Tutorial 6: Operating Rules

The purpose of this tutorial is to practice using Operating Rule Language (ORL) statements to set gate operations and flows. With operating rules, expressions can be crafted to make steer the model on-the-fly; e.g., a gate can be directed to automatically close when stage conditions reach a certain threshold. In this tutorial we will create operating rules to operate a gate and to regulate a source/sink inflow.

1. Adding a Second Gate Where Op Rule Will Be Applied

a. Open the *tutorial_grid.inp* and file.

b. In the *Gates* table:

   1) Copy the row with *gate_1,* paste it and change the following fields:

      i) Name: *gate_2*

      ii) Name/No.: *5*

      iii) to Node: *5*

e. In the *Gate Devices table*:

   1) Copy the data from gate_1, change the gate name to gate_2 and change the following fields:

      i) Elev: *-2*

f. Save the current settings.

2. Adding Output for the Second Gate:

b. Open the *output_hydro_tutorial* layer.

c. Create the OUTPUT_GATE table:

```
OUTPUT_GATE
NAME    GATE_NAME  DEVICE VARIABLE  INTERVAL PERIOD_OP  FILE
END
```

d. In the output table enter the following values into the appropriate fields:

   i) Output Name: *gate_2_weirop*

   ii) Gate name: *gate_2*

   iii) Device: *weir*

   iv) Variable: *op-from-node*

v) Time Interval: *15min*

vi) Period Op: *inst*

vii) File: *${HYDROOUTDSSFILE}*

e. Save the current settings.

f. Add the following channel outputs in a new OUTPUT_CHANNEL table:

```
OUTPUT_CHANNEL
NAME     CHAN_NO DISTANCE VARIABLE INTERVAL PERIOD_OP FILE
trigger_loc    4     7500 stage    15min    inst    {HYDROOUTDSSFILE}
ds_gate2       5        0 flow     15min    inst    {HYDROOUTDSSFILE}
END
```

Note: You don't have to use a new OUTPUT_CHANNEL block. You can have as many or as few OUTPUT_CHANNEL blocks as you want in one input file and it will not affect the model. Note, however, that you cannot redefine an entry in one file. In the present case the identifier is the combination of NAME and VARIABLE, so you cannot redefine the output named trigger_loc, variable stage in this file (layering/overriding would be ambiguous).

2. Create an Operating Rule to Close the Weir when Stage is Low:

Now we are ready to write the first operating rule. This rule closes the new gate we created during times where stage at a monitoring point is low. First we will define the rule in terms of an expression called *stage_critical* (the condition where stage violates a minimum) and *op_applies* (a seasonal condition that is True when we are controlling the gate for stage. In a later step we will define these variables.

a. Create a file called *oprule_tutorial.inp*.

b. Create the *Operating Rules* table:

```
OPERATING_RULE
NAME        ACTION        TRIGGER
END
```

1) Enter the following values into the appropriate fields:

i) Name: *weir_close*

ii) Action Definition: *SET gate_op(gate=gate_2, device=weir, direction=from_node) TO CLOSE RAMP 30MIN*

iii) Trigger Definition: *stage_critical AND op_applies*

iv) Use: Make sure that the entry contains a checkmark.

c. Create an OPERATION include block in hydro.inp and add the new file so that it will be used by DSM2-HYDRO.

```
OPERATION
oprule_tutorial.inp
END
```

d. Save the current settings.

Note that the expressions stage_critical and op_applies will be created in a later step.

3. Create an Operating Rule to Open the Weir when Stage is High:

As before, we will enter the rule to open the weir first in terms of the expressions *stage_relax* (a condition where stage is safely above a threshold where we can open the gate) and *op_applies*. In the next step we will define these expressions.

a. In the *Operating Rules table* enter the following values into the appropriate fields:
   i) Name: *weir_open*
   ii) Action Definition: *SET gate_op(gate=gate_2, device=weir, direction=from_node) TO OPEN RAMP 30MIN*
   iii) Trigger Definition: (stage_relax AND op_applies) OR NOT(op_applies)

b. Save the current settings.

c. In the *hydro.inp* file, add the following environmental variables and values into the ENVVAR section:

```
STAGE_CRITICAL  1.4
STAGE_RELAX  1.6
```

4. Define Expressions used in the rule

a. In the same file, create the OPRULE_EXPRESSION table:

```
OPRULE_EXPRESSION
NAME            DEFINITION
END
```

1) Enter the following values into the appropriate fields:
   i) Name: *op_applies*
   ii) Definition: *SEASON < 01FEB*

2) Enter the following values into the appropriate fields:
   i) Name: *stage_critical*
   ii) Definition: *chan_stage(channel=4, dist=7500) < ${STAGE_CRITICAL}*
3) Enter the following values into the appropriate fields:
   i) Name: *stage_relax*
   ii) Definition: *chan_stage(channel=4, dist=7500) > ${STAGE_RELAX}*

b. Save the current settings.

d. Now run HYDRO and QUAL:
   1) Open a command window for the *t6_oprule* directory.
   2) In the command window, type: *hydro hydro.inp*.
   3) In the command window, type: *qual qual.inp*.
   4) Open the *output.dss* file in the *t6_oprule* directory, and examine the results.

5. Add a Reduced Flow Operating Rule:

In our next operating rule, we will control the inflow to a node by having it toggle back and forth between a larger “full flow” and a “reduced flow”. First we will enter the rule and then we will define the full and reduced flows.

a. In the *Operating Rules table* enter the following values into the appropriate fields:
   i) Name: *flow_reduce*
   ii) Action Definiton: *SET ext_flow(name=source1) TO ifelse(stage_critical,reduced_flow,full_flow)*
   iii) Trigger Definition: *TRUE*

b. Now create the expressions that define *full_flow* and *reduced_flow*. In the *Oprule Expressions* table:
   1) Enter the following values into the appropriate fields that define *full_flow.* This will involve the time series *source_flow* which we will enter later:
      i) Input Name: *full_flow*
      ii) Definition: *ts(name=source_flow) [note: this is a reference to a time series we haven’t defined yet].*
   2) Do the same for *reduced_flow.* Note: we are defining *reduced_*flow in terms of the time series. There is no guarantee of what order expressions will be

evaluated, so you cannot safely define *reduced_flow* in terms of another expression such as *full_flow*. Enter the following values into the appropriate fields:

i) Input Name: *reduced_flow*

ii) Definition: *0.5*ts(name=source_flow).*

c. Save the current settings.

1) Now we will define the *source_flow* time series upon which the *full_flow* and *reduced_flow* expressions are based.

d. Create the *Operation Time Series* table:

```
OPRULE_TIME_SERIES
NAME    FILLIN        FILE          PATH
```

1) Enter the following values into the appropriate fields:

i) Input Name: *source_flow*

ii) Input File: *${TUTORIALINPUT}*

iii) Path: */TUTORIAL/SOURCE/FLOW//15MIN/CONSTANT/ [Note: there are two forward slashes between FLOW* and *15MIN]*

iv) Fillin: *none*

e. Save the current settings.

6. Override the Expression op_applies:

Recall that *op_applies* is used to determine when the weir is operated. Previously the definition of this expression was seasonal: the expression was SEASON < 01FEB. The goal now is to make the same expression depend on a time series. Rather than change the expression, we will override it in a new layer.

a. Add a new Operating Rules Layer:

1) Create a file called *oprule_ tutorial_ revision.inp*

b. Redefine the expressions that define *op_applies*. In the *Expressions* table:

1) Create the OPRULE_EXPRESSIONS table.

2) Enter the following values into the appropriate fields:

i) Input Name: *op_applies*

ii) Definition: *ts(name=op_used) [note: this is a reference to a time series we will define in the next step].*

c. Define the time series *op_used* on which the *op_applies* expression depends. In the *Operation Time Series* table:
   1) Right-click and select *Insert row*.
   2) Enter the following values into the appropriate fields:
      i) Input Name: *op_used*
      ii) Input File: *${TUTORIALINPUT}*
      iii) Path: */TUTORIAL/GATE/FLAP_OP//IR-YEAR/CONSTANT/*
      iv) Fillin: *none*
d. Add *oprule_tutorial_revision.inp* after *oprule_tutorial.inp* in the OPERATIONS block of hydro.inp so that it will be used by HYDRO.
e. Run HYDRO and QUAL and examine the results.