RTC-Tools 2.0: An open source toolbox for control and multi-objective convex optimization of environmental systems under forecast uncertainty

J.H. Baayen[a,*], M. den Toom[a], P. Gijsbers[a,b], D. J. Vreeken[c], D. Schwanenberg[a,d]

[a]*Department of Operational Water Management, Deltares, P.O. Box 177, 2600 MH Delft, The Netherlands*
[b]*Department of Water Resources Management, Deltares, P.O. Box 177, 2600 MH Delft, The Netherlands*
[c]*Department of Industrial Hydraulics, Deltares, P.O. Box 177, 2600 MH Delft, The Netherlands*
[d]*Kisters AG, Pascalstraße 8+10, 52076 Aachen, Germany*

Abstract

A new open source toolbox for control and optimization of environmental systems is introduced. RTC-Tools 2.0 distinguishes itself from other environmental optimization packages by incorporating forecast uncertainty on the one hand, and supporting multiple nonlinear goals on the other. The latter is of direct practical relevance for hydro power dispatch, by removing the need for linear approximation of power generation. The convex, multi-objective formulation of hydropower optimization goals introduced in this work allows simultaneous and nonlinear variation of turbine flow and head to be accounted for without creating local minima. RTC-Tools 2.0 distinguishes itself from the earlier 1.x versions by a framework that enables the formulation of convex, i.e., unimodal and stable, optimization problems, by its use of the generic modelling language Modelica, as well as by introducing prioritized multi-objective optimization through sequential goal programming. Applications include hydro power dispatch, energy efficient polder drainage, and water allocation.

Keywords: environmental systems, water allocation, reservoir management, hydropower dispatch, model predictive control, multi-objective optimization, convex optimization, forecast uncertainty

1. Introduction

Environmental systems differ from those designed by man in the degree of control we exert over them. Environmental systems predate and probably will outlive human existence. These systems are, in that sense, a given. We may modify environmental systems by, for instance, adding human-built structures, or by modifying the geography of the surface of the land. At the same time, the system remains in a setting that remains more or less constant on the timescale of a human life. The geological and atmospheric dynamics acting on the system are mostly beyond human control, and our failure to predict these dynamics exactly results *forecast uncertainty.* Resource flows from neighbouring systems are also, from the perspective of the system under consideration, a given. The consequence is that the ability of humans to control environmental systems is *limited.*

*Corresponding author
Email address: `jorn.baayen@deltares.nl` (J.H. Baayen)

This stands in stark constrast to those systems that are designed by man. Machines, for example, are designed specifically for particular tasks. The behaviour of a machine remains unchanged, as long as the environment it exists in remains within the design space envisioned by the engineer. Furthermore, the machine is engineered with actuators and degrees of freedom that allow it to perform the tasks it is designed for exactly and *completely*.

Around the time of second millenium A.D., engineers started to experiment with ways to automate machines, leading to the invention of the centrifugal governor in the 17th century. In the early 20th century, this led to the invention of the PID controller, a way to automatically compute an actuator setting based on measurements of the system state [1]. In the 1980s, techniques from operations research started to be applied to the control of machines, leading to optimization-based or model predictive control [2].

In the second half of the 20th century, environmental engineers started to apply optimization-based planning and control techniques to environmental systems [3]. The principle of feedback control, i.e., the computation of actuator input based on the measured system state, could be applied directly to environmental systems. Weir levels, for example, could be controlled based on measured water levels. However, whereas for machines, a given target state can, generally speaking, always be achieved, this is not the case in environmental systems. A target water level might never be reached, due to insufficient inflow frow the upstream.

The significance of the inability to steer an environmental state to any desired value is apparent once we consider multiple system states. The degrees of freedom of a machine are often equipped with independent actuators, facilitating full and independent control authority over all states of the system. In environmental systems, however, states commonly influence each other. For instance, when considering a reservoir, the need to lower the water level when a flood wave is forecasted reduces the amount of water available for power generation. Control techniques designed for machines with independently actuated states fail to take this interdependence into account.

The interdependence among states implies that the fulfillment of one control goal may impact or prohibit the attainment of other control goals. This circumstance calls for an approach that is able to make trade-offs between control objectives. In turn, the ability to make trade-offs requires awareness of the interdependence of the states, i.e., a system model. The system behaviour may be linearized using an appropriate feedback law [4], energy functions may be used to derive control laws [5], or the system model may be endowed with multiple goal functions and constraints, leading to a multi-objective optimization problem [6, 7]. The use of system models and optimization to compute control input may be extended to predict and optimize for the future system state, leading to multi-objective model predictive control [8].

A second distinguishing feature of environmental systems is the uncertainty present when trying to forecast their evolution in time. System inflows, for example, may be predicted based on meteorological forecasts. The meteorological forecasts contain a high degree of uncertainty, but so do hydrological rainfall runoff models. Deterministic optimization with a single forecast would therefore run the risk of optimizing for one very specific future, and therefore be suboptimal when applied to reality as it unfolds. Optimization under forecast uncertainty calls for techniques that explicitly take the uncertainty into account: be it through forecast ensembles [9, 10, 11], or through a mathematical description of an uncertainty set and/or probability distribution [12, 13].

In this paper, we introduce RTC-Tools 2.0, an open source toolbox for multi-objective model predictive control of environmental systems under forecast uncertainty. RTC-Tools 2.0 is a reincarnation of the RTC-Tools 1.x package developed by Schwanenberg at al. [14], rebasing it on generic modelling principles, a new mathematical footing designed for operational stability, and adding multi-objective optimization using lexicographic goal programming.

We pose six requirements that an operational model predictive controller for an environmen-

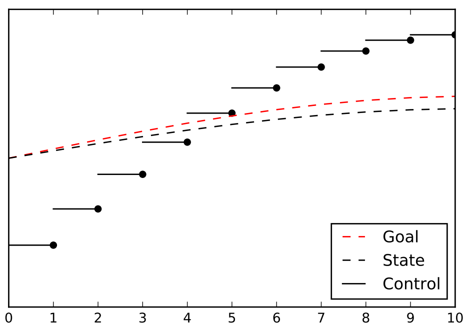


Figure 1: Model predictive control: Computing controls so that the system state reaches a prescribed target over a given prediction horizon.

tal system should satisfy. We will show that many of these requirements can be satisfied by making sure that the optimization problems satisfy certain mathematical properties, leading us to the field of convex optimization. We then review two multi-objective optimization techniques, exploring their relative pros and cons in the context of environmental systems. Subsequently, we review the stochastic optimization techniques available in the toolbox. In the final sections we discuss the model library, the ideas behind the software architecture and how to use it, and finally, a pilot application implemented for the Dutch Ministry of Infrastructure & Environment.

2. Model predictive control

Model predictive control (MPC) computes the control inputs for a system that maximize a given performance target over a predetermined time horizon. The optimal control inputs are computed using an embedded system model, forecasts of the boundary conditions, and optimization techniques.

A model predictive controller has the following components:

- An embedded predictive model of the system.
- A set of measurements to describe the current state of the system, acting as initial conditions for the predictive model.
- A set of boundary condition forecasts (e.g. water inflow to the systems, water extractions, energy costs), i.e. a future scenario.
- A set of variables singled out as control inputs, i.e., as decision variables.
- Goals (e.g. minimum and maximum water levels, cost minimization) to evaluate the system performance against.

The predictive model is discretized in time and forms, together with the goals, an optimization problem. A mathematical optimization algorithm then determines the optimal control inputs such that the model meets the goals as well as possible for the prediction horizon under the given scenario. See Figure 1.

Classical MPC systems optimize for a single performance target only, whereas RTC-Tools 2.0 is able to optimize for multiple, possibly competing, targets in a structured way using multi-objective optimization techniques. Multiple competing targets are common in water systems, where there is often too little or too much water to fully attend to all requests.

3. Reliability axioms

When using optimization in an operational context, it is essential that six conditions are always fulfilled:

1. *Accuracy*: Any solution is physically correct.
2. *Feasibility*: A feasible solution always exists.
3. *Quality*: Any solution is a "good" solution.
4. *Stability*: The solutions are *stable* in the sense that small perturbations in the configuration result in small changes in the solution.
5. *Determinism*: Given the same initial solution guess and configuration, the solution is always identical.
6. *Bounded solution time*: A solution is found within a predetermined amount of time.

The accuracy condition can be satisfied by ensuring that the physical models used for the optimization are validated for the entire control input search space. This hinges on the expertise of the modeller.

The feasibility condition can be satisfied by formulating operational constraints using prioritized multi-objective optimization using lexicographic goal programming. This will be discussed in the next section.

The quality and bounded solution time conditions may be fulfilled by ensuring that the optimization problems are *convex.* Convex optimization problems can be solved efficiently, and, in a convex setting, any locally optimal solution is also globally optimal [15]. Note that linear optimization problems, so called *linear programs*, are also convex. Yet the class of convex optimization problems is broader, allowing, for instance, quadratic as well as logarithmic constraints and objectives:

$$\text{Linear} \subset \text{Convex} \subset \text{Nonlinear}.$$

This allows RTC-Tools to operate as reliably as if it were solving linear programs, while allowing a broad class of nonlinear optimization problems to be solved without approximation.

The global optimality property of convex optimization problems also ensures that small perturbations in, e.g., the numerical settings of the optimization algorithm will not result in changes of the objective value. Furthermore, the use of lexicographic goal programming precludes the use of real-valued weighting factors in most cases, resulting in the elimination of most, and in most cases all, tuneable parameters in the optimization problem itself. This covers the stability condition.

The final condition is obtained by using deterministic methods of optimization. As convex optimization problems only have global optima, deterministic gradient-based algorithms may be used.

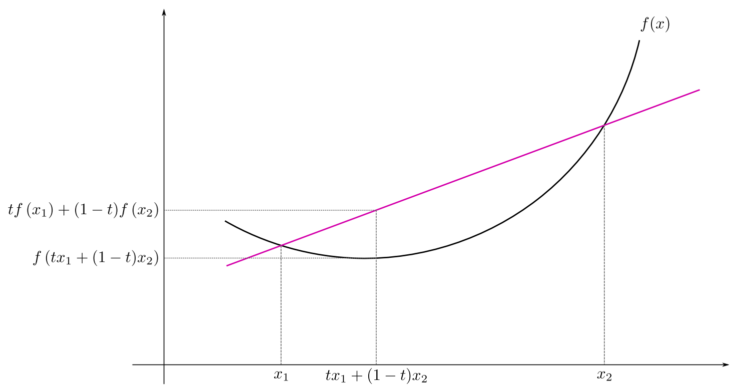


Figure 2: Example of a convex function. Source: Wikipedia. Licensed under the terms of the Creative Commons Attribution-ShareAlike license version 4.0.

3.1. Convexity and continuity

As discussed above, to ensure that our axioms are satisfied, we want to ensure the convexity of our optimization problems. In the present section, we introduce the notions of convex functions, convex sets, and convex optimization problems, and briefly touch upon continuity requirements.

3.1.1. Convex functions

A functions $f : X \to Y$ is convex on the set X whenever the following inequality holds:

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$

for all $x, y \in X$ and $t \in [0,1]$. See Figure 2. A function f is concave if the inequality is reversed.

Example 3.1. *The function $f = x^2$ is convex, whereas $f = -x^2$ is concave. The function $f = \log x$ is concave, but the function $f = -\log x$ is convex. Affine functions $f = ax + b$ are both convex and concave.*

3.1.2. Convex sets

A set X is convex whenever, for all $x, y \in X$ and $t \in [0,1]$, $tx + (1-t)y \in X$. In other words, whenever for each and every points $x, y \in X$ the straight line connecting x and y also lies in X. See Figure 3.

It is easily verified that the sublevel sets of a convex function f, i.e., the sets $\{x \in X : f(x) \leq y\}$ for $y \in f(X)$, are convex sets.

3.1.3. Convex optimization

An optimization problem is called convex whenever its objective function is convex, and its constraints result in a convex search space. In other words, the objective function must be well-behaved, and the search space must not contain any nooks or crannies.

Formally, an optimization problem of the form

$$\begin{aligned} \min_x f(x) & \qquad \text{subject to} \\ g(x) & \leq 0 \end{aligned}$$

is convex if $S := \{x : g(x) \leq 0\}$ is a convex set, and if f is convex on S.

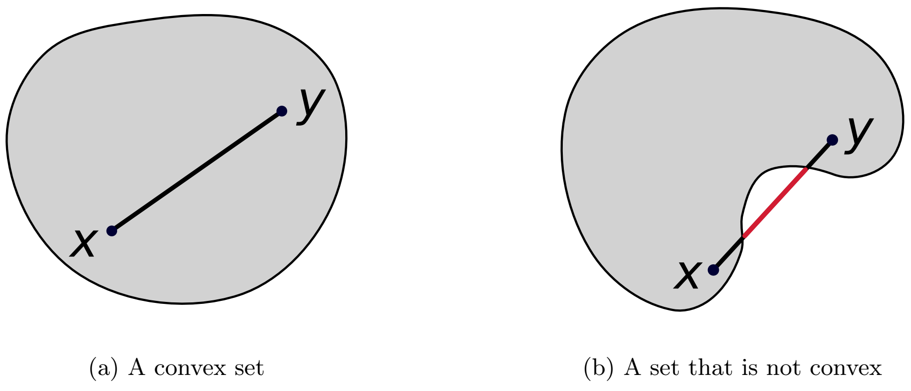


Figure 3: Examples of convex and non-convex sets. Source: Wikipedia. Licensed under the terms of the Creative Commons Attribution-ShareAlike license version 4.0.

A sufficient condition for convexity of the search space S is for the functions g to be convex. This condition is, however, not necessary. For example, the constraint $\log(x) \leq 0$ also defines a convex set in $\mathbb{R}^+$, but log is not a convex function[1].

In the present work, we will limit our attention to inequality constraints defined using convex functions. This is useful, as convexity is preserved under a number of algebraic operations, such as multiplication with positive scalars and addition: the set of convex functions forms a *cone*. This algebraic structure allows us to use knowledge of the convexity of elementary functions to construct more complex convex funtions.

To include equality constraints $h(x) = 0$, note that this is equivalent to demanding that $h(x) \leq 0$ and $-h(x) \leq 0$. The function h must therefore be both convex and concave at the same time, i.e., it must be affine.

3.1.4. Continuity

To satisfy gradient-based interior point solvers such as the open source IPOPT, the optimization problem must be twice continuously differentiable [16].

RTC-Tools 2.0 is designed to formulate convex and twice continuously differentiable optimization problems.

4. Multi-objective optimization

Typically, optimization problems are treated as having a single, well-defined goal function. Yet this model is not sufficiently rich to capture multiple, possibly conflicting, goal functions. Optimization problems with multiple goal functions are known as multi-objective problems. Characteristic of multi-objective optimization problems is the presence of a so-called *Pareto front*. The Pareto front is the space of solutions satisfying the constraints for which no goal can be improved without negatively impacting another [6]. Methods for multi-objective optimization problems seek to find one or more solutions on the Pareto front. See Figure 4. Solution on the Pareto front are known as *Pareto optimal*.

[1]It is, however, *quasiconvex*: All sublevel sets of the function are convex. The set of quasiconvex functions is however not closed under addition, and therefore quasiconvex functions are much more difficult to work with than convex functions.

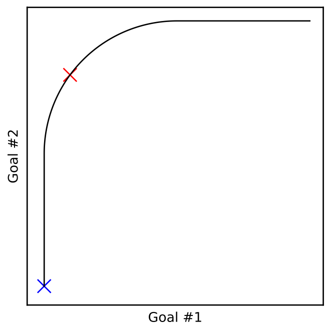


Figure 4: Pareto front with a typical weighting method solution in red, and a typical goal programming solution in blue.

In the following, we discuss two families of methods to parametrize the relative importance of the goal functions. We consider the translation of the multi-objective optimization problem into one or more single-objective optimization problems, yielding a single Pareto-optimal solution: We restrict our discussion to *a-priori methods* [6].

4.1. Weighting method

The idea of the weighting method is to assign real-valued weights, here denoted λ_i, to the goal functions, denoted f_i. The resulting single objective function becomes $f(x) := \sum_{i=1}^{N} \lambda_i f_i(x)$. Let the optimization problem be subject to a set of constraints $g(x) \leq 0$.

The resulting optimization problem is:

$$\begin{aligned} \min_x \sum_{i=1}^{N} \lambda_i f_i(x) & \qquad \text{subject to} \\ g(x) & \leq 0 \end{aligned} \tag{1}$$

The primary attraction of the weighting method is its simplicity and ease of implementation.

On the other hand, the optimal solution of the weighted optimization problem is a function of the weights. The implicition is that if there is a degree of arbitrariness in the choice of weighting factors, the location of the solution on the Pareto front likewise is arbitrary to a degree. An obvious remedy might seem to translate the different goal values into prices by means of the weighting factors. Oftentimes, however, it is hard or impossible to price a goal accurately.

In cases where the goals cannot be priced accurately, the meaning of the weighting factors becomes arbitrary, and designers resort to trial-and-error tuning of the factors. Such lack of clarity in the meaning of the weighting factors makes the weighting method opaque and hard to use for operators.

The weighting method has been applied to decision support for the short-term operation of hydropower resources [17, 18].

4.2. Lexicographic goal programming

The idea of lexicographical goal programming (LGP) is to optimize the goal funcions f_i in a given order, prioritizing earlier goals over later goals. Let us order the goals by assigning each a nonnegative integer priority value p_i. The goals are then solved in their priority order. Following the optimization of a goal f_i, its attainment level is fixed and added as a constraint to the optimization problem. The optimization of all following goals, in this way, will not worsen the attainment of any preceding goal. At each stage of LGP, optimization takes place within the degrees of freedom left open by the fixation of the attainment levels of the previous goals.

LGP results in a series of optimization problems. Let k be the priority level under consideration, and let the overall problem be constrained by the equation $g(x) \leq 0$. The k'th optimization problem is then

$$\begin{aligned} \min_x f_k(x) & \quad \text{subject to} \\ g(x) &\leq 0 \\ f_i(x) &= \varepsilon_i \quad \forall i < k, \end{aligned} \tag{2}$$

with the attainment level of the ith goal

$$\varepsilon_i := f_i(x_{opt,i})$$

and optimal solution of the ith optimization problem $x_{opt,i}$.

Contrary to the weighting method, here we see a direct relationship between the choice of priority level and the relative importance assigned to a particular goal. In LGP, higher priority goals take absolute precedence over lower priority goals. As a result, LGP solutions generally exists at extremal points of the Pareto front. See Figure 4 for an illustration of this phenomenon.

LGP, including its hybridized variant described below, has been applied to decision support for the short-term operation of hydropower resources [19, 20, 21, 22], to surface water allocation [23, 24], to water quality management [25, 26].

Finally, note that hybrid solutions are possible, where some goals in LGP are weighted sums of subgoals. The hybrid method is useful when some of the goals can be priced, and others can't. The priceable goals may be endowed with a single priority value, whereas the non-priceable goals are given other priorities.

4.2.1. Inequality goals

In water systems, one often encounters the need to keep variables within a desired range. A reservoir is a typical case, where one encounters desired lower and upper bounds for the water level. It may not always be possible to keep the water level within the desired range, as in case of drought or flooding. Inequality, or range, goals are therefore an important ingredient in a multi-objective optimization framework.

Let f_i a goal function, and let $[m_i, M_i]$ be its desired range, with $m_i \in \mathbb{R} \cup \{-\infty\}$, $M_i \in \mathbb{R} \cup \{\infty\}$, and $m_i \leq M_i$. Let k be the priority level under consideration, and let the overall problem be constrained by the equation $g(x) \leq 0$. The k'th optimization problem is then

$$\begin{aligned} \min_{x,\delta_k} \|\delta_k\|_p & \quad \text{subject to} \\ f_k(x) &\geq m_k + \delta_k(\underline{f}_k - m_k) \\ f_k(x) &\leq M_k + \delta_k(\overline{f}_k - M_k) \\ \delta_k &\geq 0 \\ \delta_k &\leq 1 \\ g(x) &\leq 0 \end{aligned}$$

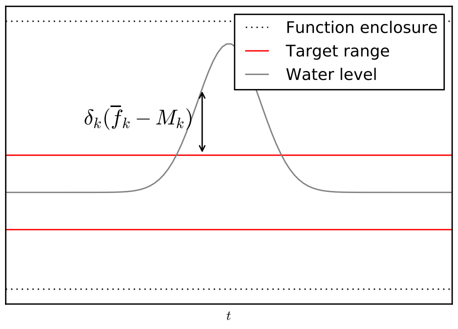


Figure 5: The notions of function enclosure, target range, and violation variable.

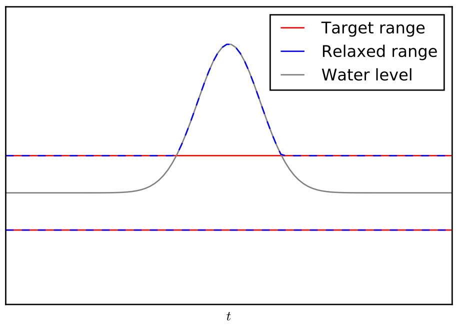


Figure 6: Constraint relaxation exaple.

with *violation variable* δ_k and goal function *enclosure* [27] $[\underline{f}_k, \overline{f}_k]$ such that $\underline{f}_k \leq f_k(x) \leq \overline{f}_k$ for all feasible x. The order $p \geq 1$ denotes the norm under consideration. Typically, one would select $p = 1$ or $p = 2$. The concept of violation variables is illustrated in Figure 5.

When $\delta_k = 1$ the goal function merely lies within its enclosure, whereas when $\delta_k = 0$ the goal is fully satisfied. One therefore tries to minimize the value of δ_k, starting from a feasible seed value of $\delta_k = 1$.

In addition, for every $i < k$, the following constraint is added to fix the goal attainment level:

$$m_i + \delta_i(\underline{f}_i - m_i) \leq f_i(x) \leq M_i + \delta_i(\overline{f}_i - M_i),$$

When applying an inequality goal for every discretized time instance along the prediction horizon, the effect of an inequality goal is best described as a *soft constraint*. First, the optimizer will try to find a state trajectory that lies within the desired range. All trajectories that lie within the range incur no penalty cost and are therefore equally preferable. If it is not possible to find a trajectory that fully lies within the desired range, the optimizer will select a trajectory that lies outside of it as little as possible. The desired range, relaxed just enough to accomodate the actual trajectory, is taken as a standard (hard) constraint for subsequent goals. This idea is illustrated in Figure 6.

LGP with inequality goals has been applied to decision support for the short-term operation of hydropower resources [19].

	Parametrization	Solver calls	Pricing
Weighting method	Real weights	1	Yes
Lexicographic goal programming	Integer priorities	n	No

Table 1: Overview of multi-objective optimization methods. n is the number of goals.

4.3. Comparison

In Table 1 we present an overview of the multi-objective optimization methods discussed in the previous section, indicating the type of trade-off parametrization of the method, the computational cost (in number of calls to the single-objective optimization solver), the location of the optimal solution on the Pareto front, the interactivity of the method, and whether or not the method can transparently handle non-priceable goals. Both methods are available in RTC-Tools 2.0.

5. Forecast uncertainty

In this section, we assume that the forecast uncertainty present in our boundary conditions is captured by an ensemble of predictions. Other ways of encoding uncertainty, such as the uniformly distributed parametric uncertainty [28], or ensemble dressing [29], are beyond the scope of this work.

In multi-stage stochastic optimization, we divide the optimization time horizon into stages. Conceptually, the time instance when a stage starts corresponds to a point in time when new information becomes available, i.e., when we can estimate the realized state of the system. The ability of the operator or control system to react to new information is modelled as a branching tree of control inputs, with every branching point corresponding to a transition between stages. Every forecast, i.e., every ensemble member, is associated with a – possibly different – sequence of control inputs, so that we end up simultaneously optimizing for all predicted futures with a set of controls that is partially shared between the ensemble members.

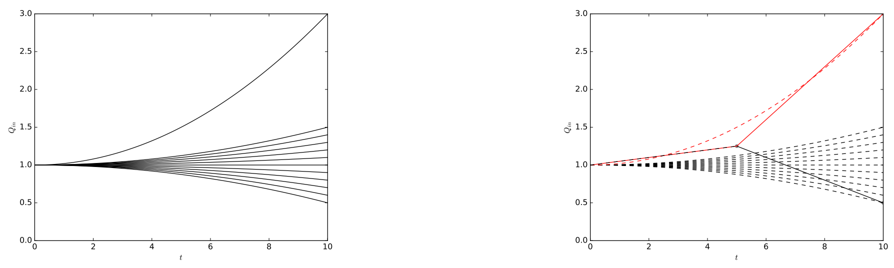
Yet a single *here-and-now* decision is required to implement the computed control strategy. This implies that, at the first discretization point, a single control input must be computed and applied to all ensemble members. At subsequent timesteps, we may recursively split the control trajectory into a number of new branches, assigning each ensemble member to a path through the tree. Every ensemble member consequently receives the control inputs associated to the branch segments that lie along the assigned path through the tree of control inputs.

There are many possible tree topologies, and many ways to assign ensemble members to paths through the tree. See [30, 31] for an overview.

Furthermore, it is also possible to reduce the size of the optimization problem by collapsing the ensemble itself into a tree structure [31, 9]. However, in this way, information on the pruned ensemble members is lost. The latter defect may be repaired by computing probability-weighted averages of clustered ensemble member segments. This, however, results in discontinuities between the segments, which may lead to unintended consequences when propagating these through the discretization of the system dynamics. In the present work, our aim is to present the basic method, and we therefore limit our attention to the original forecast ensemble.

We will furthermore limit our discussion to k-ary trees, as illustrated in Figure 7 for $k = 2$. At every branching point, we evaluate the distance between the ensemble member segments allocated to the branch preceding the branching point. We take the distance to be the 2-norm of the difference between the relevant ensemble member segments

$$d_{kl} = \|\{Q_{in}^k(t_i) - Q_{in}^l(t_i)\}_{i=i_0\dots i_1-1}\|$$



(a) A forecast ensemble.

(b) Stochastic tree.

Figure 7: A forecast ensemble, and the ensemble forecasts assigned to paths through a stochastic tree.

and the distance to all other ensemble members

$$D_k = \sum_{l \neq k} d_{kl}.$$

For $i \in \{1, \ldots, k\}$, we select the k such that D_k is largest, and assign it to the kth local branch. We then set D_k to a negative number, and continue until either the local braches, or the ensemble members, are exhausted. Any remaining ensemble members are clustured to these two ensemble members based on distance.

This yields the recursive algorithm 1. Let n be the number of time steps, m the number of branching points such that $m + 1 \mid n$, and k the number of branches per branching point. Number each branch locally with an integer number. We assign each ensemble member $k \in \mathcal{E}$ to a path through the tree, $b \in \mathbb{N}^m$. The set $\mathbb{N}^m$ is equipped with an operator $+ : (\mathbb{N}^m, \mathbb{N}) \mapsto \mathbb{N}^{m+1}$. Let B denote the t_0, the branching points, and t_N.

The above control tree discretization is implemented in RTC-Tools 2.0. It is, however, possible for the modeller to extend the discretization algorithm, or to write a plugin that also collapses the ensemble itself into a tree. We will return to the extensibility built into the software in subsequent sections.

6. Modelling framework

Following the design principle of separation of concerns, the RTC-Tools 2.0 modelling framework is independent from its optimization core. This allows application of the toolbox in different (environmental) applications by giving modellers flexibility in building new model libraries or extending existing ones, without having in-depth knowledge of numerical discretization and optimization techniques. Models are composed using the freely-available systems specification language Modelica [32], developed by the non-profit Modelica Association. Modelica is object-oriented, equation-based (acausal), and domain-neutral. As such it is suited for multi-domain modelling of large, complex systems. Various tools are available, both as open source product [33] and commerically, that can translate a Modelica model into an object suitable for simulation. For optimization, RTC-Tools relies on the JModelica.org compiler [34].

While Modelica is used extensively in industry, only few examples exist of its use in environmental applications [35, 36], where the use of tailor-made software is widespread. In many cases capturing Earth system dynamics requires consideration of a space-continuous system, de-

Algorithm 1 Stochastic tree generation.

```
function BRANCH($b$)
    $i_0 \leftarrow B(\dim b + 0)$
    $i_1 \leftarrow B(\dim b + 1)$
    $C_{ki} \leftarrow o + i \quad \forall i \in \{i_0, \ldots, i_1 - 1\} \quad \forall k, l \in b$
    $o \leftarrow o + i_1 - i_0$
    if $\dim b \geq m$ then
        return
    end if
    $d_{kl} \leftarrow \|\{Q^k_{in}(t_i) - Q^l_{in}(t_i)\}_{i=i_0 \ldots i_1 - 1}\| \quad \forall k, l \in B_b$
    $D_k \leftarrow \sum_{l \neq k} d_{kl}$
    for $i \in 1 \ldots k$ do
        $M_{b+i} \leftarrow \operatorname{argmax}_k D_k$
        $D_k \leftarrow -\infty$
    end for
    for $i \in 1 \ldots k$ do
        $B_{b+i} \leftarrow \{k\}_k \quad \forall k \quad$ such that $\quad d_{k,M_{b+i}} = \min_j d_{k,M_{b+j}}$
        BRANCH($b + i$)
    end for
end function
$o \leftarrow 0$                              ▷ Offset in the vector of optimization variables
$B_\emptyset \leftarrow \mathcal{E}$          ▷ Map of branch identifiers to ensemble member indices
BRANCH($\emptyset$)
```

scribed by a set of partial differential equations, for which Modelica is currently not well suited[2]. Especially in control applications, however, it is often possible to use simpler, lower-dimensional models that still satisfy the required accuracy at the temporal and spatial scales of interest. The `Deltares` model library, which is part of the RTC-Tools 2.0 distribution, provides a number of components that can be used in various water related control problems. The library is described in this section; specific applications are presented further down.

6.1. Deltares ChannelFlow

The `ChannelFlow` namespace (or *package* in Modelica terminology) groups together a number of classes for modeling one-dimensional free surface flow. It includes different types of nodes and links that can be used to be build a network of rivers and channels, as well as models for various types of controlled structures. The top-level structure of the `ChannelFlow` package is shown in Figure 8. The `Interfaces` and `Internal` packages contain supporting classes.

The `SimpleRouting` package consist of empirical-type models. Apart from water volume there are no conserved quantities, and flows are not explicitly related to potential differences. The implication is that water level is not part of the model formulation, but a diagnostic variable only. In Modelica the interface between components is formed by so-called *connectors*. `SimpleRouting` models employ two types of connectors, one marked as model *input* (`QInPort`) and the other marked as *output* (`QOutPort`). This means that the relation between components is causal: the flow at an output connector is not influenced by the component it is connected to.

[2]Various researchers have developed extensions to the Modelica language to make it capable of modelling certain classes of PDE. See, e.g., [37]. None of these features are currently available in mainline Modelica.

- Deltares
  - ChannelFlow
    - Interfaces
    - Internal
    - SimpleRouting

Figure 8: Top-level structure of the Deltares.ChannelFlow namespace.

The classes within the `SimpleRouting` package are structured as follows:

- `BoundaryConditions`: Package containing the classes `Inflow` and `Terminal` for specifying an (upstream) inflow boundary condition, and marking a (downstream) location where water exits the network, respectively.
- `Branches`: Several classes for flow routing, where
  - `Steady` implements steady state routing, i.e. outflow is set equal to inflow;
  - `Delay` is used for modeling a constant travel time along branches; and
  - `Muskingum` implements the linear Muskingum routing model.
- `Nodes`: The class `Node` is used for modeling flow confluences, bifurcations, or combinations thereof. Instances of `Node` may have one or more `QInPorts`, the flows of which are summed and distributed over one or more `QOutPorts` according to an externally provided distribution (e.g. from the optimizer).
- `Reservoir`: Contains the `Reservoir` class which implements the volume (V) conservation equation, supporting a distinction between turbine flow Q_T and spill flow Q_S for releases from the reservoir. Denoting inflow by Q_I the governing equation is given by

  $$\frac{\mathrm{d}V}{\mathrm{d}t} = Q_I - Q_T - Q_S. \tag{3}$$

  Note that Equation 3 is implemented in Modelica in its continuous form. Time discretization is carried out by RTC-Tools, as will be discussed in the next section.

  The power output P of a hydropower reservoir depends in a non-linear way on V and $Q_{T,S}$. By the discussion of convexity above, the computation of power can therefore not be included as an equality constraint in our optimization problem, and is therefore left out of the Modelica class. As will be discussed below, it is still possible to include power in the optimization objectives.
- `Storage`: The class `Storage` is similar to `Reservoir`, but with a single variable for flow release. It is intended to be used in combination with `Branches.Delay` to form an Integrator-Delay model.
- `Structures`: Contains `Pump`, which implements controlled flow across a structure.

6.2. Water levels

The relationship between water level and storage in a reservoir or a channel reach is typically nonlinear, and we therefore do not want to include such relations into the optimization model as equality constraints.

However, storage is a strictly increasing function of water level, and therefore the relation is invertible.

By using the level/storage relation to transform target water levels into equivalent storage values prior to running an optimization, the optimization problem may be formulated in terms of volumes only. Since bookkeeping of volumes is linear in the flow variables (see Equation 3), we obtain a fully linear model without sacrificing accuracy.

After the optimization has run, the volumes may be converted back to water levels using the inverse of the level-storage relation.

6.3. Hydropower generation

When modelling a reservoir system including hydropower generation, we need a model to relate turbine flow and head to power generation. The instantaneous power generation P of a hydroelectric turbine is given by

$$P = \eta \rho g Q \Delta H, \tag{4}$$

with turbine efficiency η, water density ρ, gravitational constant g, turbine flow Q and head difference $\Delta H = H_u - H_d$ with upstream water level H_u and downstream water level H_d.

The power generation P, viewed as function of the turbine flow Q and head ΔH, is a nonlinear function that is neither convex nor concave[3]. Direct inclusion of Equation 4 into an optimization problem would therefore render the problem nonconvex.

However, our interest in the power generation is typically related to an optimization goal: We either aim to track a generation request over time, or we want to maximize generation revenue, taking into account fluctuations of the energy price.

With an appropriate variable transformation, we can reformulate either of these goals as convex, multi-objective optimization problems. In the following, we will show how this can be done for tracking generation requests. Revenue maximization is left as an exercise to the reader.

The transformation hinges on the use of logarithms to replace the multiplication with an addition, to obtain the intermediate equation

$$\log P = \log(\eta \rho g) + \log(Q) + \log(\Delta H).$$

This equation is nonlinear in Q and ΔH. However, since log is a concave function and P_t is fixed, the constraint

$$\log P_t \leq \log(\eta \rho g) + \log(Q) + \log(\Delta H) \tag{5}$$

is convex. Generation request tracking is obtained by embedding the logarithmic constraint 5 in a goal programming context:

G1: $\log P_t \leq \log(\eta \rho g) + \log(Q) + \log(\Delta H)$,

G2: $\min Q$.

[3]To see this, note that a function is convex if and only if its Hessian is positive semidefinite on the interior of the convex set [15], and that the Hessian matrix of a product of two variables is indefinite.

The second priority goal, *G2*, makes sure that the power production is no higher than necessary by minimizing release flows. In some cases, this goal may further reduce Q by increasing ΔH. The latter is considered beneficial as it conserves water inside the reservoirs. Different behaviour may be requested by obtaining additional goals in the optimization problem.

In a reservoir management context, one would typically add more goals to ensure that reservoir water levels remain within the desired bounds.

Note that ΔH may be a concave function of the turbine flow Q and the reservoir volume, V. If ΔH is concave, then so is $\log(\Delta H)$ as log is a non-decreasing concave function. For example, a storage volume-water level relation is typically concave and may be used in the definition of ΔH.

This can be extended to multi-reservoir systems as follows. Introduce additional optimization variables l_i, and the constraints

$$\sum_i e^{l_i} \leq P_t$$

and

$$l_i \leq \log(\eta \rho g) + \log(Q_i) + \log(\Delta H_i).$$

In order to ensure that as much as possible of P_t is realized, we first maximize the l_i. Then, to make sure that no more than P_t is realized, we minimize the turbine flows:

G1: $\max \sum_i l_i$,

G2: $\min \sum_i Q_i$.

It can be shown that the optimum solution of *G1* is an equal distribution of generation targets across the reservoirs, if this is feasible. If some reservoirs are to be preferred over others, preference may be expressed by splitting *G1* into individual goals for each reservoir.

Finally, note that this a non-linear, albeit convex, treatment of turbine power generation. Unlike in packages such as Riverware [38], no linearization has been performed and consequently, head variations over the prediction horizon are fully taken into account.

7. Optimization framework

In an operational setting, the physical system model typically is set up once, is validated, and is then used time and again. But the goals and objectives of an optimization problem are more dynamic in nature. Depending on the situation they are facing, operators may decide to modify, include, or exclude particular goals and constraints.

Whereas there are extensions to the Modelica language that give the modeller the ability to specify an objective funtion and constraints, such as Optimica [39], these models would need to be heavily parametrized to be sufficiently customizable from within an operational user interface.

Instead, RTC-Tools 2.0 leverages the existing general purpose programming language Python to define optimization problems. The general approach is that each optimization problem is a Python class that inherits from a set of base feature classes defined by RTC-Tools. See Figure 9. These classes do all the heavy lifting in discretizing the model and importing the data from a certain file format. See Figure 10. To choose between the various types of data input, or between the different approaches in multi-objective optimization, the modeller selects a particular set of feature classes. To complete the problem definition Python class, the user is then left to specify the remaining required methods, e.g. for the objective or list of goals.

In many cases the standard feature classes are sufficient, but the power and flexibility of using a general purpose language like Python lies in the freedom it offers. One can change or extend

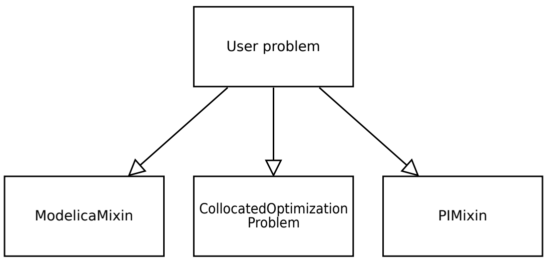


Figure 9: Inheritance structure of a typical optimization problem.

all of RTC-Tools's default behavior just by overriding methods in the problem definition, or by developing additional features in new feature classes. Feature classes are known as *mixins* in RTC-Tools.

7.1. Data input and output

As discussed previously, environmental models typically require boundary condition forecasts of run. For complex systems, the forcing variables may number in the hundreds. Furthermore, continuously getting all required data into an operational system can be challenging. Not only is there the aggregation of data from different sources, but there is also interpolation and extrapolation of missing data, as well as data validity checking. Delft-FEWS is an open platform, developed at Deltares, that facilitates these tasks [40]. RTC-Tools 2.0 can run as an adapter inside a Delft-FEWS configuration, with data I/O using the Delft-FEWS Published Interface (PI) file format.

For stand-alone applications, CSV files may be used for data I/O. Support for data I/O with CSV files is also part of the standard RTC-Tools package.

7.2. Goal programming

As described in Section 4.2, it can be desireable to optimize a problem using a prioritized list of goals instead of specifying a single (weighted) objective and constraints. In RTC-Tools, lexicographical goal programming is available by using the `GoalProgrammingMixin`.

The `GoalProgrammingMixin` takes care of translating a list of `Goal` objects to lists of objectives and constraints, ordered by priority. The optimization problem is solved for each priority, starting with the highest priority goals. The solution at after solving for a certain priority is remembered, and used as a constraint for the next priority. In this way, solving for a lower priority cannot make the solution for a higher priority worse.

Every `Goal` object consists of a priority attribute, a function definition, and target minimum and maximum values for this function. It is also possible to specify only a minimum or maximum target value, or no target value at all. In the latter case, the definined function is minimized for in the optimization.

7.3. Software implementation

RTC-Tools 2.0 uses the open-source JModelica.org compiler [34] to compile Modelica models into a differential-algebraic equation (DAE) [41] representation in CasADi [42], an open-source package for symbolic computation and algorithmic differentation. It subsequently discretizes the DAE in time using a combination of integration and/or collocation. Model parameters, initial conditions, and boundary condition forecasts are read from any of the various supported file formats and inserted into the optimization problem. The final, discretized optimal control problem is then interfaced with IPOPT [16] for continuous problems, and with BONMIN [43] for mixed-integer problems. The full chain is summarized in Figure 10.

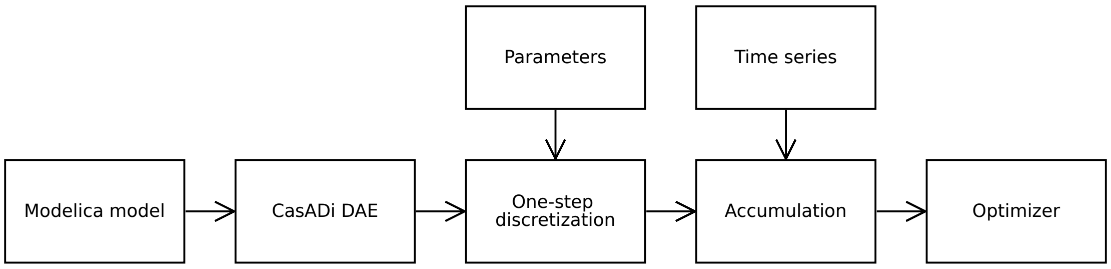
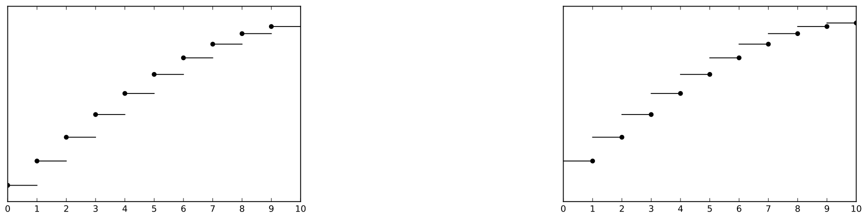


Figure 10: Process chain.



(a) $\theta = 0$ (b) $\theta = 1$

Figure 11: Influence of control inputs for different values of θ.

7.3.1. Time discretization

RTC-Tools discretizes differential equations of the form

$$\dot{x} = f(x, u)$$

using the θ-method

$$x_{i+1} = x_i + \Delta t \left[\theta f(x_{i+1}, u_{i+1}) + (1 - \theta) f(x_i, u_i)\right] \tag{6}$$

The default is $\theta = 1$, resulting in the implicit or backward Euler method. In this case, the control input at the initial time step is not used.

By default, the discretized differential equations 6 are included as collocation constraints in the optimization problem. Optionally, it is possible to mark certain states to be integrated. In this case, the equations 6 are solved using a rootfinding method, and the derivatives propagated using the implicit function theorem.

Note that depending on the value of θ, the control inputs used to compute the trajectory between two subsequent time steps are either sourced from the first step ($\theta = 0$), the second step ($\theta = 1$), or both ($0 < \theta < 1$). For $\theta = 0$, this means that the control input computed for the very last time step of the prediction horizon is not used to compute the dynamics. Similarly, for $\theta = 1$, the control input computed for the very first time step has no influence on system dynamics. In the latter case, the input at the first time step may be provided as historical data, so that smoothing goals may be used to ensure that newly computed controls transition smoothly from the currently implemented control. See Figure 11.

7.3.2. Multirate dynamics

In RTC-Tools, all states, control inputs, and external time series may be provided at arbitrary resolution. An internal interpolation layer makes sure that interpolation is used automatically whenever necessary.

7.3.3. Scaling

To ensure that optimization problems converge reliabily and quickly, proper scaling of variables and constraints is essential [16]. RTC-Tools reads the `nominal` attribute on variables in Modelica models, and uses this to perform a change of variables

$$n_x \cdot x_{scaled} = x,$$

with original variable x and nominal value $n_x \in \mathbb{R}^+$. In this way, scaling is taken care of explicitly at the model level.

7.3.4. Derivatives

Gradient-based optimization algorithms such as IPOPT require estimates of the derivatives of the objective and constraint functions. Interior point methods such as IPOPT also benefit from second order derivatives [16].

Derivative information could be obtained using finite difference approximation. However, this is computationally expensive, and does not give exact results for nonlinear functions. In order to be able to obtain first and second order derivative information efficiently for any problem, we use the algorithmic differentation package CasADi [42]. CasADi facilitates the construction of symbolic expressions as graphs, and then uses the rules of calculus to produce new symbolic graphs representing the derivatives. These symbolic graphs may then be evaluated with numerical inputs directly, or compiled and then evaluated.

7.3.5. Solvers

CasADi also supports interfacing its symbolic expressions with a variety of optimization packages through a generic interface. By default, RTC-Tools 2.0 uses the IPOPT [16] interface for continuous problems, and the BONMIN [43] interface for mixed-integer problems. However, RTC-Tools can also be configured to use other interfaces supported by CasADi, for instance to access commercial solvers.

7.4. Availability

RTC-Tools 2.0 is available online under the terms of the GNU General Public License version 3 [44]. Precompiled binaries, source code, and documentation are available from the project website at `https://www.deltares.nl/en/software/rtc-tools/`.

8. Applications

While RTC-Tools 1.x has numerous applications in the field of reservoir optimization for hydropower generation [14] and flood control in polder systems e.g. [45], the new capabilities of RTC-Tools 2.0 allow the field of application to be expanded. Water allocation among competing sectors has been the first. Within a river basin, surface water fulfills services for amongst others the public water sector, irrigated agriculture, power generation, recreation and navigation. At the same time environmental conservation is in demand for water as well, while flood wave attenuation may be desired to preserve properties at stake. Upstream uses of water influences availability for downstream uses both in timing, volume and quality. Some users extract water from the river system, and discharge it elsewhere, possibly to another water system. Other uses such as recreation, navigation and hydropower generation leave the water in the river system. Each functional use has its own demand pattern in volume, timing and location. Water policies and legal requirements generally determine which water users will have the first priority to the water. Within the Netherlands the concept of the Verdringingsreeks has been introduced in the

Water Law (art.2.9) with the actual order of priority defined in the *Waterbesluit* (art.2.1)[46]. Highest priority is given to prevention on irreplaceable damage to infrastructure, land subsidence and nature. Second priority is given to public utilities for drinking water and energy supply. Third priority is given to small scale uses with high economic value and fourth priority is given to limiting societal and economic consequences for navigation, agriculture, nature, industries, recreation, fisheries and other uses.

8.1. Pilot: Water allocation in The Netherlands

Within the Netherlands, the National Coordination Commission for Water Allocation is tasked to implement the *Waterbesluit* and make operational decisions on water allocation during periods of drought. Generally decisions focus on trade-off between priority 4 water uses. Since 2009, an important information source for the commission is *RWsOS-Waterbeheer* [47]. RWsOS-Waterbeheer is a real-time operational forecast system of Rijkswaterstaat which produces on daily basis a 10-day ahead forecast for the entire Dutch water system using the *National Hydrological Model* [48]. This National Hydrological Model is a detailed integrated surface-groundwater system of the Netherlands, combining a 250x250m grid model for groundwater (Modflow) and unsaturated zone (Metaswap) with a 8800 network elements based surface water balance and allocation model. This water allocation model uses a heuristic approach to allocate water to various prioritized purposes following the legal Verdringingsreeks. While the model provides useful insight in the current state of the system and the expected situation in the coming 10 days, it's runtimes hinders the commission to conduct multiple what-if analysis to assess trade-offs between different water allocation patterns.

Hence the need arose for a so-called Quick Scan Tool, a tool that can be used for screening various allocation strategies and assessing its regional and sectoral impacts. This Quick Scan Tool (QST) has been developed with RTC-Tools 2.0 as the core model engine. The model underlying this tool is composed of a coarse network of the Dutch water system including the major water storages, water distribution points and delivery routes to the various uses. See Figure 12. The model is fed with initial conditions for the storages, timeseries of forcings (discharge) on the upstream boundaries, and timeseries of soft constraints at each network element representing physical limitations in channel flow and pool level capacities and requests for discharges or abstractions. All time series are prioritized as illustrated in the following table.

As can be noted, the physical limitations have been assigned higher priority over the societal requests for water. The physical limitations are treated as critical goals meaning that an error will be given if these goals cannot be met. While crafting this priority table, specific attention is needed to define the maximum flow constraints such that water follows the rivers and only enters the canal systems when this is desired by a functional water request. For this purpose, priority 10 is introduced to cap the outflow of canal outlets in case the outlet does not act as a primary delivery path for downstream elements.

Also note that the soft constraints with lower priority need to shrink the search space as compared to high priority constraints. Therefore the soft constraint for the instream flow goal P7 is based on the highest value of goal P6 and the flushing request:

$$P7\_QOut = \max\{P6\_QOut,\, QFlush\}.$$

For lateral flow goals, each new lateral request (e.g. for utility purposes) needs to be added to the previous lateral goal (e.g. P1).

$$P5\_QLat = P1\_QLat + QLat\_utility.$$

Within the Quick Scan Tool the forecast of RWsOS-Waterbeheer is used as input, where the high-resolution data is spatially aggregated to obtain abstraction and instream flow requests,

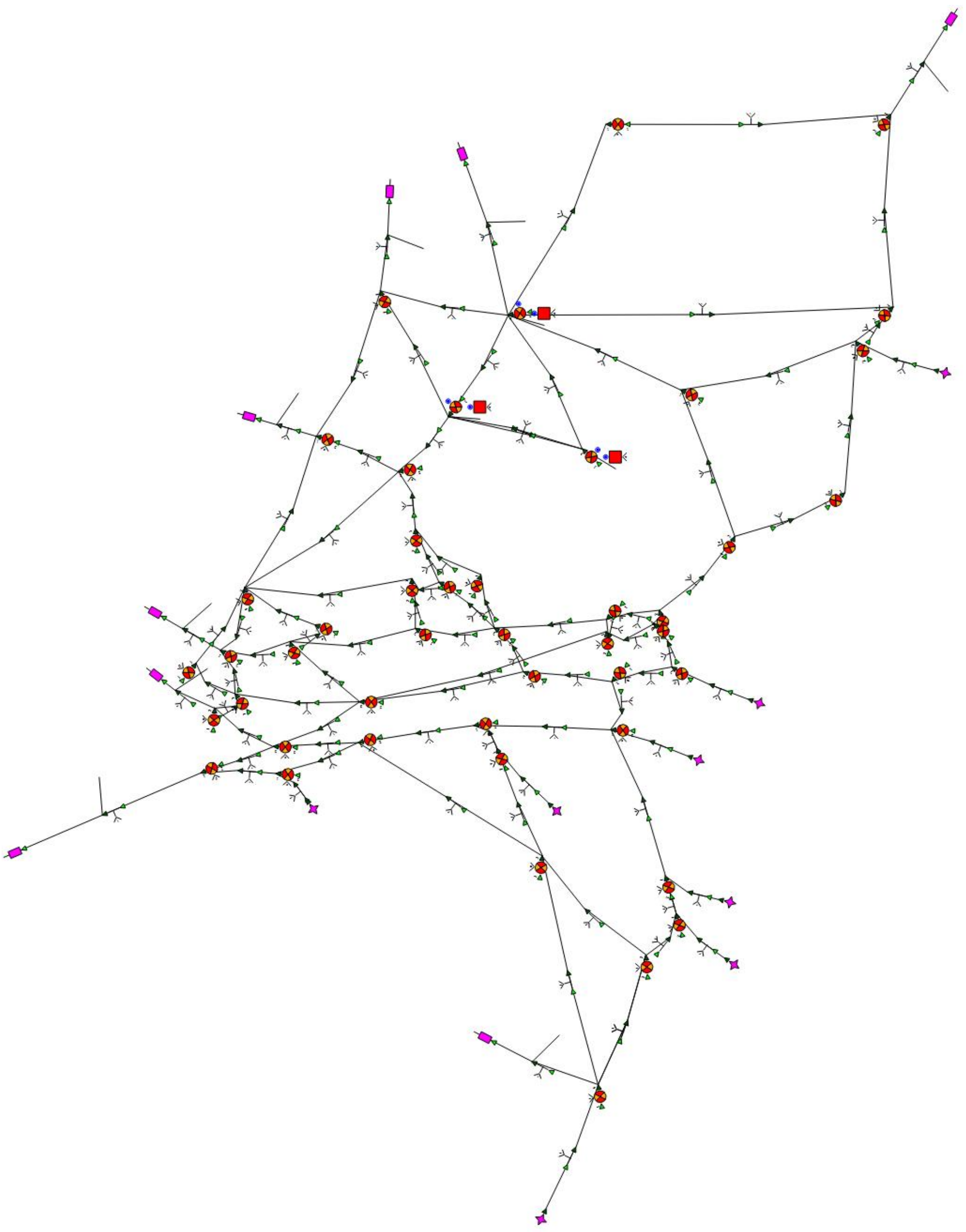


Figure 12: Network schematization for the Quick Scan Tool. Red blocks indicate storages, pink rectangles terminals, and pink stars inflows.

Purpose	Priority	ModelicaId	MinConstraintId	MinConstraintId
NaturalLoss	1	<elementid>:QLat	<elementid>:P1_QLat	<elementid>:P9_QLat
PhysicalMin/Max Q	2	<elementid>:QIn	<elementid>:P2_QIn_PhysicalMin	<elementid>:P2_QIn_PhysicalMax
PhysicalMin/Max H	3	<elementid>:H	<elementid>:P3_H_PhysicalMin	<elementid>:P3_H_PhysicalMin
Prevent damage	4	<elementid>:H	<elementid>:P4_H_TargetMin	<elementid>:P7_H_TargetMax
Utilities	5	<elementid>:QLat	<elementid>:P5_QLat	-
NavigationLocks	6	<elementid>:QOut	<elementid>:P6_QOut	-
Flushing	7	<elementid>:QOut	<elementid>:P7_QOut	-
Greenhouses	8	<elementid>:QLat	<elementid>:P8_QLat	-
Irrigation	9	<elementid>:QLat	<elementid>:P9_QLat	-
(cap)	10	<elementid>:QOut	-	<elementid>:P8_QOut[a]

Table 2: Generalizaed version of priorities table underlying water allocation model for Quick scan Tool

[a]Limited set of locations only.

subdivided by functional purpose, at the QST network. As discussed the functional request are ordered by priority and where needed added together, to obtain total abstraction requests, or maximized to obtain the overall request for instream flows. The QST user can manipulate the time series requests using a simple multiplier per functional purpose. In addition to flow and abstraction requests, the model also copes with water level requests for the various storages.

One of the challenges was to address a non-convex water diversion function which represents a diversion governed by open channel hydraulics as well as weir manipulation in one of the downstream branches. Diversion rules are available to subdivide the inflow at the diversion node in two outflows based on the flow at the model boundary. Using these rules and the boundary flow, a time series of diversion fractions is derived which the model uses to split to flow coming out of the first branch.

Once the sequential goal programming has completed its computation, the resulting flows need to be split according to the different purposes, such that flow delivery rates by purpose can be derived and presented. The results of the model meet the expectations, although a more detailed validation is needed to identify whether more stringent solution constraints or an adjusted objective penalty is desired.

9. Conclusion

In this paper we presented RTC-Tools 2.0, a new open source framework for the control and optimization of environmental systems. We started out by posing six axioms that capture the requirements posed by an operational system. Departing from the axioms, we selected convex optimization as our framework, thereby striking a balance between nonlinear accuracy and stability. Within the framework of convex optimization, we presented lexicographic goal programming as the natural approach to multi-objective opimization of water systems. Multi-stage stochastic optimization was selected as our framework for considering forecast uncertainty. We also presented a generic, extensible model library using the standardized modelling language Modelica. As a further contribution, we demonstrated how multi-objective optimization may be used to derive fully convex, yet exact, formulations for hydropower optimization problems, taking into account simultaneous variations in turbine flow and head. In the final section, we discussed a pilot application developed for the Dutch Ministry of Infrastructure & Environment.

10. Acknowledgements

We are grateful for the support of the Operational Water Management and Industrial Hydraulics departments at Deltares. In particular, we would like to mention Jesse vanderWees for developing B-Spline fitting routines with convexity and monotonicity constraints, for many bug fixes, as well as for his tireless work on writing documentation and examples, Olav van Duin for his work on robust I/O of CSV and Delft-FEWS PI files, Tiaravanni Hermawan for months of early testing, Bernhard Becker for writing a tutorial, and Joris Gillis of Yacoda/KU Leuven for teaching us how to use CasADi most efficiently. TKI Delta Technology provided a part of the funding under project DEL010.

References

[1] S. Bennett, A brief history of automatic control, Control Systems, IEEE 16 (3) (1996) 17–25. doi:10.1109/37.506394.

[2] C. E. García, D. M. Prett, M. Morari, Model predictive control: Theory and practice-A survey, Automatica 25 (3) (1989) 335–348. doi:10.1016/0005-1098(89)90002-2.

[3] P.-J. Van Overloop, Model predictive control on open water systems, IOS Press, 2006.

[4] H. K. Khalil, Nonlinear Systems, Third Edition, Prentice Hall, 2002. doi:10.1016/j.physa.2006.08.011.

[5] R. Kalman, J. Bertram, Control system analysis and design via the second method of Lyapunov: (I) continuous-time systems, IRE Transactions on Automatic Control 4 (3) (1959) 394–400. doi:10.1109/TAC.1959.1104895.

[6] Y. Collette, P. Siarry, Multiobjective optimization: principles and case studies, Springer, 2003. doi:10.1007/978-3-662-08883-8.

[7] R. T. Marler, J. S. Arora, Survey of multi-objective optimization methods for engineering (2004). doi:10.1007/s00158-003-0368-6.

[8] W. Wojsznis, A. Mehta, P. Wojsznis, D. Thiele, T. Blevins, Multi-objective optimization for model predictive control, ISA Transactions 46 (3) (2007) 351–361. doi:10.1016/j.isatra.2006.10.002.

[9] L. Raso, D. Schwanenberg, N. C. van de Giesen, P. J. van Overloop, Short-term optimal operation of water systems using ensemble forecasts, Advances in Water Resources 71 (2014) 200–208. doi:10.1016/j.advwatres.2014.06.009.

[10] D. Schwanenberg, F. M. Fan, S. Naumann, J. I. Kuwajima, R. A. Montero, A. Assis dos Reis, Short-Term Reservoir Optimization for Flood Mitigation under Meteorological and Hydrological Forecast Uncertainty: Application to the Três Marias Reservoir in Brazil, Water Resources Management 29 (5) (2015) 1635–1651. doi:10.1007/s11269-014-0899-1.

[11] S. Naumann, D. Schwanenberg, D. Karimanzira, F. Fan, C. Allen, Short-term management of hydropower reservoirs under meteorological uncertainty by means of multi-stage optimization, At-Automatisierungstechnik 63 (7) (2015) 535–542. doi:10.1515/auto-2014-1168.

[12] S. V. Braaten, O. Gjønnes, K. Hjertvik, S.-E. Fleten, Linear Decision Rules for Seasonal Hydropower Planning: Modelling Considerations, Energy Procedia 87 (2016) 28–35. doi:10.1016/j.egypro.2015.12.354.

[13] C. Gauvin, E. Delage, M. Gendreau, A robust optimization model for the risk averse reservoir management problem, Tech. rep., GERAD HEC Montréal (2015).

[14] D. Schwanenberg, B. P. J. Becker, M. Xu, The open real-time control (RTC)-Tools software framework for modeling RTC in water resources sytems, Journal of Hydroinformatics 17 (1) (2015) 130. doi:10.2166/hydro.2014.046.

[15] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004. arXiv:1111.6189v1, doi:10.1109/TMI.2010.2080282.

[16] A. Wächter, L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Mathematical Programming 106 (1) (2006) 25–57. doi:10.1007/s10107-004-0559-y.

[17] D. Schwanenberg, M. Xu, T. Ochterbeck, C. Allen, D. Karimanzira, Short-term management of hydropower assets of the Federal Columbia River Power System, Journal of Applied Water Engineering and Research 2 (1) (2014) 25–32. doi:10.1080/23249676.2014.912952.

[18] S. M. Hosseini, B. Zahraie, Development of reservoir operation policies using integrated optimization-simulation approach, Journal of Agricultural Science and Technology 12 (2010) 433–446.

[19] E. A. Eschenbach, T. Magee, E. Zagona, M. Goranflo, R. Shane, Goal Programming Decision Support System for Multiobjective Operation of Reservoir Systems, Journal of Water Resources Planning and Management 127 (2) (2001) 108–120. doi:10.1061/(ASCE)0733-9496(2001)127:2(108).

[20] E. K. Can, M. H. Houck, Real-time reservoir operations by goal programming, Journal of Water Resources Planning and Management 110 (3) (1984) 297–309.

[21] G. V. Loganathan, D. Bhattacharya, Goal-programming techniques for optimal reservoir operations, Journal of Water Resources Planning and Management 116 (6) (1990) 820–838.

[22] B. A. Foued, M. Sameh, Application of goal programming in a multi-objective reservoir operation model in Tunisia, European Journal of Operational Research 133 (2) (2001) 352–361.

[23] M. J. McGregor, J. B. Dent, An application of lexicographic goal programming to resolve the allocation of water from the Rakaia River (New Zealand), Agricultural Systems 41 (3) (1993) 349–367. doi:10.1016/0308-521X(93)90009-Q.

[24] M. Bravo, I. Gonzalez, Applying stochastic goal programming: A case study on water use planning, European Journal of Operational Research 196 (3) (2009) 1123–1129. doi:10.1016/j.ejor.2008.04.034.

[25] S. R. Agha, Use of goal programming and integer programming for water quality management – A case study of Gaza Strip, European journal of operational research 174 (3) (2006) 1991–1998.

[26] C.-S. Lee, C.-G. Wen, Application of multiobjective programming to water quality management in a river basin, Journal of environmental management 47 (1) (1996) 11–26.

[27] R. E. Moore, F. Bierbaum, Methods and applications of interval analysis, Vol. 2, SIAM, 1979.

[28] A. Ben-Tal, L. El Ghaoui, A. Nemirovski, Robust Optimization, Vol. 53, Princeton University Press, 2009. doi:10.1137/080734510.

[29] T. C. Pagano, D. L. Shrestha, Q. J. Wang, D. Robertson, P. Hapuarachchi, Ensemble dressing for hydrological applications, Hydrological Processes 27 (1) (2013) 106–116. doi:10.1002/hyp.9313.

[30] J. Dupačová, G. Consigli, S. W. S. W. Wallace, J. Dupacova, G. Consigli, S. W. S. W. Wallace, J. Dupačová, G. Consigli, S. W. S. W. Wallace, J. Dupacova, G. Consigli, S. W. S. W. Wallace, Scenarios for multistage stochastic programs, Annals of Operations Research 100 (1–4) (2000) 25–53. doi:10.1023/A:1019206915174.

[31] N. Gröwe-Kuska, H. Heitsch, W. Römisch, Scenario reduction and scenario tree construction for power management problems, in: 2003 IEEE Bologna PowerTech - Conference Proceedings, Vol. 3, 2003, pp. 152–158. doi:10.1109/PTC.2003.1304379.

[32] H. Elmqvist, Modelica – A unified object-oriented language for physical systems modeling, Simulation Practice and Theory 5 (6) (1997) p32. doi:10.1016/S0928-4869(97)84257-7.

[33] P. Fritzson, P. Aronsson, H. Lundvall, K. Nyström, A. Pop, L. Saldamli, D. Broman, The OpenModelica Modeling, Simulation, and Software Development Environment, Simulation News Europe 44 (45).

[34] J. Åkesson, M. Gäfvert, H. Tummescheit, JModelica - An Open Source Platform for Optimization of Modelica Models, MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling (2009) 8.

[35] K. Berg, K. Nyström, Hydrological modeling in Modelica, in: G. Schmitz (Ed.), Proceedings of the 4th International Modelica Conference, Hamburg, March 7-8, 2005, 2005.

[36] B. Lie, Y. Ruan, I. Andreassen, Modeling for control of run-of-river power plant, in: Proceedings, 54th International Conference of Scandinavian Simulation Society (SIMS 2013), October 16-17 2013, Bergen, Norway, 2013.

[37] L. Saldamli, P. Fritzson, B. Bachmann, Extending Modelica for partial differential equations, in: 2nd International Modelica Conference, proceedings, 2002, pp. 307–314.

[38] S. H. Biddle, Optimizing the TVA Reservoir System Using RiverWare, in: Bridging the Gap: Meeting the World's Water and Environmental Resources Challenges, 2001, pp. 1–6. doi:10.1061/40569(2001)149.

[39] J. Åkesson, Optimica - an extension of Modelica supporting dynamic optimization, in: 6th International Modelica Conference 2008, 2008.

[40] M. Werner, J. Schellekens, P. Gijsbers, M. van Dijk, O. van den Akker, K. Heynert, The Delft-FEWS flow forecasting system, Environmental Modelling and Software 40 (2013) 65–77. doi:10.1016/j.envsoft.2012.07.010.

[41] P. Kunkel, V. Mehrmann, Differential-Algebraic Equations, European Mathematical Society Publishing House, 2006.

[42] J. Andersson, J. Åkesson, M. Diehl, CasADi: A symbolic package for automatic differentiation and optimal control, in: Lecture Notes in Computational Science and Engineering, Vol. 87 LNCSE, 2012, pp. 297–307. arXiv:arXiv:1011.1669v3, doi:10.1007/978-3-642-30023-3_27.

[43] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuejols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, A. Waechter, An algorithmic framework for convex mixed integer nonlinear programs, Discrete Optimization 5 (2) (2008) 186–204. doi:10.1016/j.disopt.2006.10.011.

[44] GNU General Public License.
URL http://www.gnu.org/licenses/gpl.html

[45] J. Talsma, D. Schwanenberg, J. Gooijer, K.-J. van Heeringen, B. P. J. Becker, Model Predictive Control For Real Time Operation Of Hydraulic Structures For Draining The Operational Area Of The Dutch Water Authority Noorderzijlvest, in: International Conference on Hydroinformatics, 2014.

[46] Rijksoverheid, Waterbesluit van 30 november 2009.
URL http://wetten.overheid.nl/BWBR0026872/2016-07-01/#Hoofdstuk2_Paragraaf1_Artikel2.1

[47] A. Weerts, G. Prinsen, S. Patzke, W. van Verseveld, H. Berger, T. Kroon, Operational Water Resources Forecasting System for The Netherlands, in: AGU Fall Meeting Abstracts, Vol. 1, 2011, p. 1498.

[48] W. Berendrecht, A. Weerts, A. B. Veldhuizen, T. Kroon, An operational drought forecasting system using coupled models for groundwater, surface water and unsaturated zone, IAHS-AISH publication 341 (2010) 3–8.