# RTC-Tools

## Software Tools for Modeling Real-Time Control

## Reference Manual

Dirk Schwanenberg, Bernhard Becker

Version: 1.2.000000
Revision: 386

22 July 2014

**RTC-Tools, Reference Manual**

**Published and printed by:**
Deltares
Boussinesqweg 1
2629 HV Delft
P.O. Box 177
2600 MH Delft
The Netherlands

telephone: +31 88 335 82 73
fax: +31 88 335 85 82
e-mail: info@deltares.nl
www: http://www.deltares.nl

**Contact:**
Bernhard Becker
telephone: +31 88 335 8507
fax: +31 88 335 8582

Dirk Schwanenberg
telephone: +31 88 335 8447
fax: +31 88 335 8582

e-mail: rtc-tools@deltares.nl
www: http://oss.deltares.nl/web/rtc-tools

# Contents

# 1 Introduction

## 1.1 Management of water systems

Water management means to operate a water system in such a way that it meets one or multiple objectives, for example

- ◇ drinking water supply,
- ◇ navigation,
- ◇ irrigation water supply,
- ◇ environmental flow,
- ◇ hydroelectricity production,
- ◇ flood protection.

Stakeholders implement the management by taking control of hydraulic structures such as weirs, pumps, hydro turbines or water intakes. In this context, real-time control (RTC) refers to the automation of hydraulic structures in water systems whereas decision support suggests a control and leaves it over to the operating staff to implement it or not.

RTC-Tools (Real-Time Control Tools) is an open source, modular toolbox dedicated to the simulation of real-time control and decision support of hydraulic structures. It can be used

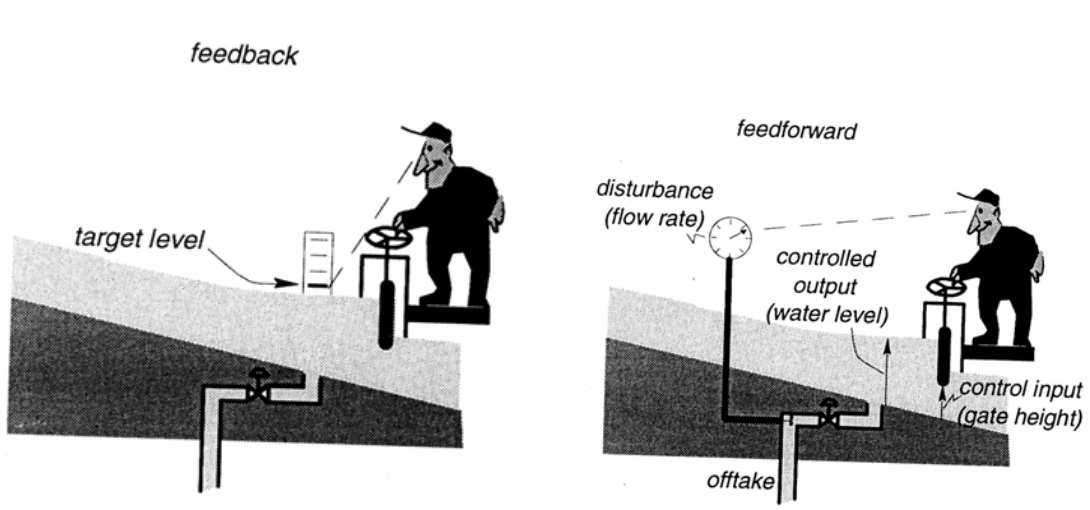
- ◇ standalone or in combination with hydraulic models for general modelling studies,
- ◇ as decision support component in operational forecasting and decision-support systems for example for drought management, water allocation, flood mitigation or the dispatch of hydro power assets,
- ◇ as a real-time control component in SCADA systems (supervisory control and data acquisition systems) in which RTC-Tools implements feedback control and advanced Model Predictive Control (MPC) for implementing state-of-the-art control strategies aiming at a safe, energy and cost aware, integral management of water resources systems.

## 1.2 Control methods

### 1.2.1 Feedback and feedforward

Feedback is a control principle where the control error (i. e. the difference between target value and actual value) is measured and used to determine control actions. Feedforward uses observed disturbances from somewhere else in the system to determine control actions (Schuurmans, 1997).

Figure 1.1 shows an example of these techniques. In both cases, the operator aims to maintain the water level at target by adjusting the downstream gate position. In Figure 1.1b the operator controls the water level using information on the offtake-structure flow. The offtake-structure flow cannot be controlled by the operator and can therefore be considered a disturbance. When measurements of disturbances are used to form control decisions the control method is called feedforward. In Figure 1.1a the operator checks the deviation of the water level from the target level. The water level is the controlled parameter. This operation technique is called feedback control (Schuurmans, 1997).




***(a)*** *feedback control* ***(b)*** *feedforward control*

***Figure 1.1:*** *Feedback control and feedforward control (Schuurmans, 1997)*

### 1.2.2 Model Predictive Control

While feedback control means to operate a water system based on the current state of the system, model predictive control (MPC) predicts future state trajectories, e.g. for anticipating on approaching flood events within current control actions. This requires internal modelling of the controlled water system for assessing and optimizing the impact of control actions on the system and its control performance.

While in Figure 1.1 the operaters base their decision on the current system state, the operator in Figure 1.2 takes also the future behaviour of the water system into account.

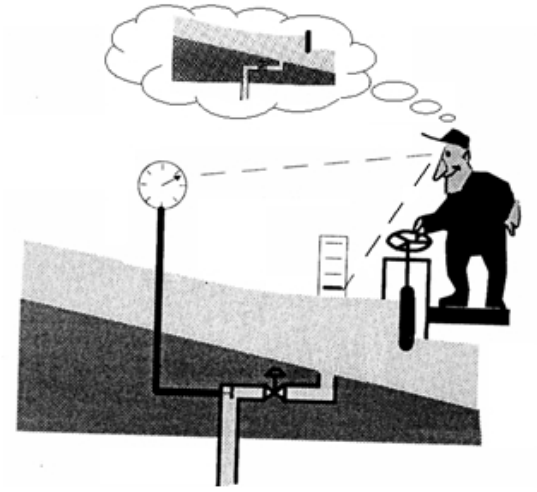


***Figure 1.2:*** *Model predictive control (Schuurmans, 1997)*

## 1.3 Fields of application

The RTC-Tools package aims at the simulation of various real-time control and decision support techniques in application to water resources systems. It includes feedback control strategies with triggers, operating rules and controllers as well as advanced Model Predictive Control (MPC) setups based on a combination of forecasting and optimization.

The MPC-based predictive control strategies require internal modeling of the controlled water system in the optimization procedure. Therefore, the package includes a number of simple hydrological and hydraulic models as well as various other simulation components for setting up water resources models. The model implementation reflects the requirements of the optimization procedure by supplying both a simulation mode and an adjoint mode for computing first-order model derivatives. Having in mind its application in operational forecasting systems, the software pays special attention to state handling. This includes by definition the system states of all existing components such as triggers, controllers, operating rules and modeling components.

RTC-Tools is designed for stand alone use or serves as a building block in a larger system architecture. We pay special attention to various interfaces for integration it into frameworks such as Delft-FEWS, Matlab or OpenDA. Furthermore, the OpenMI interface enables its online coupling to a wide range of hydraulic modeling packages.

The following examples present three typical applications of RTC-Tools:

- Stand-alone use as a forecasting model in Delft-FEWS:
  Assume we require a forecasting model for a medium-sized river basin including an HBV-style conceptual rainfall runoff model, a simple hydraulic routing components and the integration of several controlled flood detention polders. Although many other models supply the required features, the application of RTC-Tools can be advantageous, because it enables the complete representation of the system in a single model (Figure 1.3) and integrates seamless into Delft-FEWS with a minimum configuration effort and a maximum of interaction.
- Integrated application with sophisticated hydraulic model via OpenMI (Gregersen *et al.*, 2007):
  Typical hydraulic models such as SOBEK, Mike11 or HEC-RAS have on-board features for modeling the real-time control of hydraulic structures. If more advanced features beyond the available ones are required, APIs may enable the user to link external code (Figure 1.4). A main advantage of RTC-Tools against a dedicated user-programming is the availability of a wide range of already existing and tested features, the option for extending or modifying them easily, and the overall framework with file io and interfaces.
- Predictive control of hydraulic structures:
  In particular in forecasting system, MPC provides an advanced option for supervisory control and decision-making for example for scheduling pump actions in polder systems or the release of reservoirs. The set-up consisting of an optimization of the hydraulic structure by MPC included the embedded representation of the water resources system (Figure 1.5).

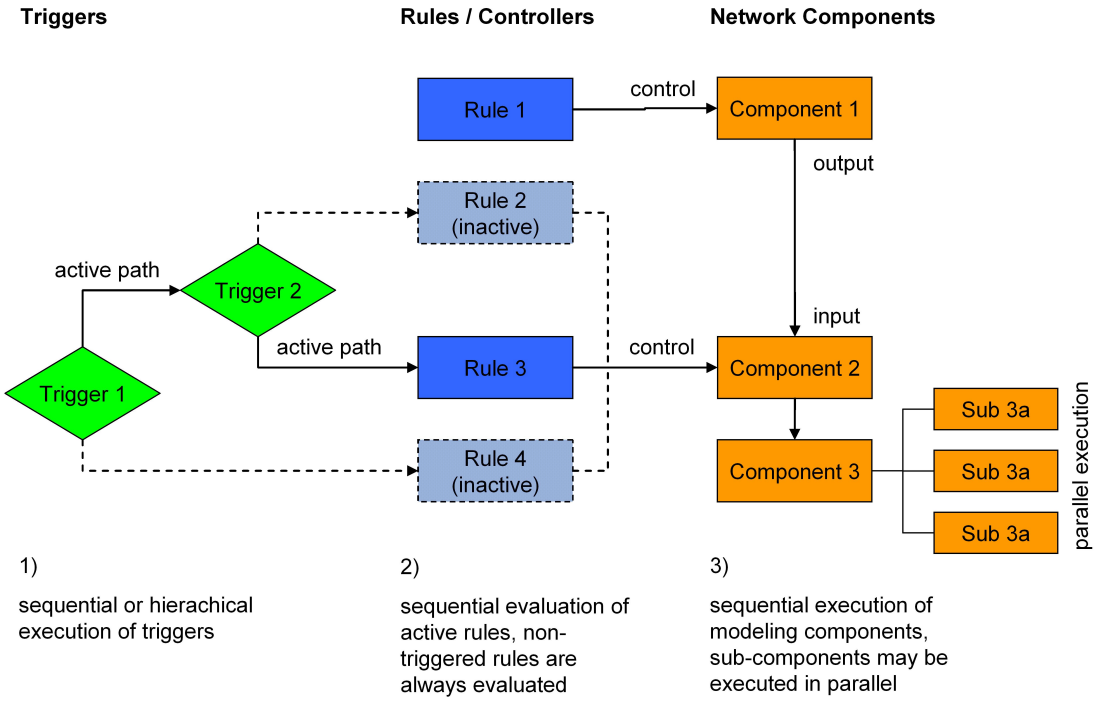



***Figure 1.3:*** *RTC-Tools configuration in typical simulation set-up: triggers, reactive operating rules and controllers, network components*

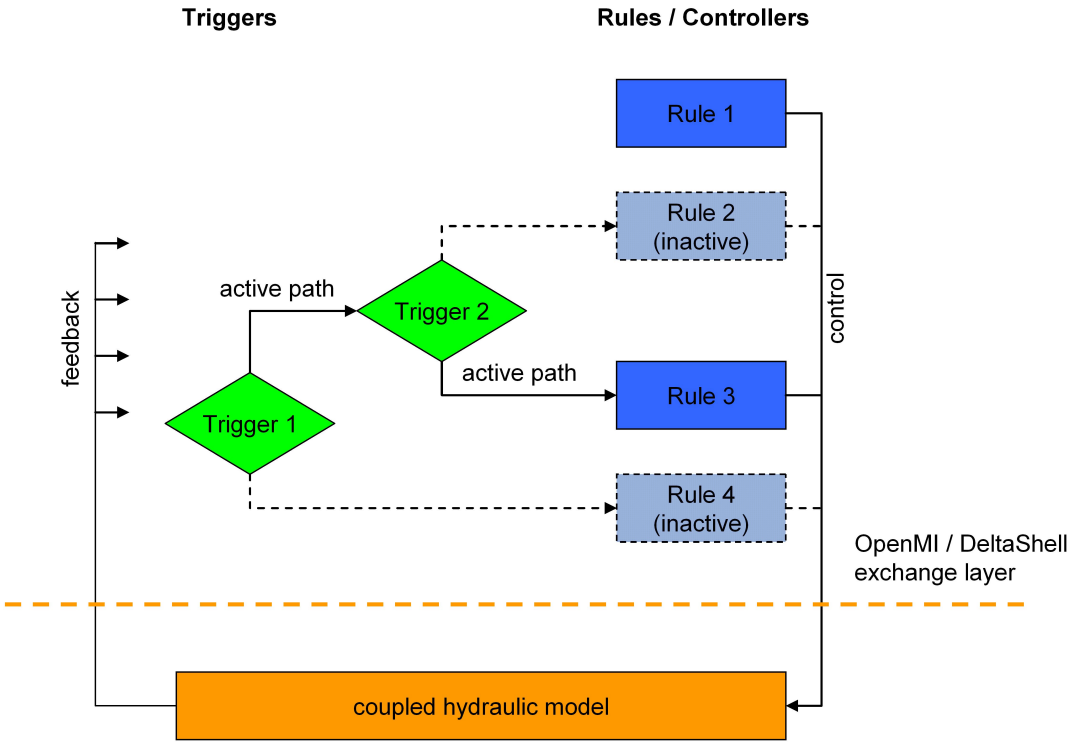



***Figure 1.4:*** *RTC-Tools configuration in typical simulation set-up: triggers, reactive operating rules and controllers linked with a hydraulic model via OpenMI*

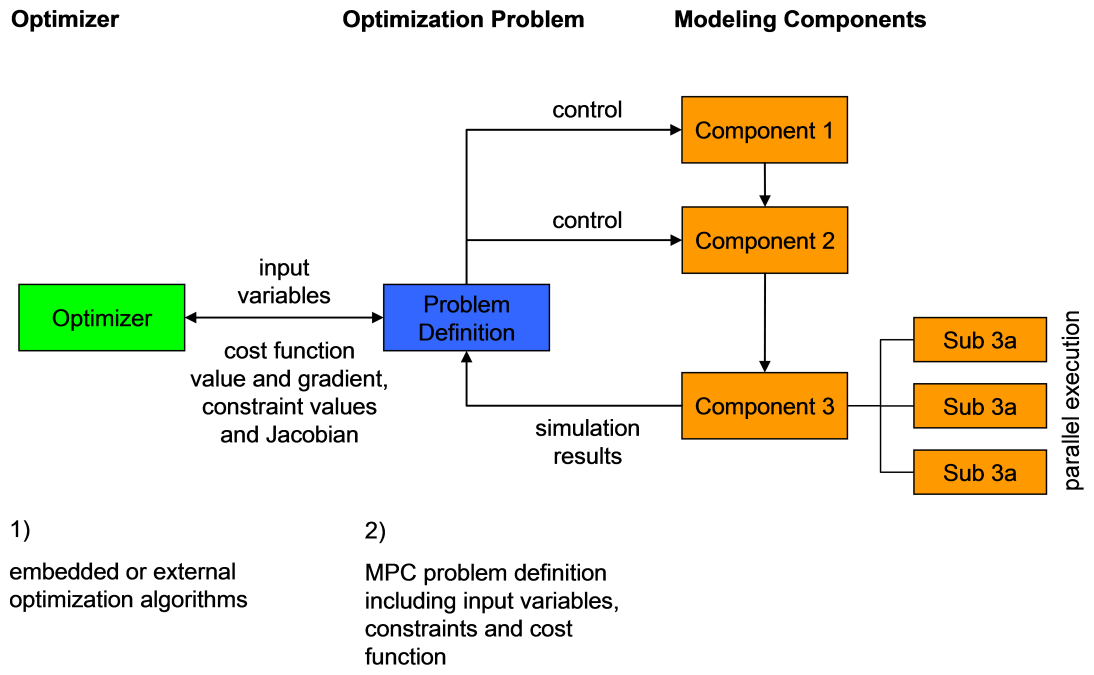



***Figure 1.5:*** *RTC-Tools configuration in typical Model Predictive Control set-up: optimizer, optimization problem definition and network components*

## 1.4 History

The RTC-Tools software package originates from the integration of several project-specific reservoir simulation modules in flood forecasting systems for river basins in Austria, Germany and Pakistan. Its original design in Java in 2007, also referred to as the Delft-FEWS Reservoir Module, aims at the simulation of pool routing in reservoirs including related feedback controllers and operating rules.

Features for supporting a sequential, nonlinear version of Model Predictive Control (MPC) were introduced in 2008 and extended in 2009 and beyond. This includes the implementation of simplified hydraulic models such as the kinematic wave or diffusive wave models as additional internal model for the predictive controller as well as the introduction of adjoint systems for selected modeling components. The latter resulted in significant speed-ups of these controllers.

In 2010, the concept of triggers for switching on and off controllers and operating rules was introduced for enabling the simulation of more sophisticated heuristic control schemes. The software was redesign in C++ and enhanced by a C# OpenMI / DeltaShell wrapper for integration into modeling packages such as SOBEK or Delft3D. Furthermore, the new formats for saving states and externalize parameters were introduced for compliant with OpenDA for implementing automatic calibration and data assimilation applications.

In 2011, the software has been integrated further into Delta Shell for replacing existing RTC functionality in SOBEK in the context of Deltares' Next Generation Hydrosoftware development. The decision to release RTC-Tools within an open source project has been made in 2012.

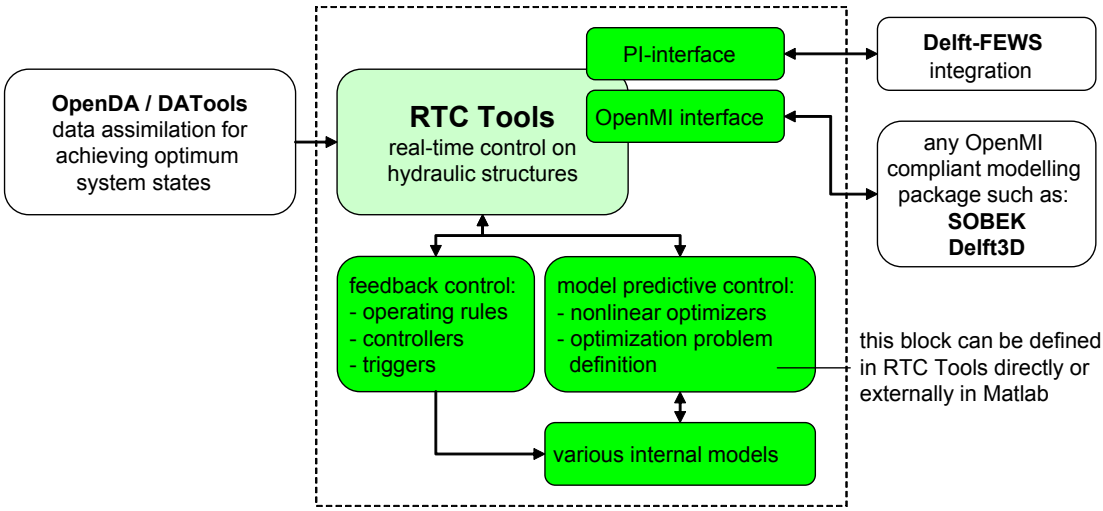



***Figure 1.6:*** *Architecture and interfaces of RTC-Tools*

## 1.5 Architecture and supported interfaces

The software does not aim at a detailed and complete simulation of large water resources systems. Therefore, it does not include any GUI or case management support. However, it is intended to solve specific RTC-related problems which we may want to integrate into larger models or forecasting systems. To support this feature, the software provides interfaces to the forecasting system Delft-FEWS (PI-XML interface) and modeling packages such as SOBEK and Delft3D via OpenMI. The latter enables a user to combine the operating rules and controllers of RTC-Tools with more detailed hydraulic modeling components founds in the hydraulic modeling packages mentioned above (see Figure 1.6).

The modeling components of RTC-Tools includes another important interface to OpenDA for applying data assimilation techniques in order to improve the system state of the modeling components. This feature may be used in the context of forecasting and predictive control application for improving the system state at forecast time and therefore also the lead time accuracy of the forecast itself.

The set-up of MPC can be done in RTC-Tools directly by using one of the integrated optimizers. Alternatively, a user may choose to externalize this part and use his preferred optimizer and optimization problem definition in Matlab or GAMS.

## 1.6 Content of this document

Section 2.3 provides background on the modeling components, governing equations and its numerical schematization. Section 2.4 and 2.5 covers the hierarchy of triggers and controllers / operating rules for setting up reactive control of a water system. In chapter 3, we present the concept of Model Predictive Control (MPC).

Details on the configuration of RTC-Tools in xml are discussed in Appendix A.

# 2 Simulation components

## 2.1 Overview

Modelling components in RTC-Tools represent water resources systems and cover the simulation of rainfall runoff and routing based on physically-based, conceptual or data-driven approaches. Hydraulic structures in these components can be controlled by (i) operating rules and controllers in combination with triggers or (ii) by predictive control techniques. The first option is based on simulation as well, so that the triggers, operating rules and controllers become part of an overall simulation model. The configuration of RTC-Tools distinguishes three layers for (1) triggers, (2) operating rules and controllers and (3) modeling components (Table 2.1). This chapter covers all three layers of simulation components (1)–(3) as well as general purpose simulation components such as the lookup table which are used in more than one layer.

According to our definition, a trigger implements conditions for

- ◇ defining when an operating rule, controller or another trigger is applied
- ◇ returning true or false, e. g. if a threshold is crossed or not.

Operating rules and controllers

- ◇ define how a structure operates,
- ◇ return a value for a control parameter, e.g. a gate opening or turbine release, which is picked up in one of the modeling components.

The combination of triggers with operating rules and controllers may form binary decision trees such as given in Figure 2.1 for representing hierarchical conditions leading to unique control actions.

From a mathematical point of view, all features in this chapter compute their outputs from available data either from the previous time step $k-1$ or from output of previous components of the same time step $k$. Note that triggers always are evaluated first, before rules and modeling components.

***Table 2.1:*** *Layers of triggers, operating rules / controllers, modeling components and general purpose components available in more than one of the layers mentioned before*

| Triggers | Rules / Controllers | Modeling Components | General |
|---|---|---|---|
| deadBand | constant | arma | accumulation |
| deadBandTime | dateLookupTable | hbv | expression |
| polygonLookup | deadBandValue | hydraulicModel | gradient |
| set | guideBand | hydrologicalModel | lookupTable |
| spreadsheet | interval | lorentGevers | lookup2DTable |
| standard | limiter | neuralNetwork | mergerSplitter |
| | pid | reservoir | unitDelay |
| | timeAbsolute | reservoirCompact | |
| | timeRelative | unitHydrograph | |

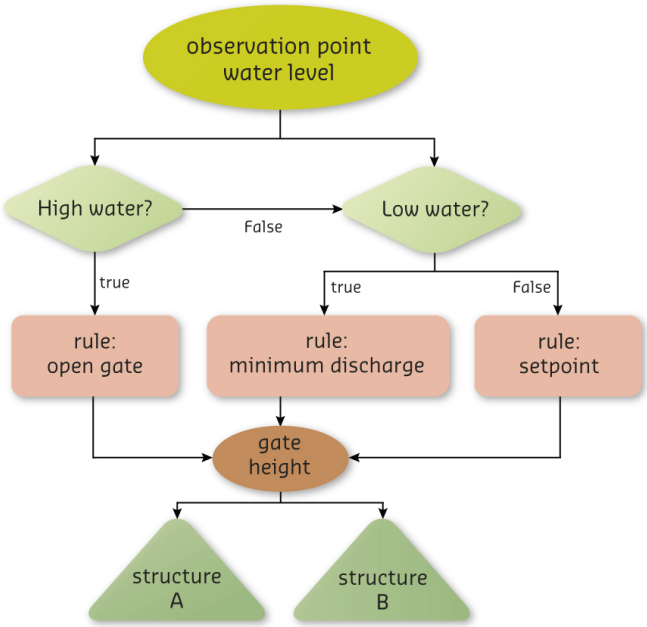



***Figure 2.1:*** *Example flow chart with feedback control*

## 2.2 General purpose components

### 2.2.1 accumulation

The component accumulates an input $x$ to the state $y$. The equation of the "accumulation" components reads

$$y^k = y^{k-1} + x^k \tag{2.1}$$

### 2.2.2 expression

The expression consists of a mathematical equation of the form

$$y^k = x_1^{k-1 \vee k} + x_2^{k-1 \vee k} \tag{2.2}$$

The following operators are supported:

$+$ (summation)
$-$ (subtraction)
$*$ (multiplication)
$/$ (division)
$\min$ (minimum)
$\max$ (maximum).

The recursive use of expressions (another expression as one of the two terms or for both) enables the implementation of more complex mathematical expressions (check the example in the configuration section).

### 2.2.3 gradient

The governing equation of the gradient reads:

$$y^k = \frac{x^k - x^{k-1}}{\Delta t} \tag{2.3}$$

### 2.2.4 lookupTable

The rule supplies a piecewise linear 1D lookup table according to

$$y^k = f(x^{k-1 \vee k}) \tag{2.4}$$

This rule is a simpler version of the dateLookupTable (section 2.4.2).

### 2.2.5 lookup2DTable

The rule supplies a piecewise linear 2D lookup table according to

$$y^k = f(x_1^{k-1 \vee k}, x_2^{k-1 \vee k}) \tag{2.5}$$

### 2.2.6 mergerSplitter

The merger rule provide a simple data hierarchy by choosing the output $y$ equal to the first of several input values $x_1, x_2, ..., x_n$ which is non-missing. Furthermore, additional output includes the sum of all input values.

### 2.2.7 unitDelay

The unit delay operator is an auxiliary tool for making data from time steps prior to the previous time step available in the simulation. By using this operator, we can refer to a historical release, for example in an operating rule, without abandoning the restarting features of the model based on the system state of a single time step. It reads

$$y^{k+1} = x^k \tag{2.6}$$

## 2.3 Modeling components

### 2.3.1 arma

The arma model (just an ar(1)-model at the moment) reads

$$\begin{aligned} e^k &= \begin{cases} x_{sim}^k - x_{obs}^k & \text{if } x_{obs}^k \text{ is available} \\ c_{ar} e^{k-1} & \text{otherwise} \end{cases} \\ y^k &= x_{sim}^k + e^k \end{aligned} \tag{2.7}$$

where $x_{obs}$ is an (external) observation, $x_{sim}$ a simulation, $e$ is the difference between observation and simulation, $c_{ar}$ is the auto regression coefficient.

### 2.3.2 hbv

This model is currently under development.

### 2.3.3 hydraulicModel

#### 2.3.3.1 Introduction

Hydraulic routing, compared to the hydrological routing techniques presented above, is a more accurate approach and may include the simulation of hysteresis and backwater effects. On the other hand, hydraulic routing has more demands with respect to the numerical solution, computational effort, and may become unstable if not properly implemented and set-up.

RTC-Tools includes a hydraulic routing method based on a mixed kinematic and diffusive wave approach. The most relevant decisions to make are the choice of an appropriate routing (kinematic / diffusive), spatial schematization (central / upwind) and time stepping scheme (explicit / implicit). The following section provides some hints on choosing the proper model and how to set it up.

#### 2.3.3.2 Diffusive wave model

The flow in one dimension is described by the De Saint-Venant equations consisting of mass (continuity) and momentum conservation. The continuity equation reads:

$$\frac{\partial A(h)}{\partial t} + \frac{\partial Q}{\partial x} = q_{lat} \tag{2.8}$$

while the non-conservative form of the momentum equation is defined by:

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} + g\frac{\partial h}{\partial x} = -c_f \frac{v\,|v|}{m}, \qquad c_f = \frac{g}{C^2} \tag{2.9}$$

with

| | |
|---|---|
| $A$ | wetted area [m$^2$] |
| $Q$ | discharge [m$^3$/s] |
| $q_{lat}$ | lateral discharge per unit length [(m$^3$/s)/m] |
| $h$ | water level [m above reference level] |
| $v$ | flow velocity [m/s] |
| g | acceleration due to gravity [m/s$^2$] |
| $m$ | hydraulic radius [m] (may be approximated to water depth for large rivers) |
| $C$ | Chézy coefficient [m$^{1/2}$/s] |
| $c_f$ | dimensionless bottom friction coefficient [-]. |

The diffusive wave equations can be derived from the complete system (2.8), (2.9) by neglecting the terms for inertia (term 1) and convection (term 2). By additional substitution of $v = Q/A$ equation (2.9) becomes

$$g\frac{\partial h}{\partial x} = -\frac{gQ\,|Q|}{C^2 A^2 m} \tag{2.10}$$

and can be converted to

$$Q = -\text{sign}\left(\frac{\partial h}{\partial x}\right) CA\sqrt{\left|\frac{\partial h}{\partial x}\right| m} \tag{2.11}$$

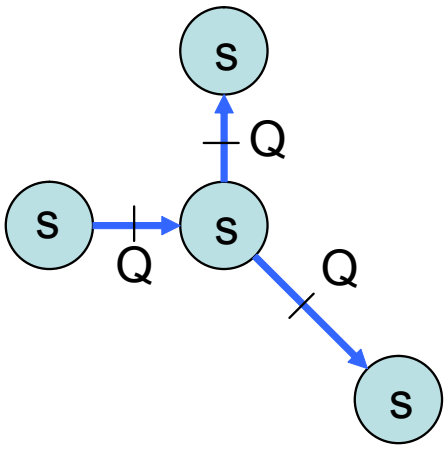



***Figure 2.2:*** *Spatial schematization of the kinematic wave model on a staggered grid*

The continuity equation (2.8) stays unchanged. Under assumption of a representative cross section for a river reach, both variables $A$ and $m$ become a geometrical function of water level $h$. By equalizing water level and energy head, hydraulic structures are represented by a simplified structure formula with the general form

$$Q = f(h_{up}, h_{down}, dg) \tag{2.12}$$

in which $dg$ = gate or weir setting (in case of a controlled structure).

The hydraulic structures are modeled by the following formulas for a weir and an orifice with fully opened gates

$$Q = \begin{cases} \frac{2}{3}\mathrm{w_s}\sqrt{\frac{2}{3}g}(\mathrm{h_{up}}-\mathrm{z_s})^{3/2}, & \text{if } h_{up}-\mathrm{z_s} > \frac{3}{2}(\mathrm{h_{down}}-\mathrm{z_s}) \\ \mathrm{w_s}(h_{down}-z_s)\sqrt{2\mathrm{g}(\mathrm{h_{up}}-\mathrm{h_{down}})}, & \text{otherwise} \end{cases} \tag{2.13}$$

in which $w_s$ = width of the structure, $z_s$ = crest level. In the case of a partially or fully closed gate ($h_{up} - z_s \geq \frac{3}{2}dg$), we apply

$$Q = \begin{cases} \mathrm{w_s}\mu\, dg\sqrt{2g(h_{up}-z_s-\mu\, dg)}, & \text{if } h_{down} < \mathrm{z_s}+\mathrm{dg} \\ \mathrm{w_s}\mu\, dg\sqrt{2\mathrm{g}(\mathrm{h_{up}}-\mathrm{h_{down}})}, & \text{otherwise} \end{cases} \tag{2.14}$$

in which $dg$ = gate setting, $\mu$ = contraction coefficient.

The spatial schematization of the kinematic wave model is done on a staggered grid. Computation nodes include the state variable storage $s$. Branches always connect two nodes and include the auxiliary variable discharge $Q$ (a major difference to the full hydraulic model where $Q$ is another state variable).

The numerical solution of the continuity equation (2.8) is resulting in:

$$\frac{A(h^{k+1}) - A(h^k)}{\Delta t} + \frac{Q_{down}^{k+1} - Q_{up}^{k+1}}{\Delta x} = q_{lat}^{k+1} \tag{2.15}$$

By substituting $s(h) = A(h)\Delta x$, we may transform equation (2.15) into a water balance in the domain of a node and get

$$s(h^{k+1}) = s(h^k) + \Delta t(Q_{up}^{k+1} - Q_{down}^{k+1} + Q_{lat}^{k+1}) \tag{2.16}$$

in which $s = storage$ at a node (state variable), $Q_{lat}$ is the total lateral inflow into the domain of the node.

The discharge in a flow branch is schematized based on equation (2.11) by

$$Q^{k+1} = f(h^k_{down}, h^k_{up}) = -\text{sign}\left(\frac{h^k_{down} - h^k_{up}}{\Delta x}\right) C(\bar{h}^k) A(\bar{h}^k) \sqrt{\left|\frac{h^k_{down} - h^k_{up}}{\Delta x}\right| m(\bar{h}^k)}, \tag{2.17}$$

in which $\bar{h}^k = \frac{h^k_{down} + h^k_{up}}{2}$

In a branch with a hydraulic structure, the flow branch is replaced by the formula of the hydraulic structures modeled by an arbitrary equation in the form

$$Q^{k+1} = f(h^k_{down}, h^k_{up}, dg^{k+1}) \tag{2.18}$$

Stability of this method turns out to be reasonable as long as the Courant-Friedrichs-Lewy (CFL) condition is fulfilled.

#### 2.3.3.3 Kinematic versus diffusive wave routing

The fact that the diffusive wave model takes into account more terms of the full dynamic Saint Venant model does not mean that it is always preferred over the simpler kinematic wave approach. Simplicity and computational performance of the latter may have advantages; in particular if results of both approaches are the same, e.g. in river reaches with steep gradients.

The following aspects may guide you to the proper approach:

- ◇ Is the slope of your river reach smaller than TODO(??): xxx?
- ◇ Is backwater a relevant effect you need to consider?
- ◇ Do you want to consider hysteresis?

If you answer one of these questions with “Yes”, consider the diffusive wave approach. Otherwise, you may try the kinematic wave method first. Note that you can mix both methods by defining the related tag in each flow branch.

#### 2.3.3.4 Model set-up

The spatial schematization of your routing network is a trade-off between accuracy (a higher resolution means more accuracy) and CPU time. For flow routing purposes only, already a very course spatial schematization may achieve the required accuracy from the control point of view. We recommend the following procedure for defining your routing network:

1. Indicate all nodes you need to include: boundary conditions, upstream and downstream node of hydraulic structures, stream flow gauges, bifurcations and confluences, nodes with lateral inflows or extractions (can be also lumped into neighboring nodes).
2. Place additional nodes between the existing ones, if required for accuracy.

All nodes require a level-storage relation. One way to get those is a detailed analysis of the surrounding channel network (typically halfway to the next node) in terms of the area

and storage at different elevations. An easier and often sufficient option is the selection of a typical cross section at the node and its multiplication by the length of the reaches around. If you selected the upwind schematization (see section 3.4.3), take the cross section you are using in the flow branch.

If you routing network includes flooding and drying, take care that the lowest level of your level-storage table or equation in your node is lower than the lowest elevation in the cross sections around. Since the model implementation does not include any dedicated procedure for flooding and drying, violating this condition may result in negative water depth at nodes and problems with robustness. If you use the upwind schematization together with the level-storage relation derived from it, this condition is already fulfilled.

Flow branches require a cross section. You may again aim at deriving an aggregated cross section from the available data of the branch. The simpler approach is again the selection of a typical cross section. Depending on the spatial schematization (see 3.4.3), select a typical cross section close to the upstream node for the upwind option and one halfway along the branch for the central option.

### 2.3.4 hydrologicalModel

This model is currently under development.

### 2.3.5 lorentGrevers

This model is currently under development.

### 2.3.6 neuralNetwork

We use a recurrent neural network formulation given by the following equations for the sum of neuron $\mu$ and the (nonlinear) transfer function

$$\begin{aligned} y_\mu^k &= \sum_{v=1}^{\mu-1} w_{\mu,v}^{\text{neuron}} x_v^k + \sum_{v=\mu}^{K} w_{\mu,v}^{\text{neuron}} x_v^{k-1} + \sum_{v=1}^{L} w_{\mu,v}^{\text{input}} u_v^k \\ x_\mu^k &= h_\mu \left( y_\mu^k \right) \end{aligned} \tag{2.19}$$

where

| | |
|---|---|
| $x_\nu^k$ | is the value of neuron $v$ at time step $k$. |
| $y_\mu^k$ | is the weighted sum of inputs to neuron $\mu$ at time step $k$. |
| $w_{\mu,v}^{\text{neuron}}$ | is the weighting given to neuron $v$ when calculating the sum for neuron $\mu$. |
| $w_{\mu,v}^{\text{input}}$ | is the weighting given to input $v$, when calculating the sum for neuron $\mu$. |
| $u_\nu^k$ | is the value of input $v$ at time step $k$. |
| $K$ | is the number of neurons. |
| $L$ | is the number of inputs. |
| $h_\mu$ | is the activation function for neuron $\mu$. |

Note that the for $\mu = 1$, the first summation term in (2.19) is empty and hence should be taken as zero.

### 2.3.7 reservoir

#### 2.3.7.1 Mathematical model

The basic equation for pool routing in a reservoir is

$$\begin{aligned} \frac{\mathrm{d}s}{\mathrm{d}t} &= I - Q(h) \\ h &= f(s) \end{aligned} \tag{2.20}$$

where

| | |
|---|---|
| $I$ | inflow into the reservoir |
| $Q$ | release of the reservoir |
| $s$ | storage in the reservoir (state variable) |
| $h$ | water level in the reservoir |
| $t$ | time. |

We assume the relation $f()$ between storage $s$ and water level $h$ to be a function or a piece-wise linear lookup table.

The release $Q$ from the reservoir can be further subdivided into an into a controlled release $Q_c$ and an uncontrolled release $Q_u$ according to

$$Q(h, dg) = Q_c(h, dg) + Q_u(h) \tag{2.21}$$

where $dg$ is the setting of a hydraulic structure. Whereas the controlled release is a function of the water level $h$ (under assumption that its maximum capacity depends on the water level in the reservoir) and the setting of the structure $dg$. The uncontrolled release is only a function of the reservoir's water level $h$ representing for example an uncontrolled spillway with a fixed crest level.

#### 2.3.7.2 Numerical schematization

The explicit schematization, also referred to as "Forward Euler", for the pool routing equation reads

$$s^k = s^{k-1} + \Delta t(I^k - Q_c^k(h^{k-1}, dg^k) - Q_u^k(h^{k-1})) \tag{2.22}$$

and is conditionally stable for sufficiently small time steps. The unconditionally stable implicit version is

$$\begin{aligned} s^k = s^{k-1} &+ \Delta t((1-\theta)I^{k-1} + \theta I^k) \\ &- (1-\theta)Q_c^{k-1}(h^{k-1}, dg^{k-1}) - \theta Q_c^k(h^k, dg^k) \\ &- (1-\theta)Q_u^{k-1}(h^{k-1}) - \theta Q_u^k(h^k)) \end{aligned} \tag{2.23}$$

where $0.5 \leq \theta \leq 1.0$ is the time weighting coefficient of the "Theta" scheme shifting gradually from a more accurate second-order Crank-Nicholson method for $\theta = 0.5$ to a more robust, first-order "Backward Euler" scheme for $\theta = 1.0$.

### 2.3.8 reservoirCompact

This is a dedicated component for hydropower reservoirs with gated spillways. Depending on the availability of a forebay elevation input $z_{fb}$, the mass balance equation is either used to compute the new reservoir storage $s^k$ or the reservoir's mass balance residuum $r^k$. In the case of no forebay elevation input, the new storage is computed by

$$\begin{aligned} s^k &= s^{k-1} + \Delta t(\sum_n Q^k_{I,n} - Q^k_O) \\ r^k &= 0 \end{aligned} \tag{2.24}$$

where $Q_I$ and $Q_O$ are reservoir inflows and outflow, respectively, and $\Delta t$ is the time step. In the case of an available forebay elevation, the mass balance is used to compute the mass balance residuum by

$$r^k = \frac{s^k(z^k_{fb}) - s^{k-1}}{\Delta t} - \sum_n Q^k_{I,n} + Q^k_O \tag{2.25}$$

The first mode is primarily used to simulate reservoirs or optimize them in a sequential mode (check details in Chapter 3). The second mode serves to run the reservoir in update mode, in case of having inflow, outflow and forebay elevation available and checking the reservoir's mass balance. Furthermore, it is used in the simulaneous optimization mode.

The forebay elevation $z_{fb}$ directly depends on the storage given by

$$z^k_{fb} = f_{ls}(s^k) \tag{2.26}$$

where the level storage relation $f_{ls}()$ can be configured as a piecewise linear lookup table or a polynominal function.

The 'reservoirCompact' component implements several options for computing the tailwater elevation $z_{tw}$ according to the equations

$$z^k_{tw} = f_{tw}(Q^k_O) \tag{2.27}$$

$$z^k_{tw} = a + bz^{k-1}_{fb,d} + c(Q^k_O)^d \tag{2.28}$$

$$z^k_{tw} = z^k_{tw,ext} + a(Q^k - Q^k_{O,ext}) \tag{2.29}$$

where $f_{tw}()$ in Equation (2.27) is a piecewise linear table with the relation between the outflow of the reservoir and the tailwater elevation. Note that this formulation is not taking into account backwater effects from downstream projects. Equation (2.28) is a tailwater equation which considers the reservoir's outflow and the forebay elevation $z_{fb,d}$ of a downstream reservoir. It includes the parameters $a, b, c$ and $d$. Equation (2.29) makes use of an external tailwater forecast $z_{tw,ext}$ for a given outflow trajectory $Q_{O,ext}$. The simulated tailwater is computed by a linear variation of the provided, external tailwater and the products of $a$ parameter a and the difference between the simulated and external outflow. Furthermore, the tailwater can be provided by an external time series or modeling component.

The head $h$ of the turbines is computed by

$$h^k = z^k_{fb} - z^k_{tw} \tag{2.30}$$

Environmental obligations may require spill operations. In this case, the spill is either provided as an absolute flow $Q_{S,TGT}$ or provided by a spill rate $q_{S,TGT}$ which is computed as percentage of the total outflow $Q_O$ according to

$$Q^k_{S,TGT} = \frac{q^k_{S,TGT}}{100} Q^k_O \tag{2.31}$$

The turbine flow $Q_T$ at an aggregated project level is bounded by the requirement of a minimum power production and the maximum turbine capacity in term of a maximum power production or an external maximum turbine flow $Q_{TX,ext}$ provided by

$$\begin{aligned} Q^k_{TM} &= f_{PQ}(P^k_M, h^k) \\ Q^k_{TX} &= \min(f_{PQ}(P^k_X, h^k), Q^k_{TX,ext}) \end{aligned} \tag{2.32}$$

where $Q_{TM}$ and $Q_{TX}$ are the minimum and maximum turbine flows, respectively, $P_M, P_X$ are the power production bounds and the function $f_{PQ}()$ provides the relations between power, head and the turbine flow. The function $f_{PQ}()$ and its corresponding inverse function $f_{QP}()$ are derived from information about the turbine efficiency, which is either provided by a lookup table as a head dependent efficiency (unit: power / discharge) or by a two-dimensional lookup table as a dependency on head and flow (dimensionless).

The spill target and the turbine flow bounds in combination with an embedded prioritized release policy enables us to split the total outflow into spill and turbine flow. The minimum generation requirements in effect for voltage support are the highest priority objective. Additional outflow is allocated for the spill target, then for additional power generation up to the maximum turbine capacity, and finally directed again to the spill. This implicit outflow distribution enables us to keep the total outflow as a single optimization variable without the need for optimizing spill and turbine flow separately together with constraints for enforcing a release procedure. The related equations for the spill and turbine flow become

$$Q^k_S = \begin{cases} Q^k_O - Q^k_{TX} - Q^k_{MISC} & \text{if } Q^k_O > Q^k_{TX} + Q^k_{ST} + Q^k_{MISC} \\ Q^k_{ST} & \text{if } Q^k_{TX} + Q^k_{ST} > Q^k_O > Q^k_{TM} + Q^k_{ST} \\ Q^k_O - Q^k_{TM} - Q^k_{MISC} & \text{if } Q^k_{TM} + Q^k_{ST} > Q^k_O > Q^k_{TM} \\ 0 & \text{else} \end{cases} \tag{2.33}$$

$$Q^k_T = \begin{cases} Q^k_{TX} & \text{if } Q^k_O > Q^k_{TX} + Q^k_{ST} \\ Q^k_O - Q^k_{ST} - Q^k_{MISC} & \text{if } Q^k_{TX} + Q^k_{ST} > Q^k_O > Q^k_{TM} + Q^k_{ST} \\ Q^k_{TM} & \text{if } Q^k_{TM} + Q^k_{ST} > Q^k_O > Q^k_{TM} \\ Q^k_O - Q^k_{MISC} & \text{else} \end{cases} \tag{2.34}$$

Finally, the power generation of a project is computed by

$$P^k = f_{QP}(Q^k_T, h^k) \tag{2.35}$$

The component can handle SI as well as imperial units. The two options are summarized in Table 2.2.

***Table 2.2:*** *Units of inputs and outputs of the 'reservoirCompact' component*

| Variable | Option 1: SI | Option 2: Imperial |
|---|---|---|
| flows | $m^3/s$ | kcfs (kilo cubic feet per second) |
| elevations | m | ft (feet) |
| storage | $m^3$ | kcfs * day (kcfs over one day) |
| turbine efficiency (option 1) | MW/($m^3/s$) | MW/kcfs |
| turbine efficiency (option 2) | - | - |
| power | MW | MW |

### 2.3.9 unitHydrograph

Unit hydrograph provides a rainfall-runoff modeling based on the concept of the unit hydrograph.

## 2.4 Operating rules and controllers

### 2.4.1 constant

This simple rule defines a user-defined constant output $y^k$ according to

$$y^k = \text{const.} \tag{2.36}$$

It is typically applied in combination with triggers (see next section) for switching between several predefined states of a structure, e.g. fully opened or fully closed.

### 2.4.2 dateLookupTable

The date lookup table is a 2D lookup table with the time axis as one of its dimensions. Its discrete form reads

$$y^k = f(t, x^{k-1 \vee k}) \tag{2.37}$$

The resolution of the time axis $t$ is in days of the year. The value axis $x$ may have any range. A typical application of the rule would be the definition of a minimum release of a reservoir as a function of the day of the year and the water level of the reservoir.

### 2.4.3 deadBandValue

The dead band value rule is a discrete rule for suppressing the output of another rule until its rate of change becomes higher than a certain threshold. It reads

$$x^k = \begin{cases} x^{k-1} & \text{if } \left|x^k - x^{k-1}\right| < \text{threshold} \\ x^k & \text{otherwise} \end{cases} \tag{2.38}$$

It is often applied to limit the number of adjustments to movable elements of hydraulic structures in order to increase their life time.

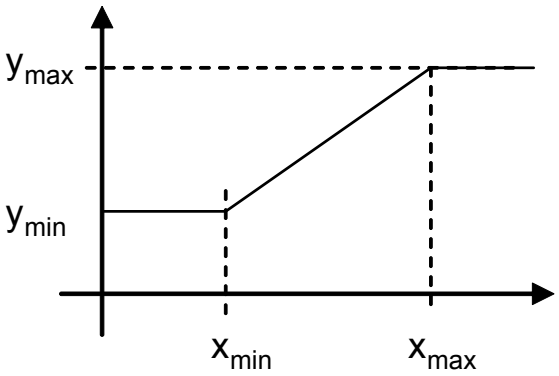



***Figure 2.3:*** *Graphical representation of guideBand rule*

### 2.4.4 guideBand

The guide band rule provides a linear interpolation from input $x$ to output $y$, if $x$ is between two input threshold $x_{\min}$ and $x_{\max}$. Otherwise, the output is limited to defined minimum and maximum output threshold $y_{\min}$ and $y^{\max}$. The rule reads

$$y^k = \begin{cases} y_{\min} \\ y_{\min} + \frac{x^{k-1 \vee k}(y_{\max} - y_{\min})}{x_{\max} - x_{\min}} \\ y_{\max} \end{cases}, \quad \text{if} \quad \begin{array}{c} x^{k-1 \vee k} \leq x_{\min} \\ x_{\min} < x^{k-1 \vee k} < x_{\max} \\ x_{\max} \leq x^{k-1 \vee k} \end{array} \tag{2.39}$$

A graphical representation of the rule is presented in Figure 2.3.

The input and output thresholds $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ can be constant, a function of time or provided by a time series, e.g. from an external input or a result from the execution of a prior rule.

A typical application of the rule is the enforcement of a reservoir storage $s$ between a certain range and the use of the available storage for equalizing the release. If the storage is approaching or down-crossing the lower storage limit $s_{\min}$, the release is set to the minimum flow (zero is no minimum flow is defined). If the storage is approaching or up-crossing the upper limit $s_{\max}$, the release is set to maximum capacity.

### 2.4.5 interval

The interval controller is a simple feedback controller according to the control law

$$y^k = \begin{cases} y_{\max} & \text{if } x^{k-1} > sp^k + \frac{1}{2}D \\ y_{\min} & \text{if } x^{k-1} < sp^k - \frac{1}{2}D \\ y^{k-1} & \text{otherwise} \end{cases} \tag{2.40}$$

where $x^{k-1}$ is an input variable, $sp^k$ is a setpoint, $D$ is a dead band around the setpoint, and $y^k$ is the controller output.

### 2.4.6 limiter

In contrary to the deadBand rule defined above, the discrete limiter rule restricts the change of a variable to a relative threshold $p$. It reads

$$y^k = \begin{cases} (1-p)x^{k-1} & \text{if } x^k < (1-p)x^{k-1} \\ (1+p)x^{k-1} & \text{if } x^k > (1+p)x^{k-1} \\ x^k & \text{otherwise} \end{cases} \tag{2.41}$$

where $p$ is the maximum relative rate of change. The configuration also accounts for an absolute rate of change according to the condition $x^k < \Delta p\, x^{k-1}$ where $\Delta p$ is the absolute rate of change.

A typical application of the rule is the limitation of release changes from a reservoir for avoiding too steep flow gradients downstream.

### 2.4.7 pid

The Proportional-Integral-Derivative controller (PID controller) is a generic feedback controller including an optional disturbance term commonly used in industrial control systems. It reads

$$\begin{aligned} e(t) &= x(t) - sp(t) \\ y(t) &= K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) + K_f d(t) \end{aligned} \tag{2.42}$$

where $e(t)$ is the difference between a process variable $x(t)$ and a setpoint $sp(t)$, $K_p, K_i, K_d$ are the proportional, integral and derivate gains, respectively, the optional feed forward term consists of a multiplicator $K_f$ and an external disturbance $d(t)$, $y(t)$ is the controller output.

The discrete form of equation (2.42) in RTC-Tools reads

$$\begin{aligned} e^k &= x^{k-1} - sp^{k-1} \\ E^k &= E^{k-1} + \Delta t\, e^k \\ y^k &= K_p e^k + K_i E^k + K_d \frac{e^k - e^{k-1}}{\Delta t} + K_f d^k \end{aligned} \tag{2.43}$$

where $E$ is the integral of $e$. The implentation includes a limitation of the maximum velocity according to

$$y^k = \begin{cases} (1 - \Delta t \Delta y_{\max}) y^{k-1} & \text{if } y^k < (1 - \Delta t \Delta y_{\max}) x^{k-1} \\ (1 + \Delta t \Delta y_{\max}) y^{k-1} & \text{if } y^k > (1 + \Delta t \Delta y_{\max}) x^{k-1} \\ y^k & \text{otherwise} \end{cases} \tag{2.44}$$

Furthermore, the integral part is corrected by a wind-up correction according to

$$\begin{aligned} eI^k &\leq \frac{y_{\max} - K_p e^k + K_i E^k - K_d \frac{e^k - e^{k-1}}{\Delta t} - K_f d^k}{K_i} \\ eI^k &\geq \frac{y_{\min} - K_p e^k + K_i E^k - K_d \frac{e^k - e^{k-1}}{\Delta t} - K_f d^k}{K_i} \end{aligned}, \tag{2.45}$$

if the minimum and maximum settings of the actuator are met.

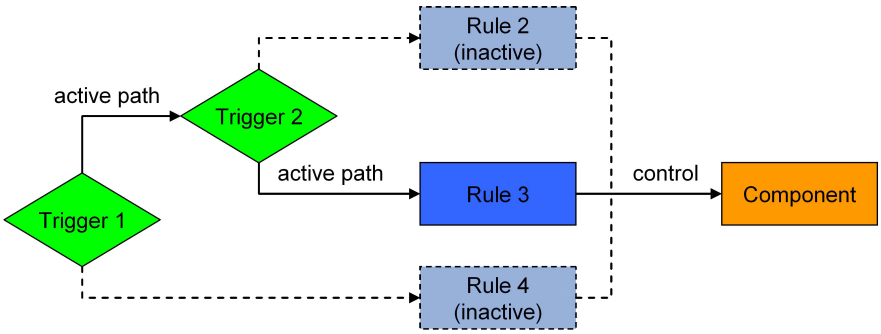



***Figure 2.4:*** *Hierarchical definition of deadBand and standard triggers*

### 2.4.8 timeAbsolute

This rule reads

$$y^k = x^k \tag{2.46}$$

where $x$ is an external time series or output of a previous model.

### 2.4.9 timeRelative

This rule reads

$$y^k = x^\tau \tag{2.47}$$

where $\tau$ is a relative time reference. When the rule is switched on, the relative time is i) put to zero, ii) put to a value based on an existing $y$ for which equation (2.47) is fulfilled.

## 2.5 Triggers

### 2.5.1 deadBandTrigger

The dead band trigger checks the input data for an upper or lower threshold crossing. The trigger is active, in case of an up-crossing of the upper threshold (condition 1). It is inactive, in case of a down crossing of the lower threshold (condition 2). In the range in-between, the trigger keeps its former state. The rule reads

$$y^k = \begin{cases} 1 & \text{if } x_1^{k-1 \vee k} > x_2^{k-1 \vee k} \\ 0 & \text{if } x_3^{k-1 \vee k} < x_4^{k-1 \vee k} \\ y^{k-1} & \text{otherwise} \end{cases} \tag{2.48}$$

where $x_1, x_2, x_3, x_4$ are either constant values or time series. The following operators are supported: $>, \geq, =, \neq, \leq, <$. Dead band triggers are used to switch on and off pumps or turbines and the dead band ensures that the device is not switched on or off too often.

The dead band trigger as well as the standard trigger (see section 2.5.6) may include other triggers which are evaluated in case of an active or inactive trigger state. This feature enables a user to build complex decision trees for selecting unique rules for controlling a structure (Figure 2.4).

Rules which are referenced in a trigger and are associated with an inactive path will be deactivated. This procedure supports that a hydraulic structure is controlled by a unique controller or rule.

### 2.5.2 deadBandTime

The dead band time trigger checks a time series for a number of subsequent up-crossings $n_{\text{up}}$ or down-crossing $n_{\text{down}}$ from its current value. If the crossing is observed for at least the user-defined number of time steps, the new value is used. The trigger reads

$$x^k = \begin{cases} x^k & \text{if } \{x^{k-n_{\text{up}}+1}, ..., x^k\} > x^{k-n_{\text{up}}} \\ x^k & \text{if } \{x^{k-n_{\text{down}}+1}, ..., x^k\} < x^{k-n_{\text{down}}} \\ x^{k-1} & \text{otherwise} \end{cases} \tag{2.49}$$

An application for this trigger is the activation or deactivation of alarm levels. The increase of alarm level may happen immediately ($n_{\text{up}} = 0$), whereas the decrease of an alarm level could be done only after a number of $n$ time steps have passed ($n_{\text{down}} = n$) without further threshold crossings.

### 2.5.3 polygonLookup

The polygon lookup trigger checks, if a point is inside of a set of polygons. If this is the case, it returns the user-defined value of the specific polygon. The point is defined by the values of two time series, referred to as the $x_1$ and $x_2$ coordinate of the point. The rule reads

$$y^k = \begin{cases} y_1 & \text{if } (x_1^{k-1 \vee k}, x_2^{k-1 \vee k}) \in P_1 \\ \vdots & \vdots \\ y_n & \text{if } (x_1^{k-1 \vee k}, x_2^{k-1 \vee k}) \in P_n \\ y_{default} & \text{otherwise} \end{cases} \tag{2.50}$$

where *y* is the result of the rule and $\{P_1, ..., P_n\}$ is a set of polygons.

Figure 2.5 presents an example for the application of the trigger to the definition of warning level for controlling a lake release in Canton Bern, Switzerland.

### 2.5.4 set

The trigger enables a logical combination of other triggers, combined by a logical operator. It reads

$$y^k = x_1^{k-1 \vee k} \wedge x_2^{k-1 \vee k} \tag{2.51}$$

The following operators are supported: $\wedge$ (AND), $\vee$ (OR), XOR.

If more than two terms needs to be combined, the set can be used recursively by defining another set as one of the two terms. Therefore, the expression $y^k = x_1^k \wedge (x_2^k \vee x_3^k)$ is represented by a hierarchy of two sets (check the example in the configuration chapter).

### 2.5.5 spreadsheet

The spreadsheet trigger enables the definition of a new trigger state based on its old state and a maximum number of three additional inputs. Besides off (0), on (1) states, all other positive integer values larger than 1 can be processed.

Figure 2.6 shows an application of the trigger to the combination of alarm levels for Lake Thun, Canton Bern, Switzerland.

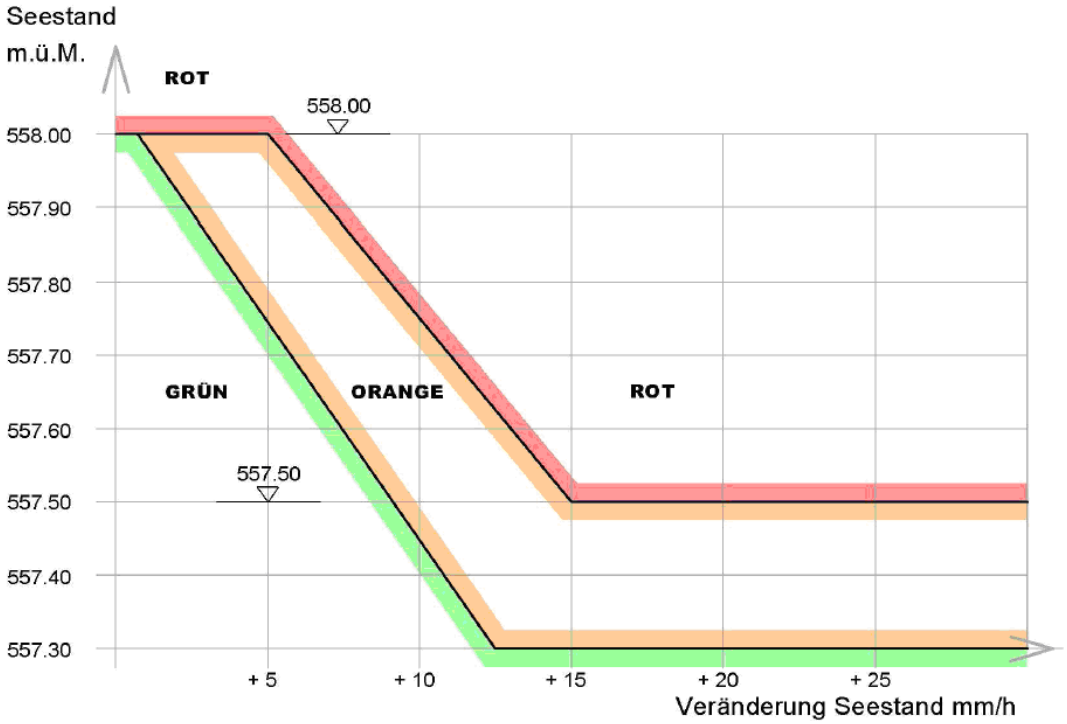



***Figure 2.5:*** *Example for the application of the polygon trigger to the definition of warning levels for controlling a lake release at Lake Thun, Canton Bern, Switzerland*

### 2.5.6 Standard

The standard trigger compares two input values and returns true (1) or false (0):

$$y^k = \begin{cases} 1 & \text{if } x_1^{k-1 \vee k} > x_2^{k-1 \vee k} \\ 0 & \text{otherwise} \end{cases} \tag{2.52}$$

Beside the $>$ operator that is used in Equation 2.52 the following operators are supported: $>, \geq, =, \neq, \leq, <$.

### 2.5.7 Expression

The expression trigger works like the expression component in section 2.2.2. But since triggers are evaluated before components within the time step simulation, it can make sense to use the expression trigger instead of an expression component in order to prepare data of the current time step.

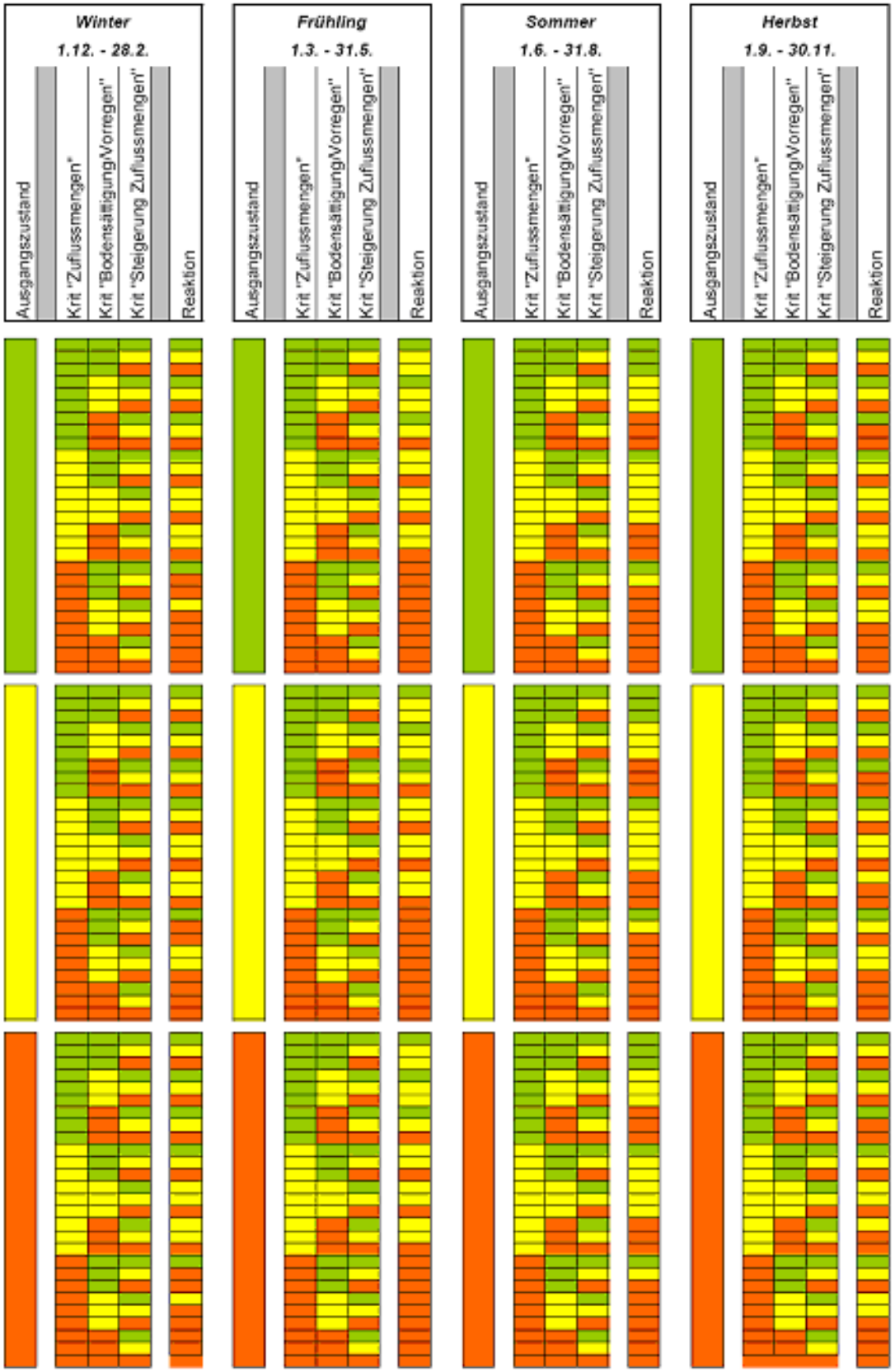



***Figure 2.6:*** *Spreadsheet for combining alarm levels, Lake Thun, Canton Bern, Switzerland*

# 3 Optimization components

## 3.1 Introduction and deterministic optimization setup

We consider a discrete time dynamic system according to

$$\begin{aligned} x^k &= f(x^{k-1}, x^k, u^k, d^k) \\ y^k &= g(x^k, u^k, d^k) \end{aligned} \tag{3.1}$$

where $x, y, u, d$ are respectively the state, dependent variable, control and disturbance vectors, and $f(), g()$ are functions representing an arbitrary linear or nonlinear water resources model. If being applied in Model Predictive Control (MPC), (3.1) is used for predicting future trajectories of the state $x$ and dependent variable $y$ over a finite time horizon represented by $k = 1, ..., N$ time instants, in order to determine the optimal set of controlled variables $u$ by an optimization algorithm. Under the hypothesis of knowing the realization of the disturbance $d$ over the time-horizon, i.e. the trajectory $\{d^k\}_1^N$, a Sequential MPC approach, also referred to as Single Shooting, can be formulated as follows

$$\begin{aligned} &\min_{u \in \{0..T\}} \sum_{k=1}^{N} J(\widetilde{x}^k(u,d), \widetilde{y}^k, u^k) + E(\widetilde{x}^N(u,d), \widetilde{y}^N, u^N) \\ &\text{subject to} \quad h(\widetilde{x}^k(u,d), \widetilde{y}^k, u^k, d^k) \leq 0, \quad k = 1, ..., N \end{aligned} \tag{3.2}$$

where $\widetilde{x}^k(u,d), \widetilde{y}^k$ are the simulation results, $J()$ is a cost function associated with each state transition, $E()$ is an additional cost function associated to the final state condition, and $h()$ are hard constraints. In the corresponding simultaneous or collocated optimization approach, the states $x$ become additional variables of the optimization problem, and the process model (3.1), gets an equality constraint of the optimization problem according to

$$\begin{aligned} &\min_{u,x,y \in \{0..T\}} \sum_{k=1}^{N} J(x^k, y^k, u^k) + E(x^N, y^N, u^N) \\ &\text{subject to} \quad h(x^k, y^k, u^k, d^k) \leq 0, \quad k = 1, ..., N \\ &\qquad\qquad x^k - f(x^{k-1}, x^k, u^k, d^k) = 0 \\ &\qquad\qquad y^k - g(x^k, u^k, d^k) = 0 \end{aligned} \tag{3.3}$$

Both methods lead to identical solutions, but have pros and cons for specific optimization problems in terms of runtime performance. The sequential approach (3.2) is more efficient, if hard constraints are defined on the control variables $u$ only and gets inefficient for hard constraints on states $x$. Because of the state dependency on all prior control variables $u$, the constraint Jacobian becomes dense in the lower trianglar matrix. In this case, the simultaneous approach (3.3) shows better efficiency, since states become an input of the optimizer and the constraint Jacobian gets sparse.

RTC-Tools does not strictly distinguish between one and the other approach. In fact, optimization problems can be set up either in a sequential or simultaneous way. Furthermore, both methods can be mixed leading to the so-called multiple-shooting approach. The choice depends on the setup of the specific modelling components. For example, the pool routing in a reservoir can be configured in such a way that the control variable is the release only, and the reservoir level is a result of a simulation. Alternatively, the optimizer may provide both release and reservoir level for computing the mass balance residuum which becomes an

***Table 3.1:*** *Model predictive control features implemented in RTC-Tools*

| Optimizer | MPC problem definition |
|---|---|
| IPOPT (embedded) | constraints |
| MINOS (Matlab/TOMLAB) | min/max bounds and rate of change limits |
| SNOPT (Matlab/TOMLAB) | for optimization variables, system states, outputs |
| | cost function terms |
| | rate of change with variable exponent |
| | absolute with variable exponent |
| | several performance indicators |

equality constraint of the optimization problem. It is obvious that the first setup corresponds to the sequential (3.2), the second one to the simultaneous approach (3.3).

## 3.2 Multi-stage stochastic optimization setup

The extension of the deterministic to a stochastic optimization is achieved by replacing the single-trace forecast by a forecast ensemble and computing the objective function values $J$ and $E$ as the probability-weighted sum of the objective function terms of the individual ensemble branches or scenarios. This lead to a reformulation of Sequential MPC approach of Equation (3.2) as

$$\min_{u \in \{0..T\}} \sum_{j=1}^{M} p^j \left[ \sum_{k=1}^{N} J(x^{j,k}(u), y^{j,k}(x,u), u^{j,k}) + E(x^{j,N}(u), y^{j,N}(x,u), u^{j,N}) \right] \tag{3.4}$$

where $p^j$ is the probability of the scenario $j = 1, \ldots, M$ and $M$ is the total number of scenarios. Whereas the disturbance $d$ as well as the model states $x$ and outputs $y$ are treated independently in each scenario, the control variable $u$ is the key to the properties of the stochastic optimization approach. The most general formulation is achieved by the use of scenario trees. One way for its definition is the scenario tree nodal partition matrix $P(j,k)$ with the dimensions $m \times n$. The matrix assigns the control at time step $k$ of scenario $j$ to the control vector $u$. This enables us to define a common control trajectory for all scenarios at the beginning of the forecast horizon when future system states are still uncertain.

When uncertainty gets resolved over the forecast horizon, for example when a forecasted precipitation is finally observed, we introduce branching points to receive an independent control in each scenario at the end of the forecast horizon. Equation (3.5) presents an example of a nodal partition matrix for a simple tree with two scenarios and a branching point at the second time step.

$$P = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 5 & 6 \end{bmatrix} \tag{3.5}$$

The introduction of multiple branching points at several time steps leads to a multi-stage stochastic optimization. From a technical perspective, the solution of the multi-stage stochastic optimization (Equation (3.4)) is very similar to the solution of the deterministic setup of Equation (3.2). The main difference is the number of dimensions of the optimization problem. According to our experience, it is typically increasing by a factor of 5-20, if we derive the scenario tree from a meteorological ensemble forecast.

The extension of the Simultaneous MPC approach to a multi-stage stochastic optimization is idetical to the one of the sequential approach presented above. An addition is the handling of hard constraints on system states and outputs. These are executed independently on every branch of the tree.

## 3.3 Set-up of the optimization problem

### 3.3.1 Control variables

The setup of the optimization problem starts with the definition of the input variables of the optimizer, i.e. the control $u$ in the sequential setup as well as the optional state $x$ and dependent variable $y$ in the simultaneous setup. Each variable can be of the types

- ◇ CONTINUOUS, representing a continuous variable
- ◇ INTEGER, representing an integer variable (Note that there is still no RTC-Tools-internal solver supporting Mixed Integer Nonlinear Programming (MINLP), therefore, this options depends on interfacing a suitable solver via the Matlab or GAMS interfaces)
- ◇ TIMEINSTANT for the setup of a Time-Instant Optimization MPC (TIO-MPC) (experimental version!)

The optimization variable may includes optional bounds for the according to $u_{\min} \leq u \leq u_{\max}$. In case of the TIMEINSTANT of the TIO-MPC, the minimum and maximum bounds are compulsary and represent the two states of the control variable. At each time instant, the state is switches from one value to the other.

Each optimization variable can include a scaling factor. It is good practice to scale all variables in such a way that they are in the same order of magnitude. For example, if variables cover both reservoir levels and releases, the scaling factor of the level should be defined in such as way that its range, i.e. the difference between maximum and minimum operating levels of the reservoirs, multiplied by the scaling factor is in the range of the releases. User-defined scaling usually outperformed automatic, built-in scaling options of the optimizers.

The time step of the control variable $u$ and the simulation can be different. This enables a courser discretization in the optimization compared to the simulation, for example in case of using an explicit model with severe time step restrictions. At this moment, the following aggregation options are supported:

- ◇ BLOCK, keeping the the recent value persistent until a new value is specified
- ◇ LINEAR, conducting a linear interpolation of the control variable between two instants at which it is defined by the optimization algorithm

### 3.3.2 Constraints

RTC-Tools takes into account constraints in two ways: i) as a soft constraint in the objective function or ii) as a hard constraint. This section covers the definition of hard constraint either as an inequality constraint or equality constraint of the optimization problem. The next section provides more details on the definition of soft constraints and the objective function in general.

Hard constraints are available for control variables, states or dependent variables. In all cases, the entity may have bounds and a maximum rate of change according to

$$\begin{aligned} &u_{\min} \leq u^k \leq u_{\max} \\ &\Delta u_{\min} \leq u^k - u^{k-n} \leq \Delta u_{\max} \quad n = k-N, ..., k-1 \end{aligned} \tag{3.6}$$

where $N$ is the number of time steps of the rate of change constraint looking back in time.

The definition of constraints on control variables is straightforward both in the sequential and simultaneous optimization setup and requires no additional information. Constraints on states and dependent variables is only efficient in the simultaneous setup and require information on how these are traced back to optimization variables. This includes the definition of the related optimization variables, the modeling components involved and the number of time steps looking back in time.

Example 1:

Consider a reservoir represented by the mass balance equation below in simultaneous optimization mode, represented by

$$r^k = s^k - s^{k-1} - \Delta t(Q_{in}^k - Q_{out}^k) \tag{3.7}$$

where $r$ is the residuum of the mass balance, $s$ and $Q_{out}$ are the two optimization variables for reservoir storage and release, respectively. For defining an equality constraint on the residuum $r$, we consider a bound constraints with $r_{\min} = r_{\max} = 0$ and define a state constraint referring to the two optimization variables $s$ and $Q_{out}$, the modeling component above and $N = 2$ time steps (note that the storage at $s^{k-1}$ and $s^k$ contributes to the mass balance equation).

Example 2:

Let's add a tailwater curve to the reservoir given by

$$tw^k = f(Q_{out}^k) \tag{3.8}$$

and define two constraints for i) keeping a minimum tailwater level and ii) limiting the maximum tailwater rate of change

$$\begin{aligned} &\text{i)} \quad tw^k \leq tw_{\min} \\ &\text{ii)} \quad \Delta tw_{\min} \leq tw^k - tw^{k-1} \leq \Delta tw_{\max} \end{aligned} \tag{3.9}$$

The implementation of the first constraint considers the variable $Q_{out}$, refers to the modeling components (3.8) and requires $N = 1$. The second constraint works in the same way except for the need for looking back an additional time step in time leading to $N = 2$.

Note that the Matlab interface provides a platform for defining more general constraints via a user programming.

### 3.3.3 Cost function terms

RTC-Tools supports the following cost function terms:

absolute (difference related to a set point)

$$J = w \sum_{k=1}^{T} \left| u^k - u_{sp} \right|^a \tag{3.10}$$

where $w$ is a weighting coefficient, $u_{sp}$ is a constant or time-dependent set point. The absolute term can be applied either on a control, state or dependent variable. The configuration supports the neglection of the upper branch ($u > u_{sp}$) or lower branch ($u < u_{sp}$) of the value range primarily for implementing a soft constraint on a variable up-crossing or down-crossing a threshold.

rate of change (difference of two subsequent values)

$$J = w \sum_{k=1}^{T} \left| u^k - u^{k-1} - u_{sp} \right|^a \tag{3.11}$$

where $u$ is either a control, state or dependent variable, $w$ is a weighting coefficient, and $u_{sp}$ is an optional set point for the rate of change. The latter is again primarily used within soft constraints in combination with the neglection of the upper branch ($u^k - u^{k-1} > u_{sp}$) or lower branch ($u^k - u^{k-1} < u_{sp}$) of the value range.

Furthermore, additional objective function terms of arbitrary types can be defined in Matlab.

## 3.4 Adjoint models

Gradient-based solvers of the Sequential Quadratic Programming (SQP) or Interior Point (IP) types require the gradient vector of the cost function $\mathrm{d}J(x,u)/\mathrm{d}u$ and the constraint Jacobian $\mathrm{d}h(x,u)/\mathrm{d}u$ for performing efficiently. The computation of these first-order derivatives by numerical differentiation is a straighforward approach, but requires at least $n+nnz+1$ model execution for an optimization problem of $n$ control variables and $nnz$ non-zero entries in the constraint Jacobian. It becomes computationally inefficient for several hundreds or thousands of dimensions, and disqualifies the approach from being applied in an operational setting.

A significantly more efficient method for computing the derivatives at computational costs in the order of a single model execution is the set-up of an adjoint model for each modeling component. The RTC-Tools framework takes care for integrating these models both in the simulation mode as well as the reverse adjoint mode.

The set-up of an adjoint model is outlined for the explicit version of the diffusive wave model presented in section 3.2.1. Consider the mass balance equation given by

$$s^k = s^{k-1} + \Delta t \sum_i f(s^{k-1}, s_i^{k-1}, dg_i^k) \tag{3.12}$$

where $s$ represent the storage at the node of interest and $s_i$ is the storage at a connected node $i$ and the function $f()$ denotes the flow contribution of a flow branch or hydraulic structure.

A straightforward way for the derivation of an adjoint model of an explicit calculation workflow is the application of algorithmic differentiation in reverse mode. It is basically a consequent

application of the chain rule leading to

$$\begin{aligned}
\widehat{s}^{k-1} &= \widehat{s}^{k} \left[ 1 + \Delta t \sum_i \frac{\partial f(s^{k-1}, s_i^{k-1}, dg_i^k)}{\partial s^{k-1}} \right] \\
\widehat{s}_i^{k-1} &= \widehat{s}^{k} \left[ 1 + \Delta t \frac{\partial f(s^{k-1}, s_i^{k-1}, dg_i^k)}{\partial s_i^{k-1}} \right] \\
\widehat{dg}^{k} &= \widehat{s}^{k} \left[ \Delta t \frac{\partial f(s^{k-1}, s_i^{k-1}, dg_i^k)}{\partial dg_i^k} \right]
\end{aligned} \tag{3.13}$$

where $\widehat{u}$ is the adjoint variable of $u$.

We compute the cost function gradient according to the following procedure:

1 Model simulation, (3.12), for computing all states and dependent variables
2 Initialization of the adjoint variables by the partial derivatives of the objective function, $\widehat{u}^k = \partial J(u^k, x^k, y^k)/\partial u^k$, with respect to control variables, states and dependent variables.
3 Model execution in adjoint mode, (3.13), in reverse order (with respect to the time loop and the execution of subsequent modeling components)

After conducting the steps above, the resulting adjoint variables represent the total derivatives of the cost function $\mathrm{d}J(u^k, x^k, y^k)/\mathrm{d}u^k$, i.e. the cost function gradient.

The computation of the constraint Jacobian is similar. The following procedure holds for a single constraint at a specific time step $k$:

1 ditto (once for all constraints)
2 Initialization of the adjoint variables by the partial derivatives of the constraint, $\widehat{u}^k = \partial h(u^k, x^k, y^k, d^k)/\partial u^k$, with respect to control variables, states and dependent variables.
3 Model execution in adjoint mode, (3.13), over $N$ time steps which contribute to the non-zero Jacobian entries of the specific constraint.

Adjoint systems are still not implemented for all available components in RTC Tool. An overview about the status of implementation is given in Table 3.2

***Table 3.2:*** *Implementation status of adjoint models*

| | **Adjoint available** | **Comments** |
|---|---|---|
| *Modeling components* | | |
| accumulation | yes | - |
| arma | yes | - |
| expression | yes | - |
| gradient | yes | - |
| hydraulicModel | yes | implementation is only available for the explicit scheme, implicit scheme will become available soon |
| hydrologicalModel | no | - |
| merger | yes | - |
| neuralNetwork | yes | - |
| reservoir | yes | - |
| reservoirBPA | yes | - |
| unitDelay | yes | - |
| unitHydrograph | yes | - |
| *Rules* | | |
| all | no | the adjoint of smooth rules such as the PID controller may become implemeneted in the future for modeling mixed systems with MPC and feedback control |
| *Triggers* | | |
| all | no | triggers switching on external input will be supported in the future |

# 4 References


Becker, B., 2013. *Inzet RTC-Tools voor het boezemmodel "Wetterskip Fryslân"*. Report 1205773-000, Deltares, Delft. In Dutch. 37

Becker, B. P. J., D. Schwanenberg, T. Schruff and M. Hatz, 2012. "Conjunctive real-time control and hydrodynamic modelling in application to Rhine River." In *Proceedings of 10th International Conference on Hydroinformatics*. TuTech Verlag TuTech Innovation GmbH, Hamburg, Germany. 37

Gregersen, J., P. Gijsbers and S. Westen, 2007. "OpenMI: Open modelling Interface." *Journal of Hydroinformatics* 9 (3): 175–191. 3

Schuurmans, J., 1997. *Control of Water Levels in Open-Channels*. Ph.D. thesis, Technische Universiteit Delft. 1, 2

Schwanenberg, D., B. Becker and T. Schruff, 2011. *SOBEK-Grobmodell des staugeregelten Oberrheins*. Report no. 1201242-000-ZWS-0014, Deltares. In German. 37

# A Configuration

## A.1 Model setup in XML

The RTC-Tools schematization is specified in a set of XML files according to Table A.1.

***Table A.1:*** *RTC-Tools configuration files*

| File | Content | Use |
|---|---|---|
| rtcDataConfig.xml | time series definitions, interface definitions for file io, in-memory data exchange etc. | required |
| rtcObjectiveConfig.xml | definition of the optimization problem including the control variables, constraints and cost function terms | optional |
| rtcParameterConfig.xml | set of externalized parameters for modification in external applications such as Delft-FEWS or Matlab | optional |
| rtcRuntimeConfig.xml | definition of runtime relevant info: time step size, simulation time, file names if deviating from standard naming convention, run mode (simulation, optimization etc.), logging information etc. | required |
| rtcScenarioTreeConfig.xml | definition of a scenario for the Tree-Based MPC option | optional |
| rtcToolsConfig.xml | RTC-Tools schematization including the modeling components, rules and controllers as well as triggers of the model | required |

All configuration files are expected in the same working directory. We highly recommend to validate all XML files against the corresponding XSD schema definitions during the model setup. We suggest using validating XML editors such as XMLSpy (http://www.altova.com/xml-editor/), XML Notepad (http://xmlnotepad.codeplex.com/) or oXygen (http://www.oxygenxml.com/).

The most efficient way to explore configuration options is to check the XSD schemas in an editor such as XMLSpy. Alternatively, the configuration options are documented in rtf/pdf format in the separate configuration manual.

## A.2 Initial conditions (state handling), boundary conditions

The primary file input and output formats of RTC-Tools for time series and model states are the Delft-FEWS PI XML formats (`pi_timeseries.xsd`, `pi_state.xsd`) and the OpenDA `treeVector.xsd` format.

The user is responsible to supply the initial condition of a model run in the file `state_import.xml` according to the OpenDA `treeVector.xsd` format. Note that no further meta data about date and time of the state is required, since the run period of the model is already defined in the runtime settings (check above). The `state_export.xml` is generated by RTC-Tools and includes the state of the last time step again in the OpenDA `treeVector.xsd` format and meta information for Delft-FEWS for picking it up in the `statePI.xml` file (described in `pi_state.xsd`).

Import and export time series comply with the PI-XML format `pi_timeseries.xsd`. Both version of the format are supported: i) pure XML, ii) a combination of XML and binary files (more efficient, but less readable). The time series of the PI-XML XML file are linked to the time series in the RTC-Tools (see file `rtcDataConfig.xml`) by the unique combination of locationId, parameterId and ensembleIndex.

### A.3 Command line options

RTC-Tools expects the XSD schemas in the directory of the binaries. Furthermore, it assumes its working directory in the folder from where it is executed. Both assumption can be overruled by the following command line options:

- ◇ `-schemaDir=<path to schema directory>`
- ◇ `-workDir=<path to work directory>`

### A.4 RTC-Tools in Delft-FEWS

We recommend the following setup of the file system for implementing RTC-Tools under Delft-FEWS:

<Delft-FEWS regional home>
    <Modules>
        <RTCTools>
            <bin> (including the executables and XSD schemas)
            `rtcDataConfig.xml` (configuration of time series and file io)
            `rtcRuntimeConfig.xml` (configuration of runtime settings)
            `rtcToolsConfig.xml` (configuration of the modeling components)
            `run.bat` (batch file for executing a model run)
            `state_import.xml`
            `timeseries_import.xml`

Note that the files above represent the minimum set of a RTC-Tools model.

The following hints summaries our recommended "best practice" for implementing RTC-Tools model in Delft-FEWS:

- ◇ Use a consistent naming convention between Delft-FEWS and RTC-Tools. We recommend to define the RTC-Tools Id by <locationId>_<parameterId>, where the location and parameter ids are the ones in Delft-FEWS.
- ◇ Try to avoid an additional the mapping in Delft-FEWS and use the one in the `rtcDataConfig.xml`.
- ◇ Make sure that you use the same units in Delft-FEWS and RTC-Tools. There will be no unit conversion.
- ◇ Put the configuration files `rtcXXXConfig.xml` into a module dataset.
- ◇ Start with a time series exchange by pure XML, then switch it to the XML/bin option later for efficiency.

### A.5 RTC-Tools in OpenMI

### A.5.1 Introduction

For conjunctive modelling of real-time control with hydraulic processes RTC-Tools is equipped with an interface according to the OpenMI standard. Applications of such conjunctive modelling have been described by Becker *et al.* (2012), Schwanenberg *et al.* (2011) and Becker (2013). Detailed technical information for an OpenMI composition consisting of RTC-Tools and SOBEK is given by Schwanenberg *et al.* (2011) and Becker (2013)

### A.5.2 The OMI-file

Below an example of an `*.omi`-file is given. The keyword `Assembly` refers to the location of the dynamic link library with the RTC-Tools computational core. This DLL must provide the OpenMI interface definition. `LinkableComponent` refers to `Deltares.RtcToolsWrapper.RtcToolsLinkableComponent`, this is hard-coded in the OpenMI interface definition. Table A.2 explains the meaning of the argument keys.

```
<?xml version="1.0"?>
<LinkableComponent Type="Deltares.RtcToolsWrapper.RtcToolsLinkableComponent"
  Assembly="..\..\RTCToolsOpenMI\bin\Deltares.RtcToolsWrapper.dll"
  xmlns="http://www.openmi.org/LinkableComponent.xsd">
        <Arguments>
      <Argument Key="modelDirectory" ReadOnly="true" Value=".\RtcTools" />
      <Argument Key="MissingValue" ReadOnly="true" Value="0" />
      <Argument Key="OpenMiTimeStepSkip" ReadOnly="true" Value="144" />
      <Argument Key="SchemaLocation" ReadOnly="true"
       Value="..\..\..\RTCTools\xsd\" />
        </Arguments>
</LinkableComponent>
```

***Table A.2:*** *OpenMI argument keys for RTC-Tools models*

| argument key | description |
|---|---|
| `modelDirectory` | Relative or absolute path from the location of the `*.omi`-file to the directory of RTC-Tools input file |
| `MissingValue` | If this value is provided to RTC-Tools via the `GetValues` method, RTC-Tools will treat it as not existent |
| `OpenMiTimeStepSkip` | 1 means data exchange at every RTC-Tools time step, 2 means data exchange every second time step, and so on. The default value is 1. |
| `SchemaLocation` | The relative path from the location of the RTC-Tools model directory to the directory with `xsd`-schema definition files. If this argument key is not given, RTC-Tools looks for the xsd files in the RTC-Tools work directory, where the xml files are located. |

### A.5.3 OpenMI exchange items

OpenMI exchange items are interpreted as an alternative to pre-defined time series, e. g. a Delft-FEWS-PI-time series in `timeseries_import.xml`. Consequently, OpenMI input exchange items and OpenMI output exchange items are defined in the file `rtcDataConfig.xml`.

In the example below the following exchange items are defined:

- input exchange items (import time series)
  - water level at Leeuwe_P498
  - water level at Bergum_P499
- output exchange item (export time series)
  - gemiddeldeBoezempeil

This RTC-Tools model runs connected with a SOBEK model, so the corresponding node in the SOBEK model schematisation has been added to the elementId of the input exchange items. In this example the RTC-Tools model gets time series data via OpenMI or from a Delft-FEWS-PI-time series from the file `timeseries_import.xml`. If RTC-Tools runs under OpenMI, priority is given to OpenMI, this means that data from OpenMI overwrites data from the Delft-FEWS-PI-time series file. If the model does not run under OpenMI, the data from the Delft-FEWS-PI-time series file is taken. So this RTC-Tools model can also run standalone, this is useful to identify bugs in the RTC-Tools-model.

Output data is both provided as OpenMI exchange item and written in the standard `*.csv`-output file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2012 rel. 2 sp1 (http://www.altova.com)
 by zws 2 (Stichting Deltares) -->
<rtcDataConfig xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
xmlns:rtc="http://www.wldelft.nl/fews"
xmlns="http://www.wldelft.nl/fews"
xsi:schemaLocation=
"http://www.wldelft.nl/fews ..\..\..\xsd\rtcDataConfig.xsd">
  <importSeries>
      <PITimeSeriesFile>
      <timeSeriesFile>timeseries_import.xml</timeSeriesFile>
      <useBinFile>false</useBinFile>
    </PITimeSeriesFile>
    <timeSeries id="Leeuwe_P498.H">
      <PITimeSeries>
        <locationId>Leeuwe_P498</locationId>
        <parameterId>H</parameterId>
        <interpolationOption>BLOCK</interpolationOption>
      </PITimeSeries>
      <OpenMIExchangeItem>
        <elementId>Leeuwe_P498</elementId>
        <quantityId>water level</quantityId>
        <unit>m</unit>
      </OpenMIExchangeItem>
    </timeSeries>
    <timeSeries id="Bergum_P499.H">
      <PITimeSeries>
        <locationId>Bergum_P499</locationId>
        <parameterId>H</parameterId>
        <interpolationOption>BLOCK</interpolationOption>
      </PITimeSeries>
      <OpenMIExchangeItem>
        <elementId>Bergum_P499</elementId>
        <quantityId>water level</quantityId>
```

```
        <unit>m</unit>
      </OpenMIExchangeItem>
    </timeSeries>
  </importSeries>
  <exportSeries>
    <CSVTimeSeriesFile/>
    <timeSeries id="gemiddeldeBoezempeil.out">
<OpenMIExchangeItem>
  <elementId>gemiddeldeBoezempeil</elementId>
  <quantityId>water level</quantityId>
  <unit>m</unit>
</OpenMIExchangeItem>
    </timeSeries>
  </exportSeries>
</rtcDataConfig>
```

# B Errors and unexpected results

## B.1 Time series from expression components used in triggers give unexpected results

### Error description

A standard trigger condition evaluates a time series provided by an expression model component (section 2.2.2). The trigger reacts unexpected.

### Reason

Triggers are always evaluated first in the RTC-Tools programme procedure. The result of the expression component time series is not available for the trigger.

### Possible solutions

- Use a trigger of type expression (section 2.5.7).
- Define the input time series explicitely: `<x1Series ref="EXPLICIT">`. The trigger will use the expression result from the previous time step.

## B.2 Values from an import time series do not appear in the result file

### Error description

Values of a time series from the `timeseries_import.xml` do not appear in the result file `timeseries_0000.csv`. The column header is present, but the column is empty.

### Possible reasons

- The time series is not referenced to in the file `rtcDataConfig.xml` under `<importSeries>`
- The begin and end time properties `<startDate>` and `<endDate>` in the time series header (`<header>`) do not match the simulation time properties given in `rtcRuntimeConfig.xml`.
- No extrapolation and interpolation options are given in `rtcDataConfig.xml`.
- A combination of `<locationId>` and `<parameterId>` is used twice in `rtcDataConfig.xml`.
- The simulation runs under OpenMI and the attribute `TimeSeries xsi:schemaLocation` directs to a location that is not valid, for example `..\..\..\badFolder\pi_timeseries.xsd`.

### Solution

- Check the list above and adjust the configuration accordingly.
- When using OpenMI, change the schema location path (attribute `TimeSeries xsi:schemaLocation`) in such a way that it points towards a valid location, by preference to the location of the binaries `..\..\..\bin\pi_timeseries.xsd`.

#### Error description

Some output variables are missing in the result file `timeseries_0000.csv`, although they are listed in the file `rtcDataConfig.xml` under `<exportSeries>`.

#### Possible reasons

- The export time series is defined twice, as import and as export time series.

#### Possible solutions

The limited memory option is switched on in `rtcRuntimeConfig.xml`. Set it to `false`:

```
<mode>
  <simulation>
    <limitedMemory>false</limitedMemory>
  </simulation>
</mode>
```

### B.3 Index not found in time series model

#### Error message

The file `diag.xml` says:

```
"int main(int argc, char *argv[]) - error -
int timeSeriesBasics::getScalarIndex(string s) -
index not found in time series model:  "
```

No name is given for the time series RTC-Tools can not find.

#### Reason

RTC-Tools looks for a time series with an empty name which it cannot find in `rtcDataConfig.xml`.

#### Possible solution

Look for incomplete items where RTC-Tools writes output for in the file `rtcToolsConfig.xml` and specify an output time series. Make sure that the time series names appear in the file `rtcDataConfig.xml` under <importSeries>. Do not specify a time series without name in `rtcDataConfig.xml`.

Examples for incomplete items are given below:

Example 1: the time series where the status of a standard trigger is to be written is empty:

```
<output>
  <status></status>
</output>
```

Example 2: the target time series of an expression is not specified:

```
<y></y>
```

Example 3: an input time series is not specified:

```
<x1Series ref="IMPLICIT"/>
```

## B.4 Instance document parsing failed

### Error message

The file `diag.xml` says:

```
error - instance document parsing failed"
level="1"
```

### Reason

A file is missing.

### Possible solution

Check if all files are available. Most likely the file `state_import.xml` is missing.

# C Programming in RTC-Tools (draft version)

## C.1 Preface

Contributions to RTC-Tools are welcome. The source code is available in an Apache Subversion repository (SVN repository) https://svn.oss.deltares.nl/repos/rtc-tools/. We recommend to use a subversion client, for example Tortoise SVN, to checkout the source code from the repository. To request read/write access contact the RTC-Tools product management[1]. We use Microsoft Visual Studio to compile RTC-Tools code. Microsoft Visual Studio projects are provided within the source code package.

We ask developers for the following:

- To make sure that the new feature fit fits well into the RTC-Tools architecture, we encourage developers to contact the principal developer[2] or RTC-Tools product management to discuss the ideas.
- Prepare a small test model that illustrates the new feature and add it to the examples in the svn-repository.
- RTC-Tools is provided in a dual license model:
  - as open source under the GNU public license version 2 (GPL2)
  - under a proprietary license.

  The dual license model is necessary to be able to provide features like commercial solvers or built-in interfaces to commercial software, which is conflictual to GPL2. Developers must transfer their intellectual property rights for the added code to Deltares, Deltares in return guarantees to provide RTC-Tools under GPL2 license.

## C.2 Implementing a new feature to RTC-Tools

Below the main working steps to implement new features to RTC-Tools are given.

- Modify the XML schema definition (XSD) file `*.xsd`. If you want to add a new hydrological model component `newModel` in `rtcToolsConfig.xml`, go to `rtcToolsConfig.xsd`.
  - Define the elements and attributes you need. In most cases they will be of ComplexType or SimpleType.
  - Specify the minimum and maximum number of occurrences (minOccurs, maxOccurs), where minOccurs = 0 means optional.
  - Maintain alphabetical order.
  - Add the newly defined elements to a feasible place in the main tree. Our hydrological model `newModel` should be added to the components group.
  - Validate the xsd file.
- Run the script `update_XSD.bat`. This script translates the XSD into C++ class files.
- Open the source code and go to the files that have been changed by the script `update_XSD.bat`.
- Create a .h-file `newModel.h` for the new class declaration. Add the variables that are needed to the class declaration.
- The class variables need to be fed with values from the file, in our case `rtcToolsConfig.xml`.
  - Open the `schematization.cpp`.

[1] `rtc-tools@deltares.nl`
[2] Dirk Schwanenberg (`dirk.schwanenberg@deltares.nl`)

- Add the file `newModel.h` to the list of include files. Maintain alphabetical order.
- Add code to read the data and assign it to the variables

◇ Create a file `newModel.cpp` and add the logic of the new feature.