops.
Object Var.	If the constraint is within a FOR statement, and if one of those statements contains an index variable of type Object, then this is the value of the outermost Object index.
DateTime Var.	Similar to the Object Var. column but for DateTime variables
Other Var.	The values of any FOR statement index variables not covered by the previous two columns
Dual Price(Change in Objective Function/ Change in RHS)	The change in the objective function value per unit change in the right-hand side of the constraint. The units given are the units of the objective per units of the right-hand-side (user units).
Dual Price (raw)	The dual price value as reported by CPLEX
Shrink to ID	The ID of the constraint to which this constraint is shrinking, if there is one (i.e. a constraint added earlier that differs only in a less restrictive right-hand-side value)
Satisfaction	The percent satisfaction of the constraint

Clicking on any column header will sort the table by that column.

Columns can be reordered by clicking and dragging on the column header.

Right-clicking on a constraint row will present an option to “Open Constraint’s Goal.” For constraints that were shrunk to another constraint (a value is present in the Shrink to ID column), there will also be an option to “Open Shrink To Constraint’s Goal.”

7.1.5 Frozen Variables Tabs

The remaining two tabs present information about Frozen Variables:

- **Frozen Slot Variables:** Variables associated directly with slots frozen immediately after the selected solution (these limited its objective function, are at bounds)
- **Frozen Non-Slot Variables:** Variables not associated directly with a slot frozen immediately after the selected solution (these limited its objective function, are at bounds)

Information in the Frozen Variables tabs corresponds to the currently selected solution in the Solution Information panel. If no solution is selected, the Frozen Variables tabs will be blank. Also, if no variables were frozen as a result of the selected solution, the Frozen Variables tabs will be blank.

The Frozen Slot Variables tab has the following columns:

Column Name	Description
Object	The object associated with the variable
Slot	The slot associated with the variable (just the slot name)
Date/Time	The timestep associated with the variable
Value	The value in the solution (in internal optimization units)
Reduced Cost (Change in Objective Function/ Change in Variable)	The change in the objective function if the variable changes by 1 unit. The units given are the units of the objective per unit of the variable (user units).
Reduced Cost (raw)	The reduced cost value as reported by CPLEX

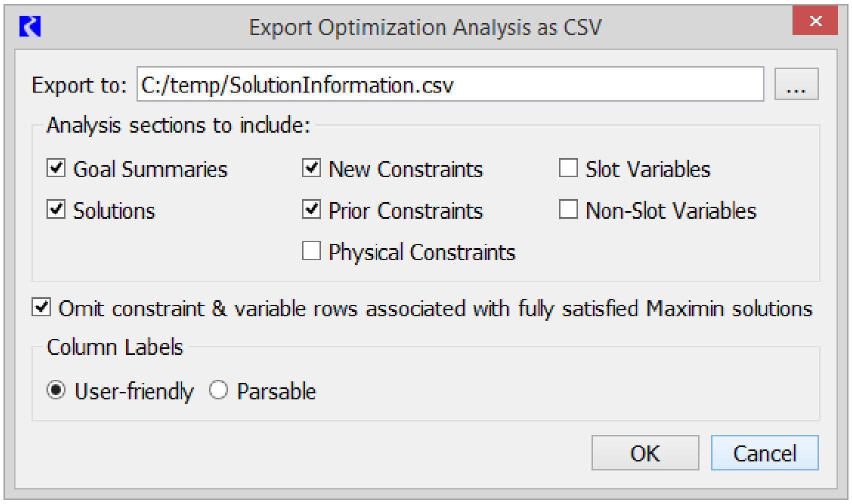
The Frozen Non-Slot Variable tab has a single Name column in place of the Object, Slot and Date/Time columns.

Clicking on any column header will sort the table by that column.

Columns can be reordered by clicking and dragging on the column header.

7.1.6 Export as CSV

If you would like to export the information in the analysis tool to a comma separated values (CSV) file, use the **File ➥ Export Analysis (CSV)...** menu. The following dialog opens where you can configure the file output.



Export to: C:/temp/SolutionInformation.csv

Analysis sections to include:

☑ Goal Summaries
☑ Solutions
☑ New Constraints
☑ Prior Constraints
☐ Physical Constraints
☐ Slot Variables
☐ Non-Slot Variables

☑ Omit constraint & variable rows associated with fully satisfied Maximin solutions

Column Labels

● User-friendly ○ Parsable

First, specify the file to use for export in the upper **Export to:** field.

Then, choose the **Analysis sections to include**. Each checkbox results in one or more rows for each optimization goal priority in the file.

The columns of the CSV file correspond to the columns shown in the analysis tool, presented in the tables above.

Specify whether to **Omit constraint & variable rows associated with fully satisfied Maximin solutions**. When checked, rows representing a fully satisfied solution are omitted.

Finally, for the column labels, choose whether you would like:

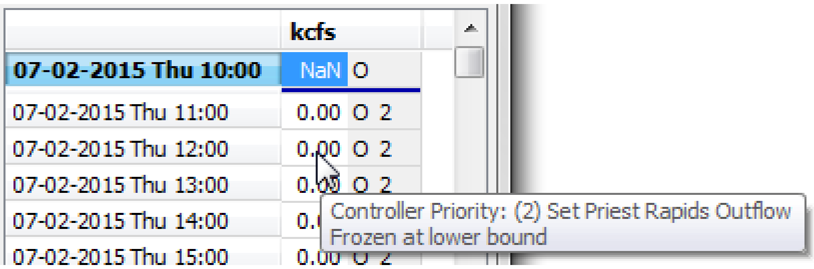
- **User-friendly** with spaces between words
- **Parsable** where all spaces are removed.

Click OK to generate the CSV file or Cancel to close without creating the file.

7.2 Tool Tips on Variables

Tool tips provide information about frozen variables. The information is shown on the slot and SCT dialogs when you hover over the slot value.. This occurs when:

- the associated variable was frozen at the slot upper or lower bound.



	kcfs	
07-02-2015 Thu 10:00	NaN	O
07-02-2015 Thu 11:00	0.00	O 2
07-02-2015 Thu 12:00	0.00	O 2
07-02-2015 Thu 13:00	0.00	O 2
07-02-2015 Thu 14:00	0.	
07-02-2015 Thu 15:00	0.00	O 2

Controller Priority: (2) Set Priest Rapids Outflow
Frozen at lower bound

- the associated variable was the only variable in a constraint that was frozen. When a constraint is frozen, RiverWare internally converts the inequality constraint to an equality constraint for the remainder of the solution. When there is only a single variable in the constraint, this is conceptually the equivalent of freezing the variable.

Examples include:

- **Frozen at lower bound**

This type of message occurs if the variable is frozen at one of the slot bounds.

- **Frozen by (3) Minimum Load 47.6% between limits set by 3.1.1.1 and 2.1.1.1.**

This type of message occurs if the variable is in a soft constraint that was not fully satisfied. In this example, the best the solution could do was get 47.6% of the way from the old limit set on the variable in goal 2 (statement 2.1.1.1) and the new limit set by goal 3 (statement 3.1.1.1).

- **Frozen by (22) Maximize Generation Value at a limit set by 4.1.1.1.**

The third example occurs when the variable is in a constraint that is fully satisfied but forced to be tight (frozen at the constraint limit) by a lower priority constraint or objective. In this example, assume that the variable is Pool Elevation, and goal 4 has a constraint statement (4.1.1.1) that constrains Pool Elevation to be greater than or equal to some value (a minimum Pool Elevation). The tool tip is stating that the Maximize Generation Value objective at priority 22 forced the Pool Elevation to the minimum set by goal 4.

These text strings that appear in the tool tips can be exported with the values using output DMIs described **HERE (DMI.pdf, Section 3.2.3)**.

8. Technical Appendix

8.1 Optimization Units and Scaling

Optimization computations are carried out in the same RiverWare standard units used for Simulation computations with one difference. As with Simulation, values are converted automatically from display units to internal units for computations, but for Optimization, that internal conversion might also include scaling. For example, storage volumes in standard units are often on the order of 10^9 m^3, whereas elevation changes might be on the order of 10^{-1} m. Introducing variables with such a wide range of magnitudes can lead to instability in an Optimization solution. Storage and elevation values are therefore scaled when added to the Optimization problem so that they will tend to be of similar orders of magnitude.

RiverWare determines how to scale values automatically based on their unit types, and all conversions are carried out internally. The user, therefore, does not typically need to be concerned with scaling. The only case in which it might be necessary to be aware of this scaling is if viewing the full LP problem generated for debugging purposes.Values in the LP text file will be the scaled values. Contact CADSWES (riverware-support@colorado.edu) with further questions about scaling.

8.2 Mixed Integer Programming

Some user-selectable engineering methods require a mixed integer programming solution. The Unit Power Table method, for example, requires integer programming to model the start-up and shut-down of individual turbines. Mixed integer programming adds substantial computational time to an Optimization model. **It is recommended that users contact CADSWES before attempting to use methods that require mixed integer programming.**

Inquiries can be sent to riverware-support@colorado.edu.

Mixed Integer Programs are optimization problems that contain both continuous and integer variables. In the river and reservoir modeling context, many of the variables are continuous. For example, outflow, pool elevation, and storage are all continuous variables. In contrast, hydropower turbines are operated such that a unit is either off or fully on. Thus, the Unit Power Table method has introduced some integer and binary (zero or one) variables. More specifically, the number of units generating at a given plant during a given time step is an integer variable and the decision to operate a single unit is a binary variable; the unit is either off or on. The modeling of integer and binary variables is sufficiently similar that the term “integer” is used to refer to both integer and binary variables.

Integer variables are also used to model discontinuities. For example, avoidance and cavitation zones are regions where power cannot be generated without damaging the equipment. These zones create a discontinuity in the power curve, and integer variables are used to model this discontinuity.

RiverWare solves this using Mixed Integer Linear Programs, but we will generally omit the word "linear". The "linear" restriction reflects the fact that all non-linear functions have been replaced by linear or piecewise linear functions. The use of linear approximation allows the optimization to solve faster. We will also omit the word "mixed" and refer to Mixed Integer Linear Programming as simply Integer Programming.

Integer Programs are more difficult to solve than linear programs. For this reason, RiverWare has limited use of integer variables and the heuristic solution methods used are discussed in more detail in the rest of this section.

8.2.1 Variables that bring Integer Programming into the problem

Two conditions are necessary for Integer Programming in RiverWare; integer programming is only allowed in the lowest priority policy and the policy must directly or indirectly use integer variables.

Integer Programming is limited to the lowest priority for performance reasons. To include integer variables at additional priorities would lead to what it called Integer Goal Programming (IGP). While there is no theoretical barrier, IGP is too computational challenging at this time. Consequently, RiverWare only allows integer variables in the final priority. If integer variables are referenced within a previous objective, RiverWare will relax the integrality constraints and solve the relaxed linear programming problem.

If a RiverWare policy doesn't reference integer variables then integer programming is not used. Currently the only source of integer variables is the Unit Power Table and related methods, **HERE (Section 5.5.2.17.5)**. Therefore, if only plant power methods are selected, integer variables will not be introduced and RiverWare will be solved as a Linear Goal Program.

The integer variables can be introduced either by directly referencing them in policy or by indirectly referencing them in policy. Integer variables are typically introduced indirectly by a user policy that optimizes the economic value of power using slots on the model's Thermal object. During optimization, these slot variables will be replaced with equivalent expressions involving Power and related quantities on the individual power reservoirs. In this way, if the Unit Power Table method is selected, constraints which model unit level power concerns will be introduced into the optimization problem.

In addition, the user might write policy constraints which refer to binary quantities. For example, the user might want to forbid scheduling "holes", i.e. turning a unit off for a single time period. One normally thinks of writing this constraint as

- Unit Is Generating [u,t] >= Unit Is Generating [u,t-1] + Unit Is Generating [u,t+1] - 1

However, to avoid referencing future time periods they will need to write this as:

- Unit Is Generating [u,t-1] >= Unit Is Generating [u,t-2] + Unit Is Generating [u,t] - 1

8.2.2 How it is solved

Understanding the method used to solve integer programs is useful for understanding the solution information and adjusting parameters that control the solution process.

Integer programs are solved by first starting with a "linear programming relaxation," for which integrality restrictions (i.e. that variables are either 0 or 1) are temporarily removed. From this relaxed solution two subproblems are created by branching on an integer variable. For example, the solver might branch on Unit 1 being used in the first period, with one branch requiring it to be used and the other branch requiring it not be used. This branching process can be continued with additional variables. The resulting process creates a tree structure that contains an optimal solution at one of the leaf nodes. Explicitly exploring the whole tree is impossible for all but the smallest problems. Fortunately, large portions of the tree can frequently be pruned off without exploration. The pruning is possible because the linear programming relaxation of a node in the tree may either be an integer solution, infeasible, or sufficiently suboptimal.

8.2.3 Timestep approach

The integer programs formulated in RiverWare are sufficiently hard that an optional, but highly recommended heuristic has been added to speed up the solution process. The heuristic is an iterative timestep approach that focuses on solving early timesteps with integer variables while temporarily relaxing the integrality conditions for later timesteps. By relaxing the integrality constraints for most of the timesteps, the problem can be solved much more quickly. After obtaining a solution, the integer variables for the earliest timesteps are frozen at their integer values, and some previously relaxed variables are treated as integer variables. The problem is resolved, and the process continues until all timesteps have been modeled as integer variables and frozen. For each iteration the integer variables are constrained both by the frozen integer variables for previous time steps and the linear programming relaxation for future timesteps.

There are two time windows that are advanced at each iteration in this process, the range of (non-frozen) integer variables in the problem, and the possibly smaller range of integer variables that should be frozen after the problem is solved. The size of each of these windows is an important parameter of this process. By default, one time step is used for each of these time windows, but the user can increase these values by setting appropriate parameters, **Mip Integer Window Size** and **Mip Freeze Window Size**. If both variables are set to the number of timesteps in the run, then the MIP will be solved as a single problem containing all of the integer variables. At this time, we do not recommend increasing the window sizes beyond 1 because of the dramatic increase in computation time.

This process may find the optimal solution, but it is not guaranteed. During testing, the timestep approach has found high quality, feasible solutions in a fraction of the time to find an optimal solution and prove it is optimal.

8.2.4 Defining Constraints

The details of the mathematical formulation of the mixed integer programming to model unit power is presented in the appendix **HERE (Section 8.2.7)**.

8.2.5 Additional parameters that can be used to control this solution

CPLEX provides a variety of parameters to tune the solution of optimization problems. RiverWare makes these parameters accessible to the user. In addition, RiverWare has added some additional parameters to the list that are related to RiverWare's use of goal programming and time windows. All of the parameters have default values determined by CPLEX or by CADSWES. In some cases CADSWES has replaced the default values of CPLEX parameters with values that have performed better for RiverWare models. These changes in the CPLEX parameters are in the cplex.par file packaged with RiverWare. Similarly, any non-default values for CADSWES parameters are contained in the goal.par file.

The vast majority of the parameters need never be examined, but some of them are potentially useful for integer programming. While branch and bound will theoretically find an optimal solution given enough time, the exponential growth of the search tree means that the time to do so is prohibitive for many practical problems. Instead, the tree can be used in a heuristic manner to find near optimal solutions. By limiting the number of solutions, the time allowed, and/or the optimality tolerance, the search can be terminated early with a good solution.

8.2.5.1 Setting Parameters

The parameters (both for CPLEX and RiverWare specific parameters) are set through the RiverWare interface. From the **Run Control** dialog, select **Optimization** as the controller and then select the menu option **View ➥ Optimization Run Parameters**. From the Optimization Run Parameters dialog, select **Set CPLEX** and **Goal Parameters**. The parameters are organized by categories. Select the desired parameter and change the current value to the desired value. Select either **Apply** (to save and continue with other parameter changes) or **Ok** (to save and exit the parameter dialog). If you check the "Save non-default settings to parameter file" box these settings will be saved to a file called cplex.init.par, and this file will be used to initialize parameter settings for future RiverWare sessions.

8.2.5.2 Lower Cutoff Parameter

This parameter usually will not be effective when the timestep approach (**HERE (Section 8.2.3)**) is used. However, when optimizing the entire problem, this parameter can be very influential because of the structure of the branch and bound solution method. For maximization problems, the lower cutoff parameter will help prune the search tree until an integer feasible solution is found (an upper cutoff parameter can be used for minimization problems). Ideally, this parameter will be set high enough to eliminate large portions of the tree without eliminating all of the feasible integer solutions. Since the optimal integer solution isn't known in advance, use of this parameter requires a little experimentation.

A starting point for the experimentation is the linear programming relaxation. By definition, integer solutions cannot improve on this value. For a maximization problem this means the optimal integer solution will be less than or equal to the linear programming relaxation and the lower cutoff parameter should certainly be set less than the objective function value for the linear programming relaxation. Our limited experimentation with test models suggests cutoff values ranging from about 90% of the linear programming relaxation to just below the linear programming relaxation.

Some experimentation is unavoidable with this parameter. If the lower cutoff value is set too high the search will end without a feasible solution. If the value is set too low (or there are too many time steps) the run will continue until some parameter stops the run or the user aborts the run. In both cases there won't be an integer solution. If the problem is small enough to be solved, a reasonable cutoff value can typically be determined with a couple of tries.

The objective function of the linear programming relaxation can be obtained approximately by aborting the run with a little delay after the last objective has started. The delay is necessary to give CPLEX time to solve the relaxation. After aborting, the objective function of the best node in the tree will be printed in the RiverWare diagnostics window. This may be slightly less than the linear programming relaxation. In general this value will decrease as branch and bound proceeds. The number of nodes processed in the tree are also printed in the diagnostic. Unfortunately, for technical reasons the value can't be printed while CPLEX is still running. The objective function is printed in "optimization units", which are typically scaled from user units for numerical stability. For this reason, direct interpretation of the value is not advised. However, these are the same units that should be used for the lower cutoff value.

The lower cutoff parameter is in the **MIP Tolerance Parameters** category.

8.2.5.3 Limiting the Number of Solutions

Sometimes, branch and bound will find a good or even optimal solution early in an optimization and spend the bulk of its time proving optimality or making relatively minor improvements in the solution. Limiting the number of solutions can improve the solution time. However, testing has shown that setting this parameter overly low while using the timestep approach can lead to integer infeasibility in the middle of the timestep iterations.

The solutions parameter is in the **MIP Limit Parameters** category.

8.2.5.4 Time Limit

Optimization problems can take a long time to solve. A time limit parameter allows a solver to exit gracefully when the solver takes too long. For integer programs, exiting when the time limit expires will not generate an error if a feasible solution has been found. Thus, a time limit parameter can be used to truncate exploration of the branch and bound tree.

The time limit parameter in the **General Parameters** category. The default value is set to 3000 seconds (50 minutes).

8.2.5.5 Optimality Tolerance

Optimality tolerance parameters define how close the solution must be to optimal to be considered close enough, either as a fraction of optimality (0.0 - 1.0) or as an absolute number (in optimization units). Even on relatively easy problems, proving 100% optimality (an optimality tolerance of zero) can require very large computation times. A small optimality tolerance (e.g., e-4 to e-6 absolute) generally yields a great improvement in runtime, and for certain problems larger tolerances (e.g., 1-10% fractional) can also be beneficial. Experimenting with these parameters may prove worthwhile.

The fractional and absolute optimality tolerance parameters are in the **MIP Tolerance Parameters** category and are called respectively, **mipgap** and **absmipgap**.

8.2.6 Unit Power

At the end of optimization, the optimized values are known but haven't been returned to the workspace yet. The challenge is to know what variables will be returned to drive the post-optimization RBS run. When using the Unit Power Table method, the Rulebased Simulation can be driven in several ways:

Optimization could return the individual Unit Turbine Releases and some other piece of information to cause the object to dispatch and solve. For example, the additional information could be the Outflow, plant Turbine Release or the "U" flag (implementation soon) on Turbine Release. In fact, it is recommended, to return the individual Unit Turbine Release and total Outflow. This approach calculates power very similarly to the optimization calculation and is perhaps the most natural way to do a post-optimization simulation. Also, given unit turbine releases will always produce an answer. Note, if a regulation method is selected, optimization will also need to return Unit Flow Reduction for Regulation and Unit Flow Addition for Regulation. This approach is more reliable than the next approach and will be the only one supported at this time.

Optimization could return the individual unit energy, the total Spill, and some other piece of information to cause the object to dispatch and solve. For example, the additional information could be the Plant Energy or the "U" flag (implementation soon) on Energy. This approach would be similar to option but would return unit energies instead of unit flows. In both of these options, given the individual unit values (and even spill) would not be enough to cause the reservoir to dispatch. A plant level value must be specified. Given unit energies, there is the potential to have an infeasible solution in simulation. An operating head and flow could be calculated that because of linearization errors, cannot meet the given energies. This approach is not supported or recommended.

For the time being, the first option is the only supported. The rule should return the plant level Outflow and the Unit Turbine Release, marking both with rule flags. The plant level value will cause dispatching in simulation. With these two approaches, the user has options on what should be returned from optimization and how the object should solve. Experimentation will allow the user to determine the variables that provides the best solution for the application.

8.2.7 Mathematical Formulation of MIP for Unit Power

This section presents an improved mathematical formulation of the relationship between power and other quantities. This formulation models power generation at the unit level to capture the fact that power generation occurs at one power reservoir with a heterogeneous collection of turbines. Integer variables allow the user to model the fact that these units have discrete states (off, generating power, spinning, etc.) and transitions between states are potentially costly.

The following table summarizes information concerning the slots used by the Unit Power Method, methods in the related dependent categories, and methods on the thermal object. The "Symbol" column indicates the symbolic notation that will be used in subsequent sections to refer to the slot.

The slot dimensionality column is included to give a rough idea of the resources required by these methods. This column uses the following abbreviations:

- T = The number of time steps.
- U = The number of units.
- P = Average number of points in the Unit Power Table for a single unit.
- H = The number of distinct head values in the Unit Power Table.
- TW = The number of distinct Tailwater values in the Unit Power Cavitation Table
- A = The number of distinct avoidance zones in the Avoidance Zone Table
- Q = The number of Turbine Release values in the Shared Penstock Loss Table
- B = The number of blocks in the Block Regulation slots

Symbol	**Slot Name**	**Type**	**# of Rows**	**# of Col.'s**	**Status**
Unit Power Method Slots					
	Number of Units	TableSlot	1	1	Input
	Unit Power Table	TableSlot	P	3*U	Input
	Unit Priority Table	TableSlot	U	1	Input
UIG	Unit Is Generating	AggSeriesSlot	T	U	Input / Output
UP	Unit Power	AggSeriesSlot	T	U	Input / Output
UEQT	Unit (Expected) Turbine Release	AggSeriesSlot	T	U	Input / Output
NUG	Number Units Generating	SeriesSlot	T	1	Input / Output
PP	(Plant) Power	SeriesSlot	T	3	Exists
PEQT	(Plant Expected) Turbine Release	SeriesSlot	T	3	Exists
PQS	(Plant) Spill	AggSeriesSlot	T	3	Exists
MinPPE	Minimum Power Elevation	TableSlot	U	1	Input
PE	Pool Elevation	AggSeriesSlot	T	2	Exists
MinPE	Min value of Pool Elevation Slot	Configuration			Exists
TW	Tailwater Elevation	AggSeriesSlot	T	2	Exists
Q	(Plant) Outflow	AggSeriesSlot	T	3	Exists
Startup Cost - Unit Lumped Cost method Slots					
USU	Unit Startup	AggSeriesSlot	T	U	Input / Output
USD	Unit Shutdown	AggSeriesSlot	T	U	Input / Output
NUSU	Number Units Startup	SeriesSlot	T	1	Input / Output
NUSD	Number Units Shutdown	SeriesSlot	T	1	Input / Output
	Unit Startup Cost Table	TableSlot	U	1	Input
USC	Unit Startup Cost	AggSeriesSlot	T	U	Output
PSC	Plant Startup Cost	SeriesSlot	T	1	Output

Symbol	Slot Name	Type	# of Rows	# of Col.'s	Status
Shared Penstock Head Loss method Slots					
	Shared Penstock Head Loss Table	TableSlot	Q	2	Input
	Shared Penstock Head Loss LP Param Table	TableSlot	3+	3	Input
SPH	Shared Penstock Head Loss	SeriesSlot	T	1	Output
Unit Head and Tailwater Based Regions method Slots					
	Unit Power Cavitation Table	TableSlot	~H*TW	4*U	Input
	Unit Cavitation Optimization Tolerances	TableSlot	U	2	Input
Unit Head Based Avoidance Zones method Slots					
	Unit Avoidance Zone Table	TableSlot	~H	3*U*A	Input
Unit Regulation method Slots					
	Unit Regulation Table	TableSlot	P	U*6	Computed
USMP	Unit Scheduled Mechanical Power	AggSeriesSlot	T	U	Output
USQT	Unit Scheduled Turbine Release	AggSeriesSlot	T	U	Output
URU	Unit Regulation Up	AggSeriesSlot	T	U	Input / Output
URD	Unit Regulation Down	AggSeriesSlot	T	U	Input / Output
UPRU	Unit Possible Regulation Up	AggSeriesSlot	T	U*P	Dynamic
UPRD	Unit Possible Regulation Down	AggSeriesSlot	T	U*P	Dynamic
UTSR	Unit Two Sided Regulation	AggSeriesSlot	T	U	Input / Output
UQAR	Unit Flow Addition For Regulation	AggSeriesSlot	T	U	Output
UQRR	Unit Flow Reduction For Regulation	AggSeriesSlot	T	U	Output
UOCPR	Unit Operating Cost Per Regulation	TableSlot	U	1	Input
UOC	Unit Operating Cost	AggSeriesSlot	T	U	Output
POC	Plant Operating Cost	AggSeriesSlot	T	U	Exists
PSMP	Plant Scheduled Mechanical Power	SeriesSlot	T	1	Output
PSQT	Plant Scheduled Turbine Release	SeriesSlot	T	1	Output
PRU	Plant Regulation Up	SeriesSlot	T	1	Input / Output
PRD	Plant Regulation Down	SeriesSlot	T	1	Input / Output
PR	(Plant Two-Sided) Regulation	SeriesSlot	T	3	Exists
PQAR	Plant Flow Addition For Regulation	SeriesSlot	T	1	Output
PQRR	Plant Flow Reduction For Regulation	SeriesSlot	T	1	Output

Symbol	Slot Name	Type	# of Rows	# of Col.'s	Status
Tailwater method specific slots affected by Unit Power Method					
TWBV	Tailwater Base Value	AggSeriesSlot	T	2	Exists
TWL	Temp Tailwater Lookup	SeriesSlot	T	1	Exists
SF	Stage Flow Tailwater Table	TableSlot		3	Exists
TWT	Tailwater Table	TableSlot		2	Exists
Thermal Object Slot					
SSC	System Startup Cost	MultiSlot	T	1	Input?/Output
Thermal Object Slots for one-sided regulation					
SRU	System Regulation Up	MultiSlot	T	1	Input?/Output
BRU	Block Regulation Up	AggSeriesSlot	T	B	Output
RUBC	Regulation Up Block Costs	AggSeriesSlot	T	B	Input
RUV	Regulation Up Value	SeriesSlot	T	1	Output
	Regulation Up Marginal Value	SeriesSlot	T	1	Output
	Regulation Up Previous Marginal Value	SeriesSlot	T	1	Output
SRD	System Regulation Down	MultiSlot	T	1	Input?/Output
BRD	Block Regulation Down	AggSeriesSlot	T	B	Output
RDBC	Regulation Down Block Costs	AggSeriesSlot	T	B	Input
RDV	Regulation Down Value	SeriesSlot	T	1	Output
	Regulation Down Marginal Value	SeriesSlot	T	1	Output
	Regulation Down Previous Marginal Value	SeriesSlot	T	1	Output

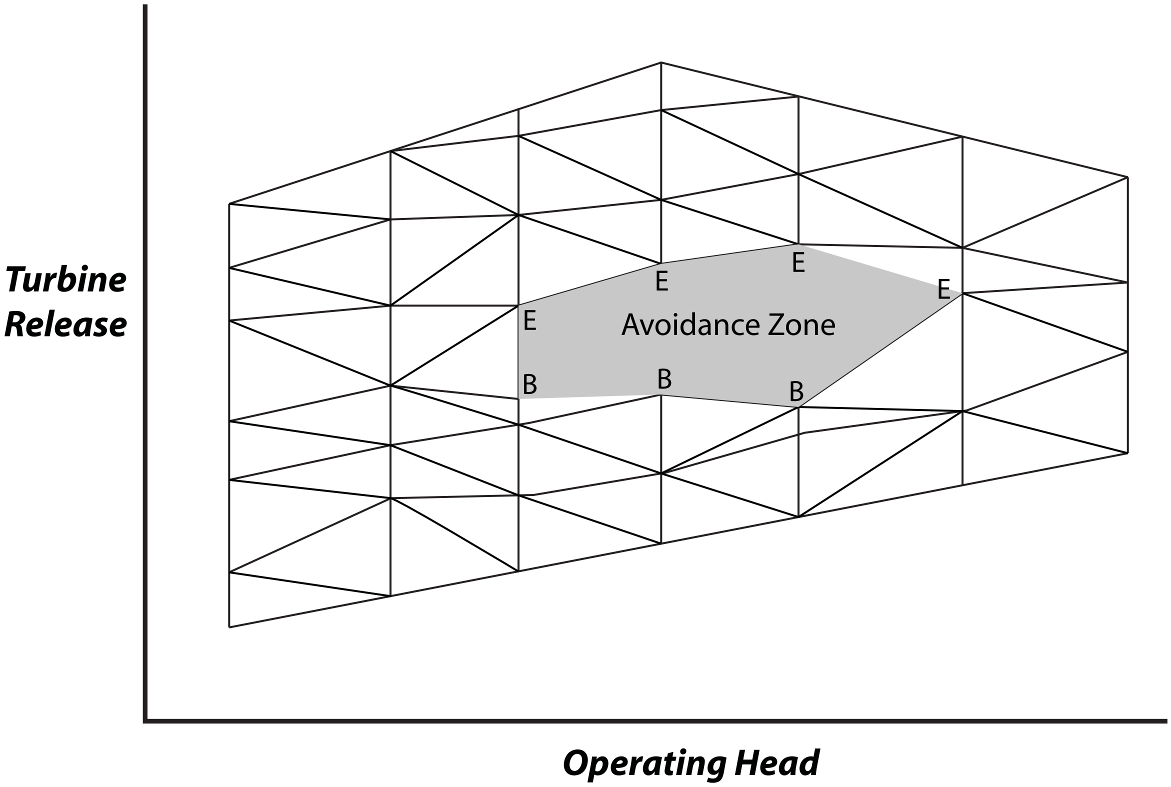
8.2.7.1 Triangulation

At the heart of the optimization controller's model of power is the triangulation algorithm for partitioning the power curve domain (specifically flow and head), which are described here. The method successfully excludes zones that are infeasible because of vibration, cavitation, etc.

The input for this method is a 3-dimensional power table and a table indicating zones that must be avoided. It is assumed that the power table contains the end points of head that define the zone and for each head the flow values that define the zone. The points themselves are assumed to be feasible operating points unless they represent minimum or maximum values of head or flow. There may be multiple separate avoidance zones.

An example of a triangulated power domain with an avoidance zone is in the following figure. In this figure, each vertex corresponds the data in a Unit Power Table row which apply to a single unit, head (operating head or net head), and Turbine Release. Power, an additional dimension represented in that table, is not shown.

Figure 1. Illustration of the triangulation algorithm applied to data from the Unit Power Table for a single unit.



Let us identify the avoidance zone values from the power table in the following way for the algorithm:

B - the beginning, low flow, of an avoidance zone for a given head

E - the end, high flow, of an avoidance zone for a given head. If either the maximal or minimal head has a single flow value, then that point is marked E.

8.2.7.1.1 Pseudo-code for triangulation

The basic idea is to iterate through the Head values in the (reduced) Unit Power Table, partitioning the vertical space between that head and the next into triangles. In general when deciding which of two potential triangles to create, RiverWare creates the one that has the lesser slope for the upper side.

```
// Triangulate between each pair of adjacent heads
For H1 = first head to next to last
    H2 = next H

    // Start with the minimum flow for each head
    Q1 = First Q for H1
    Q2 = First Q for H2

    // At each step increment one of the flows to form a triangle
```

```
While (next Q1 != NULL || next Q2 != Null)
        // If only Q1 can be incremented or incrementing Q1 leads to a flatter triangle top, increment Q1 and form
        // the triangle
        If (next Q2 = NULL || next Q1 - Q2 <= next Q2 - Q1)
                Create a Triangle with coordinates(H1.Q1, H1.next Q1, H2.Q2)
                Q1 = next Q1
        Else
                Create a Triangle with coordinates(H1.Q1, H2.next Q2, H2.Q2)
                Q2 = next Q2

        // If the two heads are at an avoidance zone, increment Q past it
        If (Q1 is marked B && Q2 is marked B or E)
                Q1 = next Q1
        If (Q2 is marked B && Q1 is marked B or E)
                Q2 = next Q2
```

If head is specified, then the triangles become line segments, and line segments become points.

8.2.7.2 Lambda Variables for Convex Combinations of Discrete Operating Points (triangles)

In the context of optimization, a lambda set is (roughly) a discretization of a continuous space, so RiverWare uses this technique to model the discretization of the power curve domain into triangles. While solvers don’t require it, usually it is assumed that lambda variables are constrained as follows:

0 <= Lambda_subscripts <= 1
Sigma (some subscripts, Lambda_subscripts) = 1

Lambda sets may additionally be constrained to be a specially ordered set of type 1 or 2, aka SOS1 or SOS2. The individual variables in an SOS are ordered and a reference row of weights is associated with each variable. An SOS1 may have at most one value that is nonzero while an SOS 2 may have at most two nonzero values and they must be adjacent. Solvers internally treat these sets as being similar to integer variables with branching to enforce the set conditions. The branching is based on weights in the reference row. The general idea is to calculate a weighted average of the variables using the reference row. The two branches are roughly that either variables on one side of the weighted average are zero or the variables on the other side are zero.

In addition, frequently the individual variables belonging to an SOS1are also binary variables. The variables in a SOS2 are almost always continuous.

Power is a nonlinear function of both flow and head. Hence, when it is discretized, the domain of the function is two-dimensional. The SOS technique can be applied here as well. In this case, a combination of 3 points is used to define flow and head. Alexander Martin has described how to model this case as an SOS type k, where k is 3 in this case (A. Martin, “Approximation of Non-linear functions in Mixed Integer Programming” Workshop on Integer Programming and Continuous Optimization, Chemnitz University of Technology, November 7-9, 2004, www-user.tu-chemnitz.de/~helmberg/workshop04/martin.ppt). Martin then modifies a solver to directly branch on the SOS k, creating separate branches for each region of the domain. Note: some solvers have an SOS 3, but this typically does not refer to the same thing as it is used here.

A variant of his method that is compatible with CPLEX and other solvers is used here. The flow and head domains are partitioned into triangles and define a variable for each triangle, Lambda_1. The triangles in an SOS 3 are analogous to the line segments in an SOS 2. The triangulation is described in section 2.7.

The following lambda variables are part of the formulation:

- Lambda_1rut: selection of triangular region r on unit u

This variable binary, technically an SOS k, Lambda_2 is used to implement this functionality. Therefore, there is no need for a reference row and its weights. This variable may represent 2 points instead of 3 if a feasible region is one-dimensional. For example most data sets will contain points with zero flow for several heads, but disallow small flows. Thus, the segment of zero flow between adjacent heads will be a Lambda_1 variable even though it is not a triangle. In theory, there could be a single isolated operating point that would need its own Lambda_1 variable, but this is unlikely in practice.

- Lambda_2h'ut: selection of a triangle with approximate head h' for unit u.

  This is a binary variable, SOS1 for each t: reference row weights are the approximate head (half way between the two head values for the points in the triangle). The set of Lambda_1rut triangles with r in h' form an SOS1 for each h',u,t combination. The reference row weights are the average q value of the points in the triangle. In addition the sum is constrained to equal Lambda_2h'ut:
- Lambda_4hut: selection of head h

  This is a continuous variable, SOS2 for each u,t: the reference row weights are head.
- Lambda_6hqut: combination of o and q

  This is a continuous variable, not an SOS. The data should include q=0 for all possible heads, even those that cannot generate power.

  The set of Lambda_6hqut points with the same o form an SOS2 for each h,u,t combination. The reference row weights are the q values.

8.2.7.3 Defining constraints

This section presents the constraints which will be introduced into the optimization problem as power-related quantities are encountered in the optimization policy (when the Unit Power method is selected).

8.2.7.3.1 Notes on variables and notation

All variables are constrained to be positive unless otherwise noted.

8.2.7.3.2 Binary Variables

These variables are required to be zero or one, and will be enforced on user inputs in the corresponding slots at the beginning of the run.

- Unit Is Generating (UIGut)
- If the Startup method is selected
  - Unit Startup (USUut)
  - Unit Shutdown (USDut)
- Lambda 6 variables (Lambda_6put)

8.2.7.3.3 Continuous Variables

- If the Shared Penstock Head Loss method is selected,
  - PNHt = Net head after losses in the shared penstock
  - SPHt = Shared penstock losses
- If the Unit Regulation method is selected,
  - RUFhqut = Fraction of possible regulation up at point indexed by hqu being used
  - RDFhqut = Fraction of possible regulation down at point indexed by hqu being used

8.2.7.3.4 Symbols

The following symbols will be used in the formulation below (see also slot summary table):

- Xp = The X component of point p in a data table (i.e., the value in the column labled X)
- WQru = turbine flow weighting of triangle r on unit u. Equal to the average of flow on the points r contains.
- WHru = head weighting of unit triangle r on unit u. Equal to the average of head on the points r contains.
- TWMpu = minimum tailwater that allows point Lambda_6put to be feasible
- MinTW = minimum possible tailwater at the plant
- TW = plant tailwater elevation
- QTr = maximum turbine release for triangle r

The following indices are used:

h' = approximate head index for power triangles with their center of mass within a tolerance of the operating
r = triangle region of the power curve where convex combinations are allowed.
u = unit index
t = time
p = point in a table. E.g., for the Unit Power table each point is a triplet (QT, H, P), so QTp denotes the Flow component of p, Hp denotes the head component, and Pp denotes the Power component. Similarly, the Unit Regulation Table has sextets of (QT,H, RUL,QTUL,RDL, QDL). RULp denotes the Regulation Up Limit, the maximum power generation that can be attained by increasing flow at the specfied operating head without pasing through an avoidance or cavitation zone. Similarly, RDLp is the Regulation Down Limit, QTUL and QTDL are the flow values associated with these power values.

Sigma indicates capital sigma used for a summation. The arguments following sigma in parenthesis are respectively the indices to sum over and the expression summed.

Bold typeface for a variable indicates the constraint is a defining constraint for that variable.

The constraints are presented with an unspecified timestep (t) and unit (u). Instantiations will be created during the run only for those units and timesteps which are referenced by the user's policy.

8.2.7.3.5 Unit Level Constraints

1. **Unit Is Generating is equivalent to selecting a triangular region above zero flow/generation**

> "Unit Generation Constraint"
> Sigma (r with QTru > 0, Lambda_1rut) = **UIGut**

2. **Unit Start up and Shut down**

If the Unit Lumped Cost startup method is selected,

> "Unit Startup Constraint"
> UIGut – UIGut - 1 = **USTu,t** – **USHu,t**

3. **Operating Head or Net Head is discretized by Lambda_4**

If there is no addtional head loss (No Method in the Head Loss category) is selected

> "Operating Head Unit Lambda Constaint"
> POHt = Sigma (head h in Unit Power Table with unit u, Hhu * **Lambda_4hut**)

Else If the Shared Penstock Head Loss method is selected

> "Net Head Unit Lambda Constaint"
> PNHt = Sigma (head h in Unit Power Table with unit u, Hhu * **Lambda_4hut**)

4. **Net Head Definition**

If the Shared Penstock Head Loss method is selected:

> "Net Head Constraint"
> **PNHt** = OHt - SPHt (Net Head = Operating Head - Shared Penstock Loss)

5. **Specified Power – Flow curve from Lambda_6 points**

If the Unit Frequency Regulation method is selected:

> "Unit Scheduled Mechanical Power Constraint"
> **USMPut** = Sigma (p in Unit Power Table with unit u, Ppu * Lambda_6put)

Else

> "Unit Power Constraint"
> **UPut** = Sigma (p in Unit Power Table with unit u, Ppu * Lambda_6put)

If the Unit Frequency Regulation method is selected:

> "Unit Scheduled Turbine Release Constraint"
> **USQTut** = Sigma (p in Unit Power Table with unit u, QTpu * **Lambda_6put**)

Else

> "Unit Expected Turbine Release Constraint"
> **UEQTut** = Sigma (p in Unit Power Table with unit u, QTpu * **Lambda_6put**)

6. Tailwater Limit to Prevent Cavitation

"Tailwater Cavitation Constraint"
TWt - MinTW >= Sigma (p in Unit Power Table with unit u, (TWMpu - MinTW) * **Lambda_6put**)

Note: TWMpu - MinTW should be 0 for most power points

7. Minimum Power Elevation

if MinPPEu != NaN

> "Minimum Power Elevation Constraint"
> PEt - MinPE >= (MinPPEu - MinPE) **UIGut**

8. Power – Flow regulation up equals a fraction of the limit

If the Unit Frequency Regulation method is selected,

> "Unit Regulation Up Constraint"
> **URUut** = Sigma (p in Unit Regulation Table with unit u, RULpu * UPRUut)
> "Unit Flow Addition for Regulation Constraint
> **UQARut** = Sigma (p in Unit Regulation Table with unit u, QTULpu * **UPRUut**)
> "Unit Regulation Up Lambda_6 Constraint"
> For all unit u, for all p in Unit Power Table
>
> > **UPRUut** <= Lambda_6pt

9. Power – Flow regulation down equals a fraction of the limit

If the Unit Frequency Regulation method is selected,

> "Unit Regulation Down Constraint"
> **URDut** = Sigma (p in Unit Regulation Table with unit u, RDLpu * UPRDut)
> "Unit Flow Reduction for Regulation Constraint
> **UQRRut** = Sigma (p in Unit Regulation Table with unit u, QDLpu * **UPRDut**)
> "Unit Regulation Down Lambda_6 Constraint"
> For all unit u, for all p in Unit Regulation Table
>
> > **UPRDut** <= Lambda_6put

10. Two-sided regulation tied to regulation up and down

If the Unit Frequency Regulation method is selected,

> "Two-sided Regulation Limited by Regulation Up"
> **UTSRut** <= URUut

“Two-sided Regulation Limited by Regulation Down”
UTSRut <= URDut

11. Expected power and flow after regulating

If the Unit Frequency Regulation method is selected,

“Expected Unit Power Constraint”
UPut = USMPut + URUut/2 - URDut/2

12. Unit Operating Costs

If the Unit Frequency Regulation method is selected,

“Unit Operating Cost Constraint”
UOCut = timestepHRS * UOCPR * (RUut + RDut)

13. Unit Startup Costs

Unit Startup Cost is replaced using the following definition:

USCut = Unit Startup Cost Table u * USTut

14. Unit Priority Constraints

Ordering the units will considerably reduce the search space, particularly if the some of the units are identical. In addition, this is often an operating policy.

For unit i with a higher priority than unit j:

“Precedence Constraint”
UIGit > **UIGjt**

8.2.7.3.6 Defining constraints for lambda variables

1. Lambda1 Unity Constraint

Sigma(r with unit u, **Lambda_1rut**) = 1

2. Lambda6 Unity Constraint

Sigma(p in Unit Power Table with unit u, **Lambda_6put**) = 1

Note that the Lambda_2 unity constraint is omitted because it is implied by Lamba_1, and the Lambda_4 unity constraint because it is implied by Lambda_6.

3. Lamba1 Definition Constraint

For all h’:
Sigma (r in h’, **Lambda_1rut**) = Lambda_2h’t

4. Lambda 2 Definition Constraint

For each Lambda_4 variable from the triangles, the sum of the adjacent Lambda_4 variables must be greater than the Lambda_2 variable:

For all h':
Sigma (h adjacent to h', Lambda_4hut) >= **Lambda_2h'ut**

5. **Triangle Definition Constraint**

The sum of the use of the points of a triangle must be greater than the triangle variable:

For all r:
Sigma (p in r, **Lambda_6put**) >= Lambda_1rut

6. **Lambda 6 Definition Constraint - the sum is constrained to equal Lambda_4hut**

For all h:
Sigma (p with Head value = h, **Lambda_6put**) = Lambda_4hut

8.2.7.3.7 Unit/plant relationship constraints

Any plant variable, PVx, where x represents the appropriate subscripts is defined by the sum of the unit variables, Vux, indexed by unit u and x. Note that plant variables are only created if they are used directly or indirectly by policy. Thus, the defining constraint is:

"Plant Unit V Constraint"
PVx = Sigma(u,Vux)

8.2.7.3.8 Redefining existing plant level linearizations

The modeling of power will be more accurate when lambda variables and SOS sets are used to replace the plant level linearization of pool elevation and tailwater. Both of these affect operating head. This has been temporarily postponed until a potential problem is addressed: the associated lambda variables might be frozen by an objective solved using linear programming before it reaches the power objective. If this were to occur, the intended model for integer programming would be disrupted. This is more likely to be a problem for pool elevation than tailwater because pool elevation is important for non-power policy, whereas tailwater is usually only important for power. Moreover, the error in existing methods is larger for tailwater than pool elevation. For these reasons, it is more likely beneficial to change the modeling of tailwater.

The pool elevation constraints are:

"Pool Elevation Lambda Constraint"
PEt = Sigma(p in Elevation Volume Table, PEp * Lambda_7st)

"Pool Elevation Lambda Storage Constraint"
St = Sigma(p in Elevation Volume Table, SEp * **Lambda_7**st)

"Pool Elevation Lambda Unity Constraint"
Sigma(p in Elevation Volume Table, **Lambda_7**pt) = 1

Lambda_7 is an SOS2

If the tailwater method is optTWStageFlowLookupTable, the tailwater constraints define tailwater, outflow, and tailwater base value in terms of Lambda_8 points directly or indirectly in terms of Lambda_9 and Lambda_10:

“Tailwater Stage Flow Lambda Constraint”
TWt = Sigma(p in Stage Flow Tailwater Table, TWp * Lambda_8pt)

“Tailwater Stage Flow Lambda Flow Constraint”
Qt = Sigma(flow q in Stage Flow Tailwater Table, Qq * **Lambda_10**qt)

“Tailwater Stage Flow Lambda Base Value Constraint”
TWBVt = Sigma(tailwater base value v in Stage Flow Tailwater Table, TWBVv * **Lambda_9vt**)

“Tailwater Stage Flow Lambda Unity Constraint”
Sigma(p in Stage Flow Tailwater Table, **Lambda_8pt**) = 1

“Tailwater Stage Flow Lambda Base Value Lambda Constraint”
For all v:
Sigma (p in Stage Flow Tailwater Table with tailwater base value = v, **Lambda_8pt**) = Lambda_9vt

“Tailwater Stage Flow Lambda Flow Lambda Constraint”
For all q:
Sigma (p in Stage Flow Tailwater Table with outflow = q, **Lambda_8pt**) = Lambda_10qt

Lambda8 is an SOSk implemented by making Lambda_9 and lambda_10 SOS2s. An alternative implementation would be triangles as used for power.

If the tailwater method is optTWBaseValuePlusLookupTable, the new tailwater constraints define temp tailwater lookup and outflow in terms of Lambda_11:

“Tailwater Lookup Lambda Constraint”
TWLt = Sigma(f, TWTf * Lambda_11ft)

“Tailwater Lookup Lambda Flow Constraint”
Qt = Sigma(f, TWQf * **Lambda_11**ft)

“Tailwater Lookup Unity Constraint”
Sigma(f, **Lambda_11**ft) = 1

Lambda_11 is an SOS2

For this method, the existing tailwater constraint is retained:

TWt = (TWBVt + TWBVt-1) / 2 + TWLt

8.2.7.3.9 Thermal Object

The thermal object has a number of multi-slots that are potential variables in an optimization problem. These multi-slots typically represent the system wide total of individual reservoir values that are linked to these slots. Equations for these sums are added automatically if a policy either directly or indirectly references one of the thermal object multi-slots.

Many of the existing variables and definitions are already on the thermal object. In addition, to the existing equations, the following new equations will be added.

System Regulation Up is greater than the sum of Block Regulation Up variables:

SRUt * timestepHRS >= Sigma(b, **BRUbt**)

The Regulation Up Value equals the sum of the value of the Block Regulation Up variables:

RUV = Sigma(b, RUBCb * BRUbt)

System Regulation Down is greater than the sum of Block Regulation Down variables:

SRDt * timestepHRS >= Sigma(b, **BRDbt**)

The Regulation Down Value equals the sum of the value of the Block Regulation Down variables:

RDV = Sigma(b, RDBCb * BRDbt)