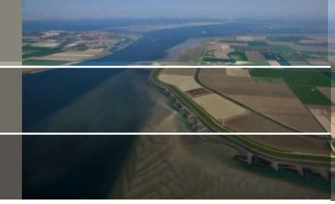
Deltares
Enabling Delta Life

Introduction to RTC-Tools 2.0

Jorn Baayen

October 28, 2016

Schedule



- 09:00: Introduction of participants
- 09:15: Introduction to RTC-Tools 2.0
- 10:30: Coffee break

- 11:00: Multi-objective optimization with RTC-Tools
- 11:30: Optimization under forecast uncertainty
- 12:00: Software installation
- 12:30: Lunch

- 14:00: Breakout session #1
- 15:30: Coffee break

- 16:00: Breakout session #2
- 17:30: Drinks

Breakout sessions

	Room	Taught by
Multi-objective optimization of a reservoir system	Colloquium	Olav van Duin, Matthijs den Toom (in partial absentia)
Water allocation	High Tech	Peter Gijsbers
Energy-efficient polder drainage	Ambition	Tjerk Vreeken, Jan Talsma

Every session will be held twice: From 14:00 to 15:30, and from 16:00 to 17:30.

Outline for first slot

- Introduction round.

- RTC-Tools: What is it?
- The history of RTC-Tools.

- Model predictive control.
- Reliability conditions for operational optimization.
- Convex optimization.

- Modelling with Modelica and RTC-Tools.

Introduction round

Who is who?

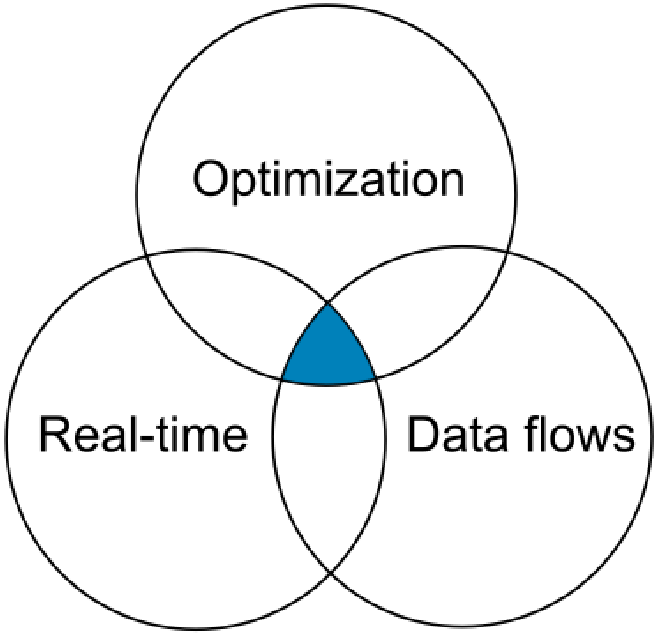
- Why are you interested in RTC-Tools?
- What are your expectations for the day?

RTC-Tools: Scope

RTC-Tools is the Deltares toolbox for control and optimization of environmental systems.

Delft-FEWS is an open data handling platform, used for the aggregation of (real-time) environmental data flows.

Together, they provide a platform for the development of decision support systems.



Netherlands: Noorderzijlvest water board

A decision support and control system for the Noorderzijlvest water board. The system helps to reduce drainage costs, by making use of energy price, tidal sea water level, and rainfall predictions.

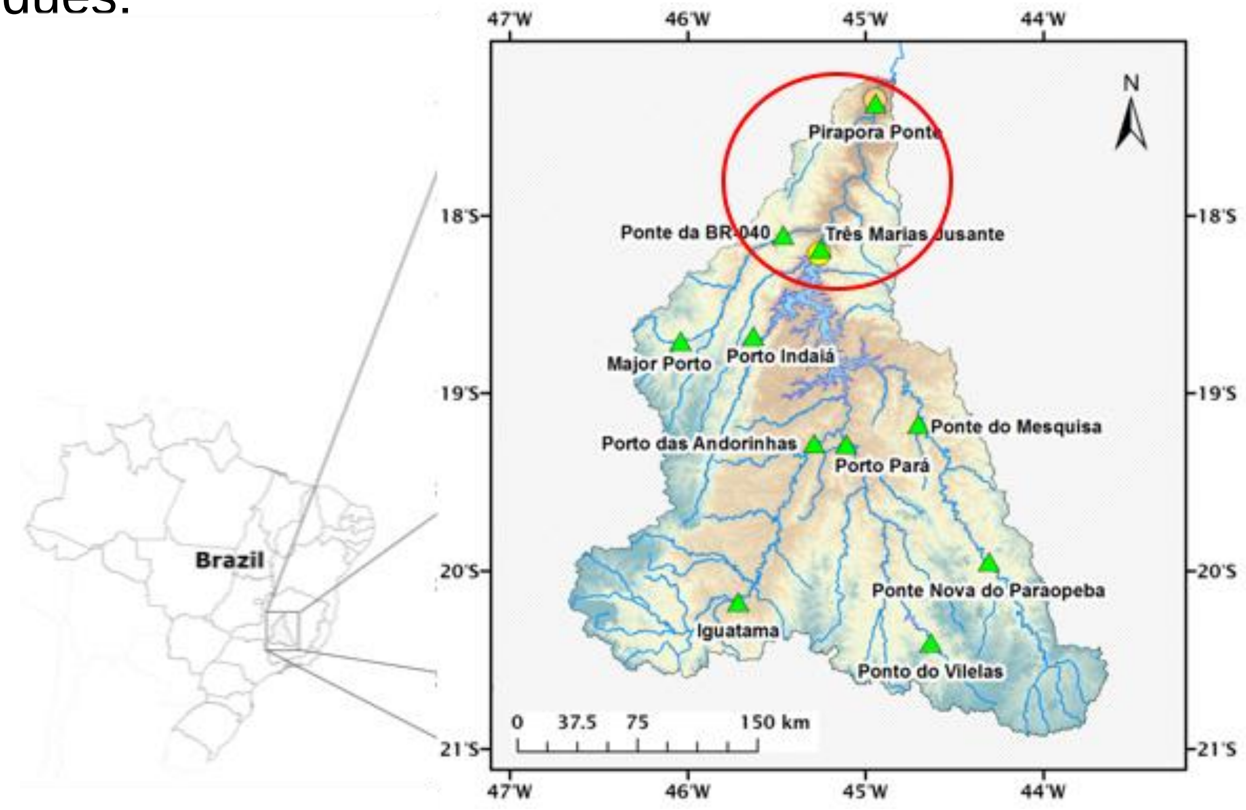
USA: Bonneville Power Authority

A decision support system for hydropower dispatch on the Columbia river. The system helps to maximize revenue from power sales, while keeping the system compliant with regulations, through multi-objective optimization techniques.

Deltares

Brazil: Decision support for Tres Marias dam (CEMIG)

A decision support system for the Tres Marias dam. The system helps to reduce flooding in Pirapora using stochastic optimization techniques.

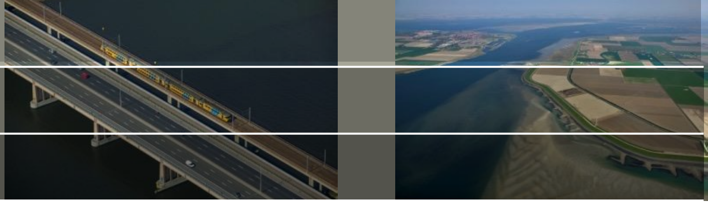


Deltares

History

- 2005: Reservoir module for Delft-FEWS.
- 2012: Dirk Schwanenberg releases first version of RTC-Tools source code to the public. RTC-Tools 1.x connected non-linear hydraulic and reservoir models to the IPOPT optimizer.
    - Promising results, many scientific publications
    - High interest from reservoir operators
    - But challenging to operationalize, and hard to extend
- 2015: Work starts on new mathematically rigorous foundation, initially as an experiment of Jorn Baayen and Matthijs den Toom.
- 2016: First pilot projects on new foundation. Peter Gijsbers develops water allocation tool for Rijkswaterstaat using new framework. Klaas-Jan van Heeringen and Ivo Pothof launch projects to develop decision support systems for a number of water boards in The Netherlands.
- 2016: RTC-Tools 2.0 released.

RTC-Tools 2.0



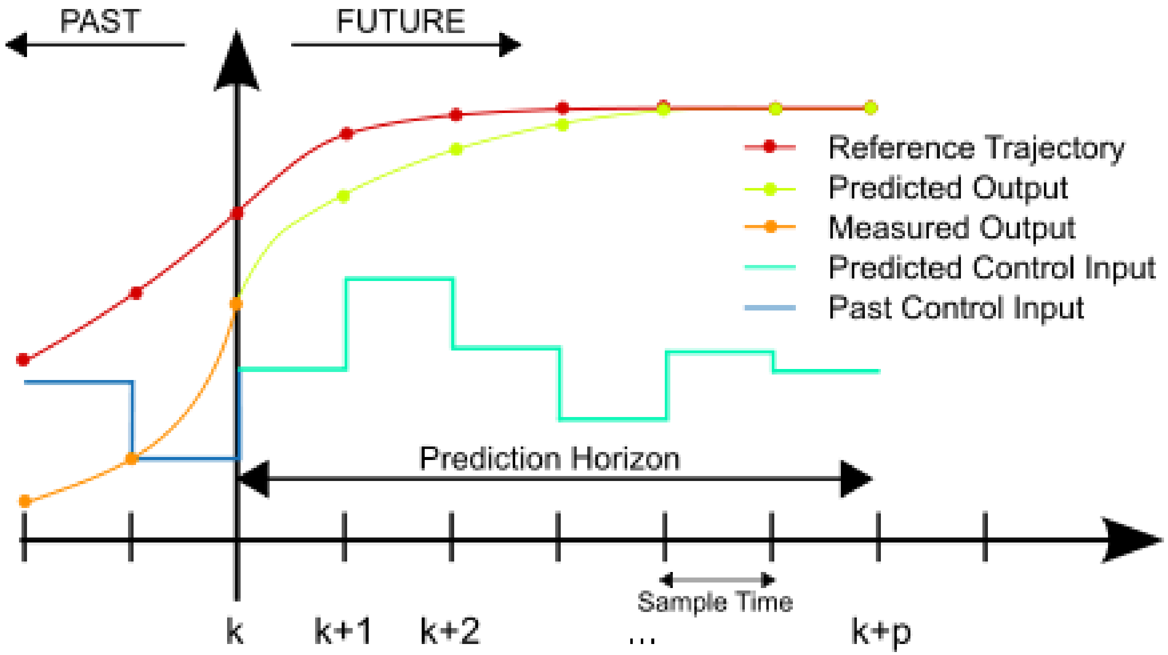
RTC-Tools 2.0 is a toolbox for control and optimization of environmental systems.

- Interdisciplinary, object-oriented modeling using Modelica
- Mathematical framework designed for stable operation in environments that require consistent results
- Optimization under uncertainty
- Multi-objective optimization
- Integration with Delft-FEWS
- Python scripting

python™

MODELICA

Model predictive control



Source: Wikipedia. CC BY-SA 3.0.

- Predict system state based on model
- Compute control inputs that maximize performance over prediction horizon
- Implement first computed control input
- Repeat procedure at next time step

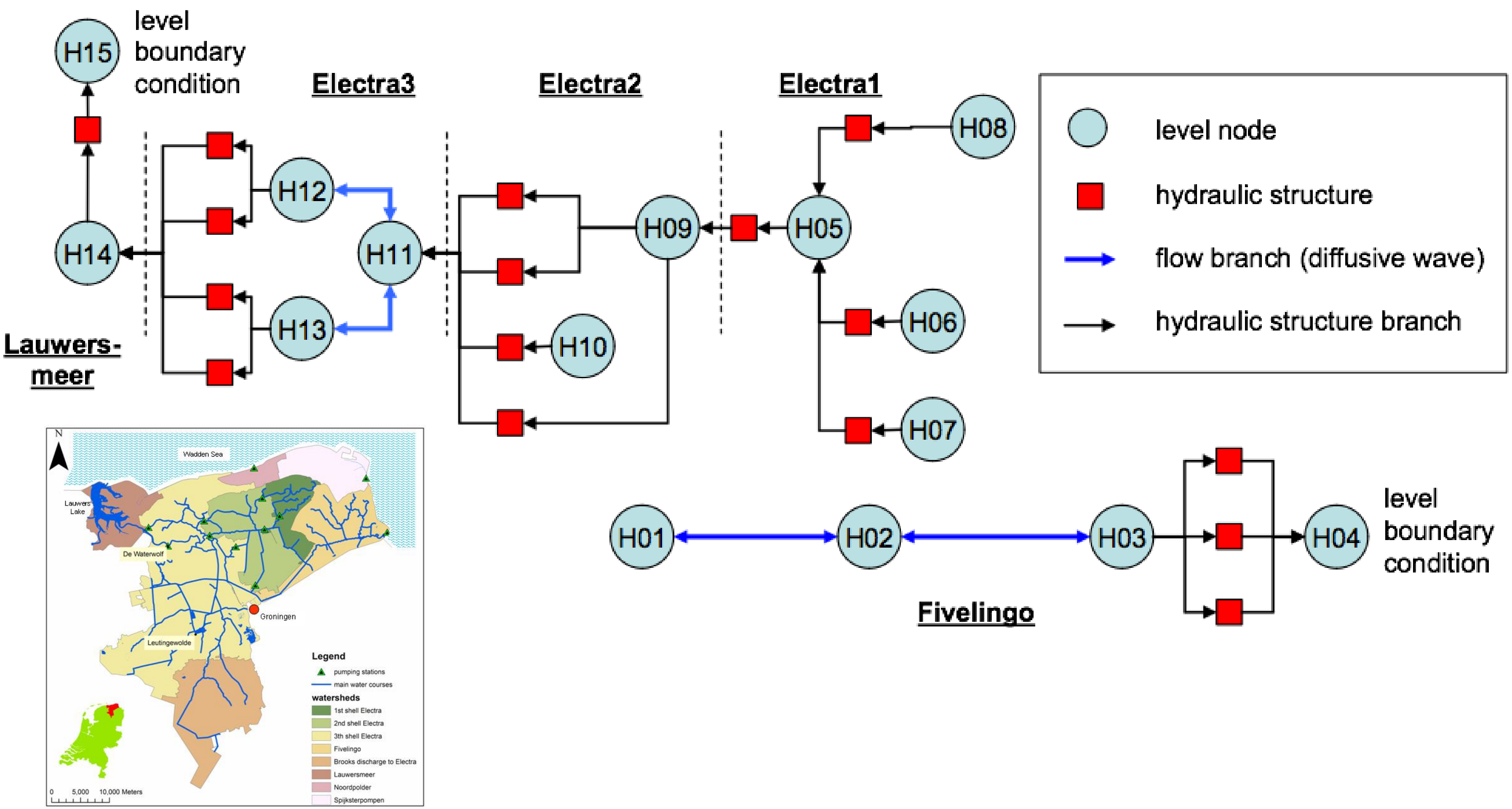
Prediction model

To be able to predict the state of the system inside the optimization, the following are required:

1. A predictive model relating control inputs and boundary conditions to the evolution of the system state.
2. Forecasts for the boundary conditions of the system over the prediction horizon:
   - Inflow forecasts
   - Load forecasts
   - …

Note that the predictive model is always used, even when the controller is used to control a simulation: predictive model ≠ simulation model!

Prediction model




Prediction model

A good prediction model satisfies several requirements:

- *Accurate*: It captures the relevant physical processes with sufficient accuracy.
- *Simple*: It focuses on the essential processes. Details are left out. Optimizing for details is a bad idea, considering the inaccuracies inherent in any inflow forecast. Less = more.
- *Quick*: As it will need to be evaluated many times during optimization, a single run needs to be computationally inexpensive.

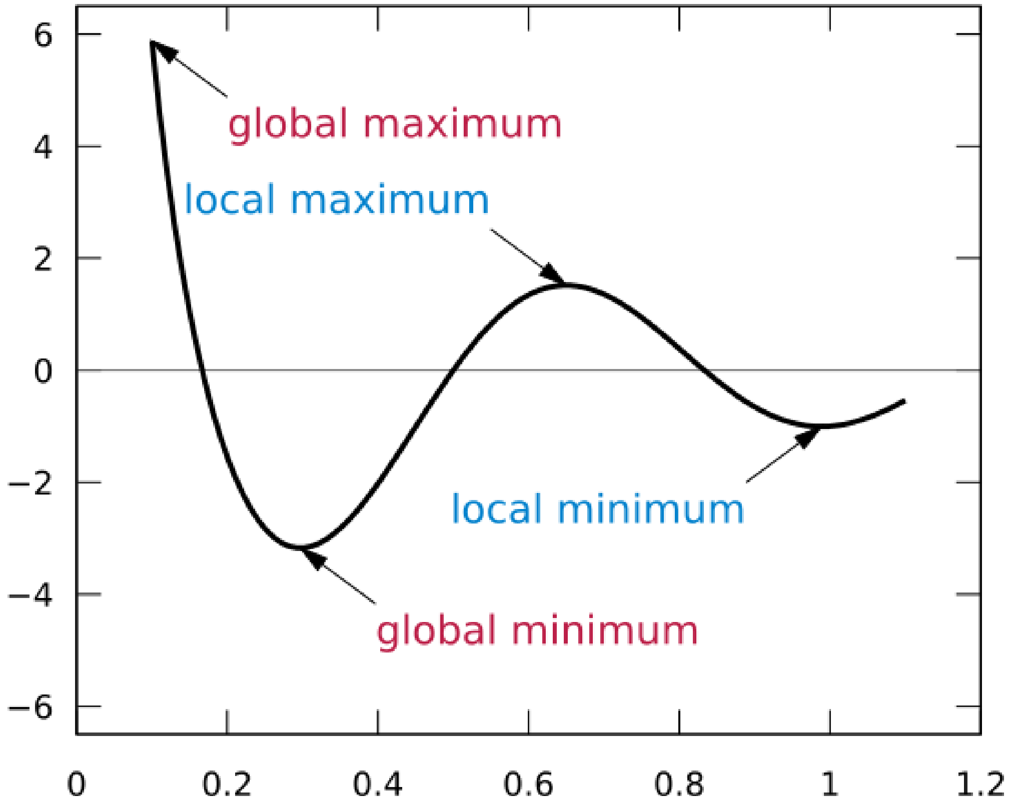
On top of these conceptual requirements, there are several mathematical requirements to ensure stable and consistent optimization results. We will briefly discuss these later on.

Reliability axioms

A decision support system that is used day in, day out needs to be reliable. This need can be made precise with six axioms:

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

Local and global optima



Source: Wikipedia. GFDL 1.2.

From axioms to convexity

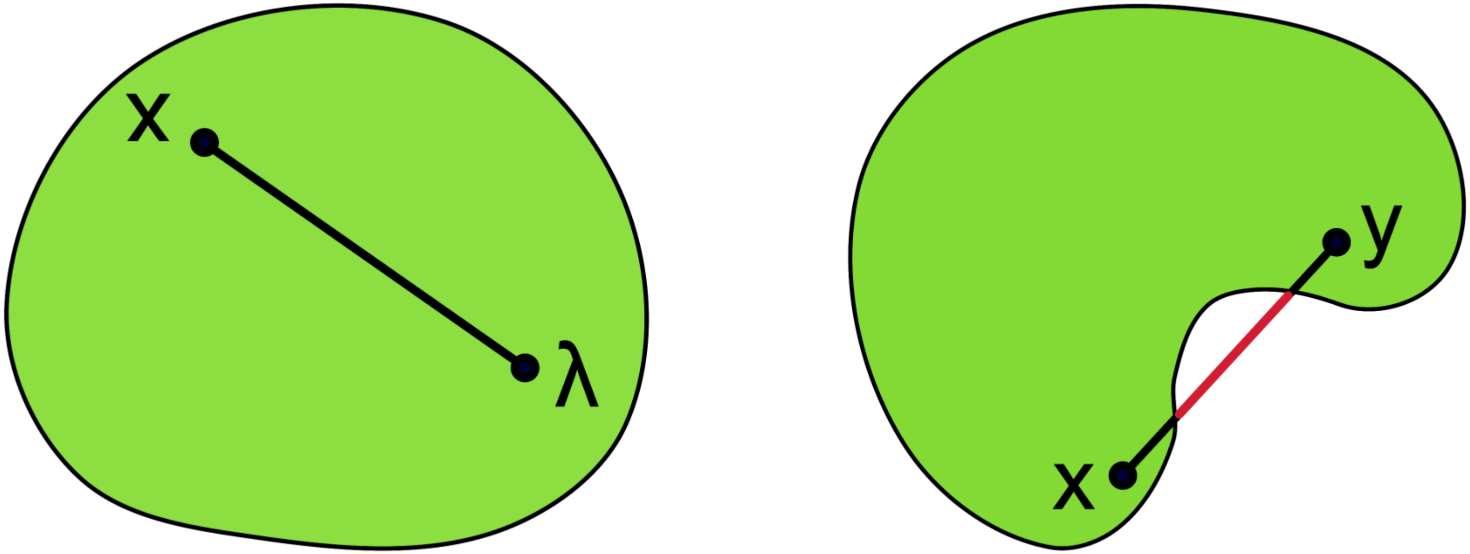
Suppose we had an optimization problem that would only have globally optimal solutions.

That would give us:

- *Quality*: Every solution is a globally optimal solution.
- *Stability*: Changing seed solutions or optimizer settings won't change the quality of the end solution.

So-called *convex* optimization problems only admit globally optimal solutions. Convex problems can be solved efficiently using deterministic methods.

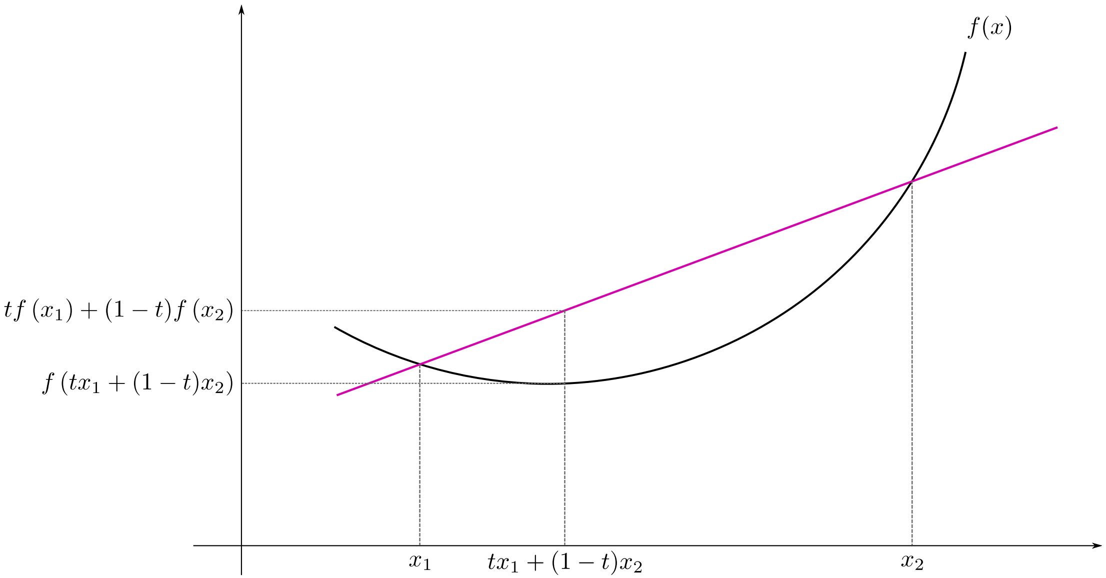
Convex sets



Source: Wikipedia. CC BY-SA 3.0.

Deltares

Convex functions



Source: Wikipedia. CC BY-SA 3.0.

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

Deltares

Convex optimization

$$\min f(x) \quad \text{subject to}$$
$$g(x) \leq 0$$
$$h(x) = 0$$

Problem is called convex when:

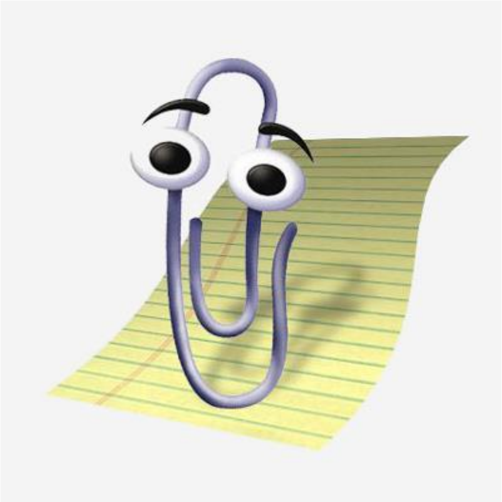
- f is a convex function
- g is a convex function
- h is an affine function:
  - $h(x) = 0 \Leftrightarrow h(x) \leq 0$ and $-h(x) \leq 0$.
  - h must be both convex and concave, i.e., affine: $h(x) = ax + b$.
  - **This is quite restrictive**

Convex problems only admit global optima.

Convex optimization in practice

By following certain simple rules when composing an optimization problem, convexity can be guaranteed.

RTC-Tools helps the modeler follow these rules by emitting warnings whenever a rule is violated.

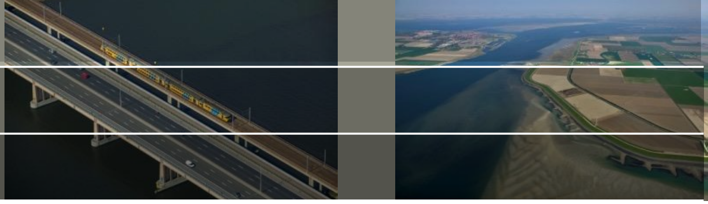


Interdisciplinary modeling: Modelica

Modelica is a language for object-oriented, declarative, equation-based modeling of dynamical systems.

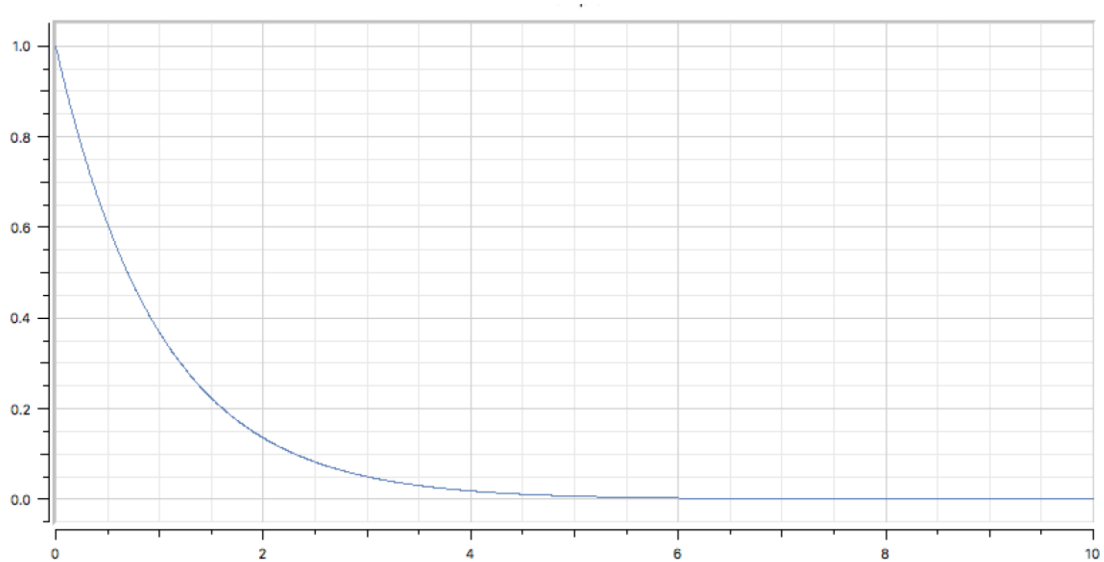
- Open standard
- Independent of application domain
- In industrial use at BMW, Airbus, Toyota, Alstom, Siemens, …

Models can be written using a text editor (with a Python-like syntax), or using a GUI.



Hello World: Exponential ODE

```
model Example
  parameter Real k = -1.0;
  Real x(start = 1.0);
equation
  der(x) = k * x;
  ¤;
end Example;
```



Types

- *Real*
- *Integer*
- *Boolean*
- *String*
- …

```
Real x;
Boolean switch;
Integer count;
```

Classes create new types: More on this later.

Type prefixes

Variability:

- (continuous in time)
- *parameter* (constant in time)

Relation to environment:

- *input* (value provided by environment)
- *output* (value provided to environment)

```
parameter Real k = -1.0;
Real x(start = 1.0);
input Real u;
output Real y;
```

Units

Good practice to use a real type annotated with a physical unit.

Real types with SI units live in the *package* “Modelica.SIunits”.

```
Modelica.SIunits.Velocity v;
Modelica.SIunits.Position x;
```


Equations

The equality sign, “=“, declares an equality between the expressions on the left and right hand sides.

It is *not* an assignment! Different from Python.

The operator *der* gives the time-derivative of a real variable.

```
der(x) = k * x;
0 = y - 4 * x;
```

Model objects

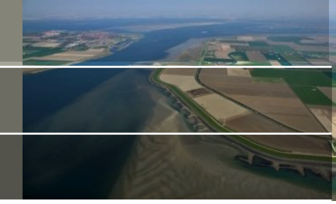
Modelica is object-oriented. Like Python, Java, and C++. A Modelica model object is declared using the keyword *model*.

A model generally consists of two sections:

- Variable declarations
- Equations

```
model Example
  Modelica.SIunits.Velocity v;
  Modelica.SIunits.Position x;
  ...
equation
  der(x) = v;
  ...
  ¤;
end Example;
```

Inheritance



Make more complex models from simpler, more general ones:

```
model ComplicatedModel
  extends Example;
  Real z;
equation
  z ^ 2 = x;
  ¤;
end ComplicatedModel;
```

Nesting models

```
model SimpleModel
  parameter Real k = -1.0;
  Real x(start = 0.0);
  input Real u;
equation
  der(x) = k * x + u;
end SimpleModel;
```

```
model ParentModel
  SimpleModel s;
equation
  s.u = sin(time);
end ParentModel;
```

Packages

A *Package* is a special kind of object, which contains other objects such as models and possibly other packages.

```
Modelica.SIunits.Position x;
```

Deltares

Connectors

```
connector HQPort
  Modelica.SIunits.Position H;
  flow Modelica.SIunits.VolumeFlowRate Q;
  ¤;
end HQPort;

partial model HQTwoPort
  HQPort HQUp ¤;
  HQPort HQDown ¤;
  ¤;
end HQTwoPort;

connect(model1.HQUp, model2.HQDown);
```

$$H_i = H_j$$

$$\sum Q_i = 0$$

Good modeling practice

Make your components *balanced:*

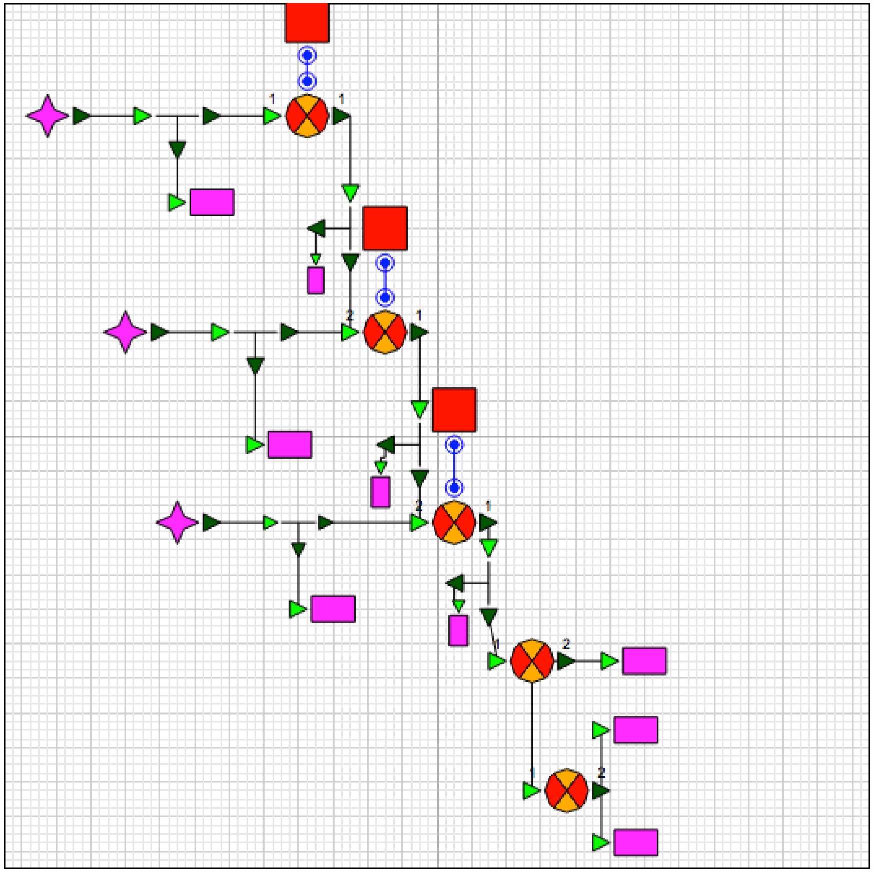
Number of equations = number of non-input, non-constant variables

If the components are balanced, then so is the model.

Balanced models can be simulated and therefore optimized.

Water allocation in Citarum basin, Indonesia

```
model citarum
  import SI = Modelica.SIunits;
  input SI.VolumeFlowRate inflow_Saguling_Q;
  input SI.VolumeFlowRate inflow_Cirata_Q;
  input SI.VolumeFlowRate inflow_Jatiluhur_Q;
  input SI.VolumeFlowRate lateralLoss_Saguling_QLat1;
  input SI.VolumeFlowRate lateralLoss_Cirata_QLat1;
  input SI.VolumeFlowRate lateralLoss_Jatiluhur_QLat1;
  input SI.VolumeFlowRate lateralLoss_SagulingCirata_Qlat1;
  input SI.VolumeFlowRate lateralLoss_CirataJatiluhur_Qlat1;
  input SI.VolumeFlowRate lateralLoss_JatiluhurDemand_Qlat1;
  input SI.VolumeFlowRate terminal_Agriculture_Qin2;
  input SI.VolumeFlowRate terminal_River_Qin2;
  input SI.VolumeFlowRate lateralLoss_Saguling_Qin2;
  input SI.VolumeFlowRate lateralLoss_Cirata_Qin2;
  input SI.VolumeFlowRate lateralLoss_Jatiluhur_Qin2;
  input SI.VolumeFlowRate lateralLoss_SagulingCirata_Qin2;
  input SI.VolumeFlowRate lateralLoss_CirataJatiluhur_Qin2;
  input SI.VolumeFlowRate lateralLoss_JatiluhurDemand_Qin2;
  Deltares.Flow.SimpleRouting.Branches.LateralLoss lateralLoss_Saguling ;
  Deltares.Flow.SimpleRouting.BoundaryConditions.Inflow inflow_Saguling ;
  Deltares.Flow.OpenChannel.Storage.Linear linear_Cirata(Area = 30000000, Htail = 103, Hloss = 4) ;
  Deltares.Flow.SimpleRouting.Branches.LateralLoss lateralLoss_Cirata ;
  Deltares.Flow.SimpleRouting.BoundaryConditions.Inflow inflow_Cirata ;
  Deltares.Flow.SimpleRouting.Nodes.Node node_Agriculture(nout = 2) ;
  Deltares.Flow.SimpleRouting.Nodes.Node node_Drinking(nout = 2) ;
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_Drinking ;
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_Agriculture ;
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_River ;
  Deltares.Flow.SimpleRouting.Nodes.NodeHQPort nodeHQPort_Saguling(nout = 1) ;
  Deltares.Flow.SimpleRouting.Nodes.NodeHQPort nodeHQPort_Jatiluhur(nout = 1, nin = 2) ;
  Deltares.Flow.SimpleRouting.BoundaryConditions.Inflow inflow_Jatiluhur ;
  Deltares.Flow.SimpleRouting.Branches.LateralLoss lateralLoss_Jatiluhur ;
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_Saguling ;
  Deltares.Flow.SimpleRouting.Branches.LateralLoss lateralLoss_JatiluhurDemand ;
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_Jatiluhur ;
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_Cirata ;
  Deltares.Flow.SimpleRouting.Branches.LateralLoss lateralLoss_SagulingCirata ;
  Deltares.Flow.SimpleRouting.Branches.LateralLoss lateralLoss_CirataJatiluhur ;
  Deltares.Flow.OpenChannel.Storage.Linear linear_Jatiluhur(Area = 63000000, Htail = 27, Hloss = 1) ;
  Deltares.Flow.SimpleRouting.Nodes.NodeHQPort nodeHQPort_Cirata(nout = 1, nin = 2) ;
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_CirataJatiluhur ;
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_SagulingCirata ;
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_JatiluhurDemand ;
  Deltares.Flow.OpenChannel.Storage.Linear linear_Saguling(Area = 20000000, Htail = 252, Hloss = 28) ;
equation
  inflow_Saguling.Q = inflow_Saguling_Q;
  inflow_Cirata.Q = inflow_Cirata_Q;
  inflow_Jatiluhur.Q = inflow_Jatiluhur_Q;
  lateralLoss_Saguling.QLat_control = lateralLoss_Saguling_QLat1;
  lateralLoss_Cirata.QLat_control = lateralLoss_Cirata_QLat1;
  lateralLoss_Jatiluhur.QLat_control = lateralLoss_Jatiluhur_QLat1;
  lateralLoss_SagulingCirata.QLat_control = lateralLoss_SagulingCirata_Qlat1;
  lateralLoss_CirataJatiluhur.QLat_control = lateralLoss_CirataJatiluhur_Qlat1;
  lateralLoss_JatiluhurDemand.QLat_control = lateralLoss_JatiluhurDemand_Qlat1;
  node_Drinking.QOut_control[1] = 0;
  node_Agriculture.QOut_control[1] = terminal_Agriculture_Qin2;
  nodeHQPort_Saguling.QOut_control[1] = lateralLoss_SagulingCirata_Qin2;
  nodeHQPort_Cirata.QOut_control[1] = lateralLoss_CirataJatiluhur_Qin2;
  nodeHQPort_Jatiluhur.QOut_control[1] = lateralLoss_JatiluhurDemand_Qin2;
  connect(nodeHQPort_Saguling.HQ, linear_Saguling.HQ) ;
  connect(terminal_JatiluhurDemand.QIn, lateralLoss_JatiluhurDemand.QLat) ;
  connect(lateralLoss_SagulingCirata.QLat, terminal_SagulingCirata.QIn) ;
  connect(lateralLoss_CirataJatiluhur.QLat, terminal_CirataJatiluhur.QIn) ;
  connect(lateralLoss_SagulingCirata.QOut, nodeHQPort_Cirata.QIn[2]) ;
```



Modelica models and convexity

RTC-Tools discretizes ODE of the form

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

using the θ-method:

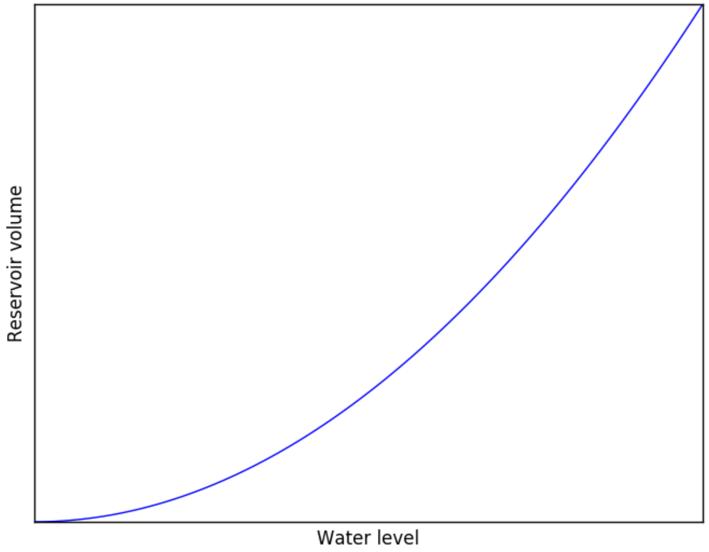
$$x(t_{i+1}) - x(t_i) = \Delta t[\theta f(x_{i+1}, u_{i+1}, t_{i+1}) + (1 - \theta) f(x_i, u_i, t_i)]$$

The discretized equations are included as constraints in the optimization problem (*collocation*). Consequently, for convexity to hold, the model equations must be <u>linear</u>.

Nonlinearity #1: Storage geometry

Storage volume is an increasing, but generally nonlinear, function of water level.

So this function cannot be included in the model.



- However, accounting of volumes is linear:

$$\dot{V} = Q_{in} - Q_{out}$$

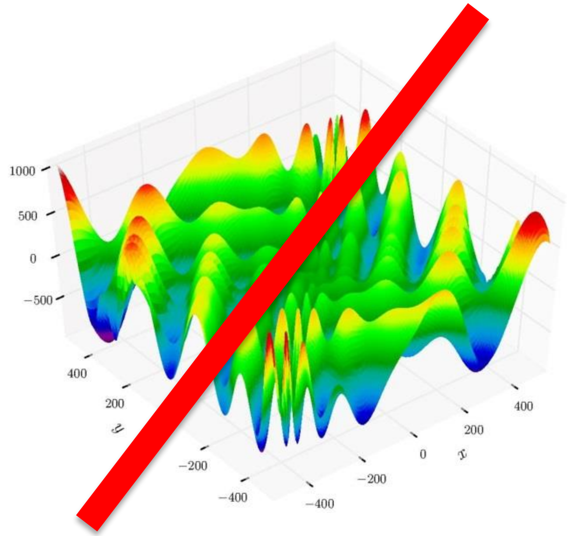
- **Solution**: Preprocess water level goals to volume goals.

Nonlinearity #2: Hydraulics

- Highly nonlinear friction term in diffusive wave equation:

$$\frac{\partial H}{\partial x} + \frac{C}{R} Q^2 = 0$$

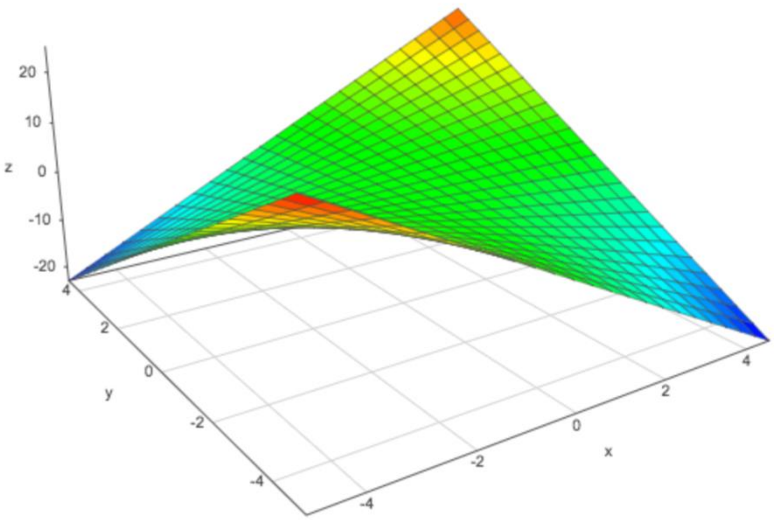
- When using many diffusive wave branches, large numbers of local minima are created. What to do?
- Linearization results in large errors; piecewise linearization results in large numbers of integer variables.
- We recommend to stay with <u>integrator-delay</u> (storage-and-delay) models.
- Ongoing research.



Deltares

Nonlinearity #3: Hydropower generation

Instantaneous power from a hydroelectric turbine: $P = \eta \rho g Q H$.



- Q and H are optimization variables.
- P nonlinear, even nonconvex, function of Q and H.
- **Solution**: Change of variables results in nonlinear but convex formulations for load balance and generation maximization goals.

Deltares

Outline for second slot

- Multi-objective optimization
  - Pareto front
  - Weighting method
  - Goal programming
- Forecast uncertainty
  - Sources of uncertainty
  - Forecast ensembles
  - Multi-stage stochastic optimization
- Opportunities ahead
- Software installation
- Lunch

Deltares

Multi-objective optimization

Suppose we have the following goals:

- Keep water levels within bounds as much as possible
- Maintain minimum spill flows for fish migration, if possible
- Apply best effort to track the generation request

Let $\{f_i : i \in I\}$ denote the set of functions encoding these goals. We have:

$$\min f_i \;\; \forall i \in I \text{ subject to}$$
$$g(x) \leq 0$$
$$h(x) = 0$$

How to solve this?

Pareto optimality

A solution x^* of the problem

$$\min f_i \ \forall i \in I \text{ subject to}$$
$$g(x) \leq 0$$
$$h(x) = 0$$

Is *Pareto-optimal* if there is no x^{**} such that for a j

$$f_j(x^{**}) < f_j(x^{**})$$

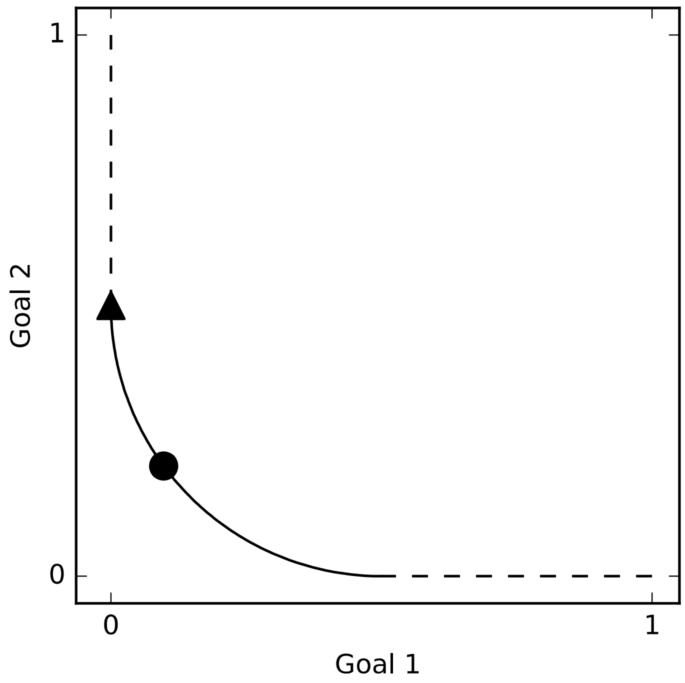
and for all $i \neq j$

$$f_i(x^{**}) \leq f_i(x^{**})$$

In words: Pareto optimality implies that no goal can be improved without making another one worse.

Pareto front

The *Pareto front* is the set of all Pareto-optimal solutions.



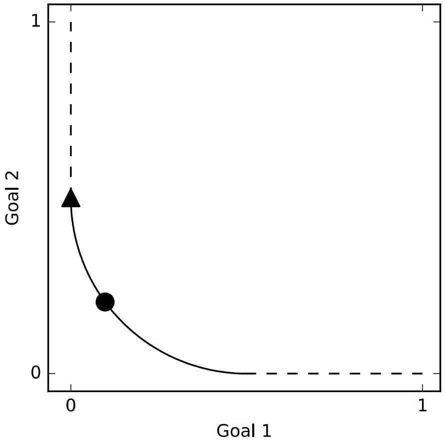
Weighting method

The weighting method transforms the multi-objective problem to the scalar problem

$$\min \sum_i \lambda_i f_i \text{ subject to}$$
$$g(x) \leq 0$$
$$h(x) = 0$$

- Problem: How to pick the weighting factors λ_i.
- And if the weighting factors are arbitrary to a degree, then so is the solution!

Solution on Pareto front shown with a circle.



Lexicographic goal programming

In lexicographic goal programming, we transform the multi-objective problem to a sequence of scalar optimization problems.

First, we order our goals. For example:

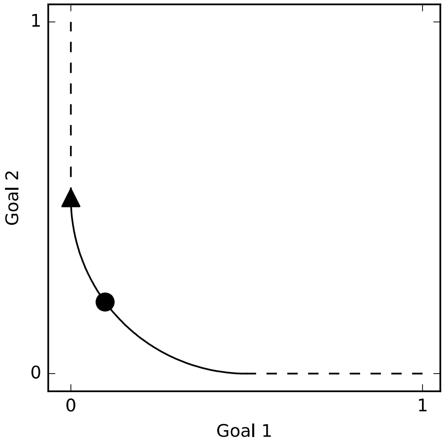
1. Keep water levels within bounds as much as possible
2. Maintain minimum spill flows for fish migration, if possible
3. Apply best effort to track the generation request

Lexicographic goal programming

The idea of the algorithm is:

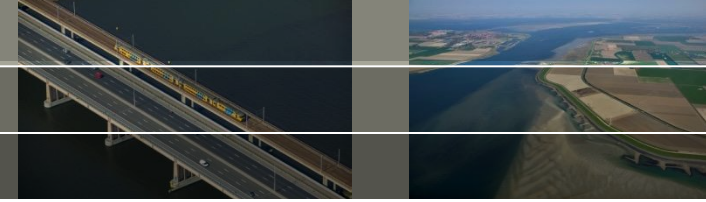
1. Minimize f_1 to yield a minimum objective value of ε_1.
2. Minimize f_2 to yield ε_2 subject to the additional constraints
   - $f_1(x) = \varepsilon_1$
3. Minimize f_3 subject to the additional constraints
   - $f_1(x) = \varepsilon_1$
   - $f_2(x) = \varepsilon_2$
4. …

Solution on Pareto front shown with an arrow.



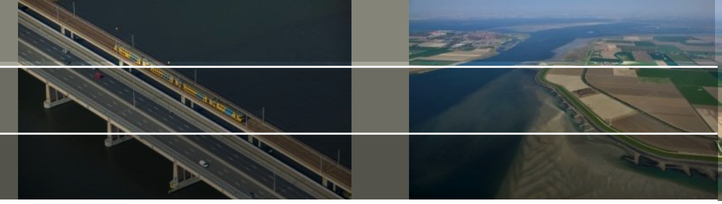
Deltares

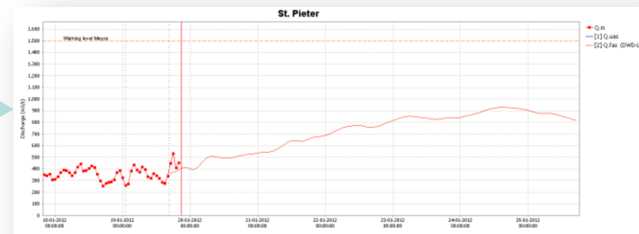
Probabilistic forecasting



Courtesy of Jan Verkade

Probabilistic forecasting

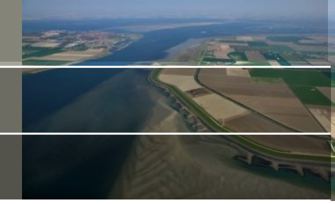




Where are the uncertainties?

Estimating predictive uncertainty: techniques

Ensemble techniques



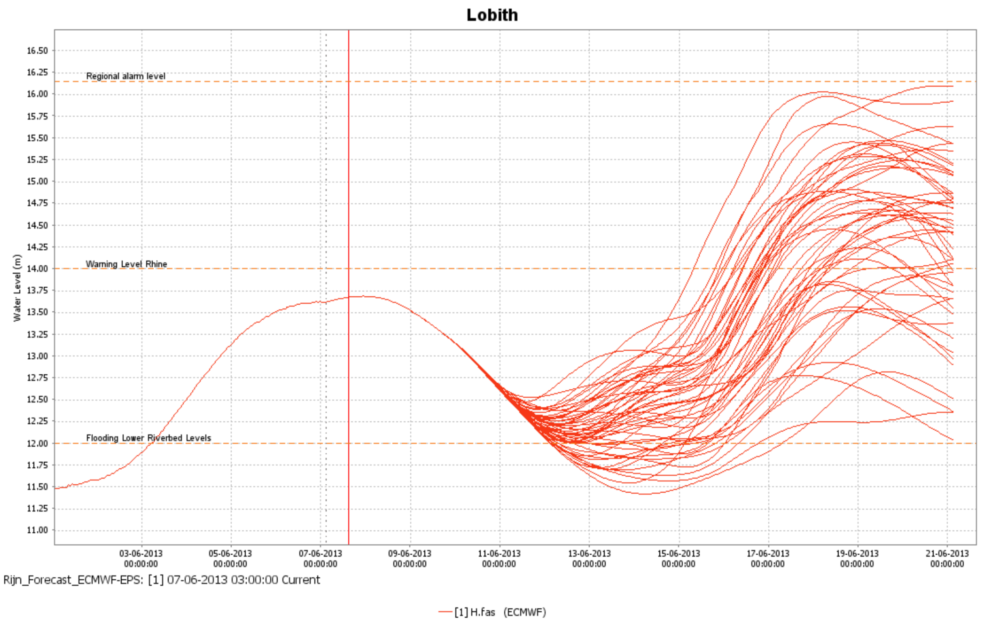
1. Use multiple, equally plausible inputs
   - Weather forecasts
   - Initial conditions
   - Parameters
   - …

2. Route all through a model:
   - Using one single model
   - Using multiple models (“multi-model”)

→ Model outputs will vary → “ensemble”

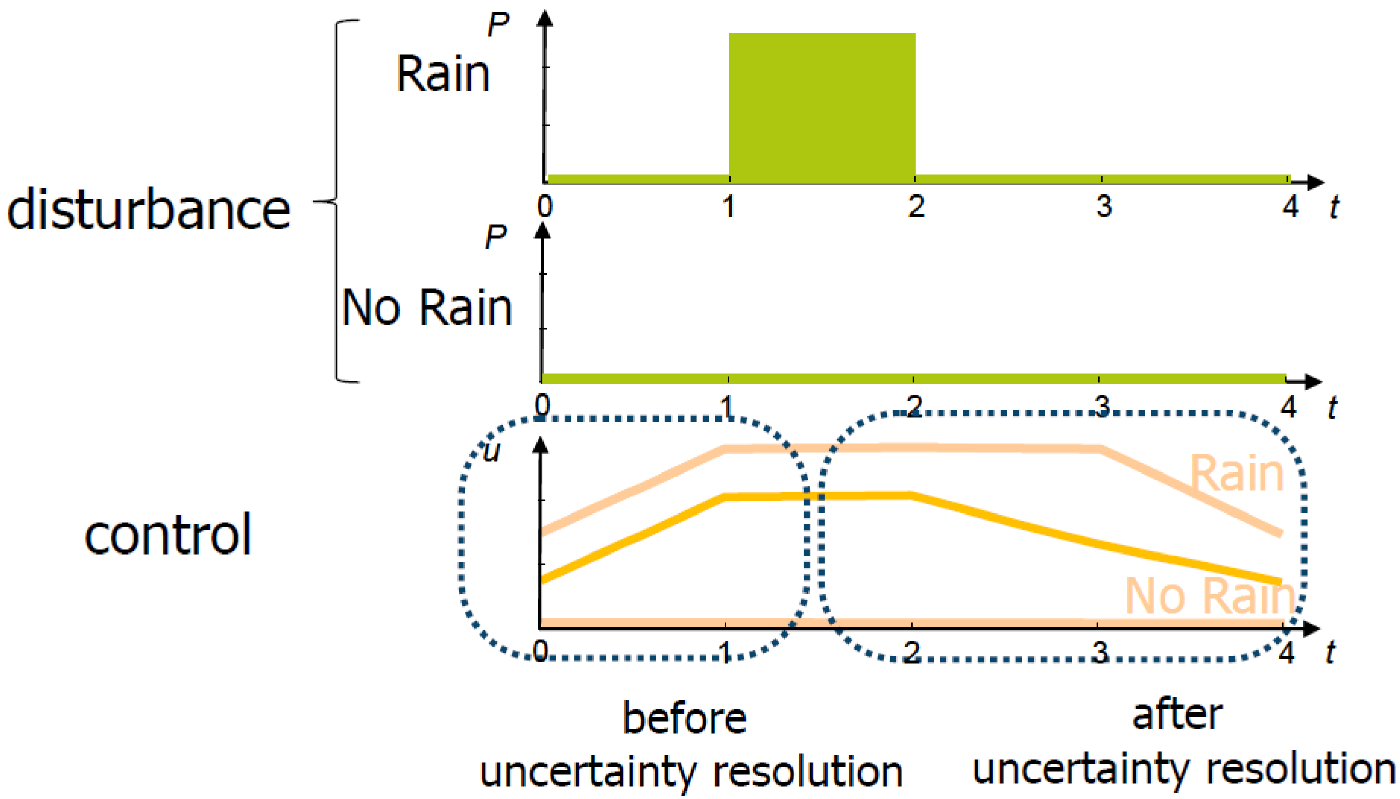
→ Individual model results are called “members”

Ensemble techniques



Multi-stage Stochastic Oprimization

Decision   Uncertainty Resolution   Decision

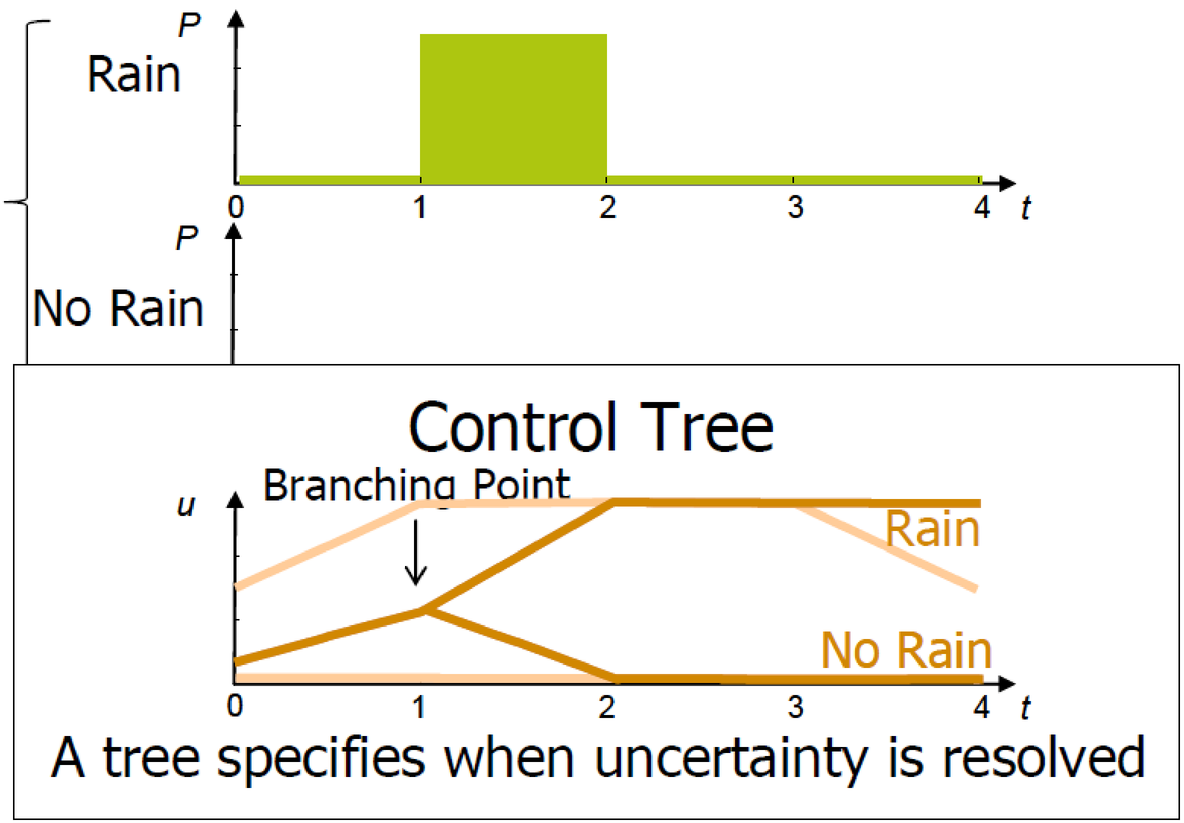


Multi-stage Stochastic Oprimization

Decision Uncertainty Resolution Decision

disturbance

control



Joint optimization of hydro and transmission systems

- Joint optimization of hydro generation and electrical power flow (OPF).
- Goal: Hydropower dispatch schedules that are robust against meteorological uncertainty and transmission grid contingencies (failing power lines). I.e., SCOPF + water accounting.

Source: bchydro.com

Software installation

	Download location	Password
RTC-Tools 2.0	download.deltares.nl	
OpenModelica	www.openmodelica.org	
Example pack	https://we.tl/2FHBTs1pzP	
	pw4GASTatDeltares	

Deltares