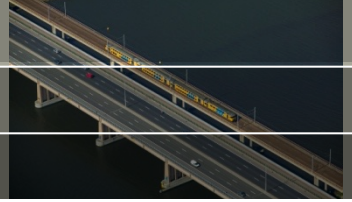
Deltares
Enabling Delta Life

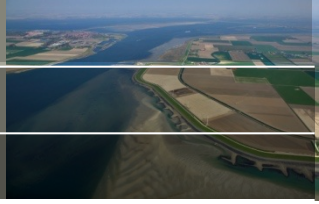
Introduction to RTC-Tools

Jorn Baayen

Vienna, September 7, 2017

Outline

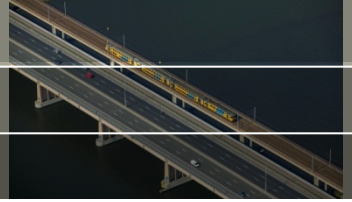


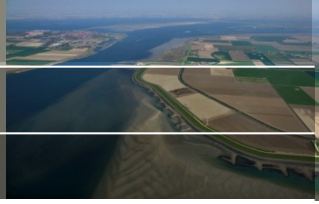


- What is RTC-Tools?
- Applications
- Technology

Deltares

RTC-Tools: Scope

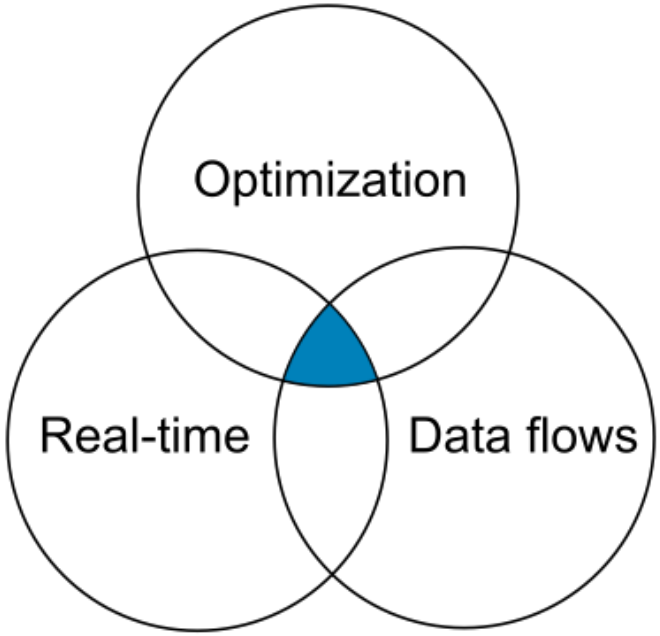




RTC-Tools is an open source toolbox for control and optimization of environmental systems.

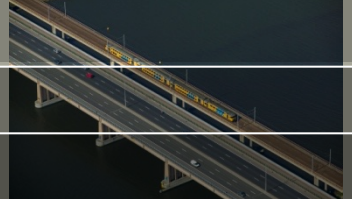
Delft-FEWS is an open data handling platform, used for the aggregation of (real-time) environmental data flows.

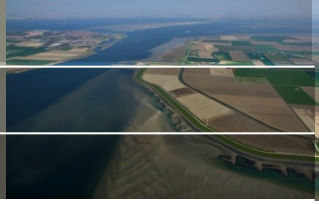
Together, they provide a platform for the development of decision support systems.



Deltares

RTC-Tools 2.0





RTC-Tools is an open source 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 predictive uncertainty
- Multi-objective optimization
- Integration with Delft-FEWS
- Python scripting

python™

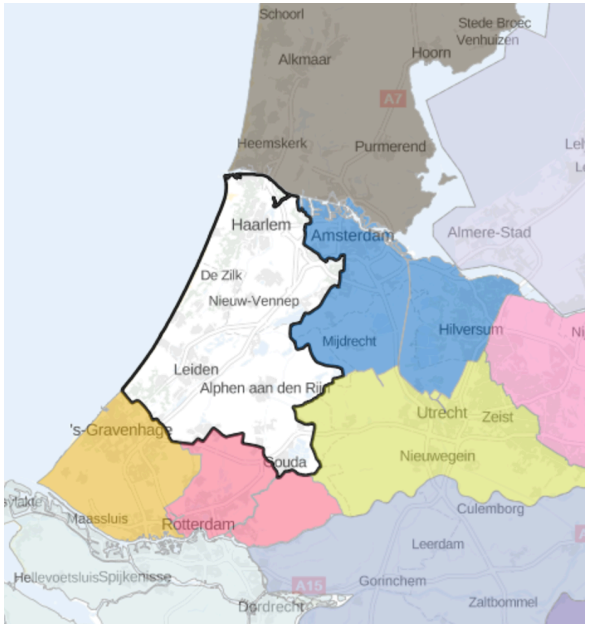
MODELICA

Deltares

Netherlands: Water board of Rijnland

A decision support and control system for the water board of Rijnland was brought online earlier this year.

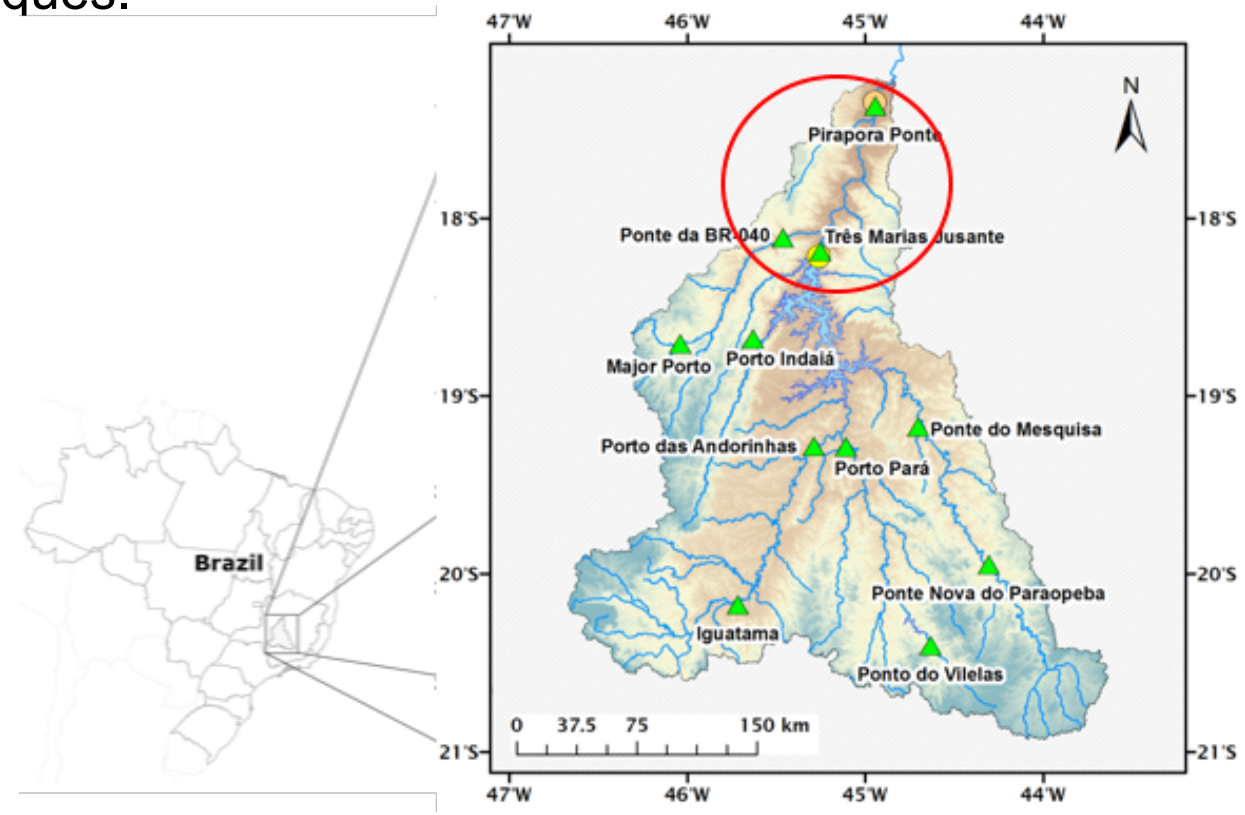
The system provides advice on the dispatch of pumping stations, taking into account the operational objectives of flood control, water quality, and cost savings.



Deltares

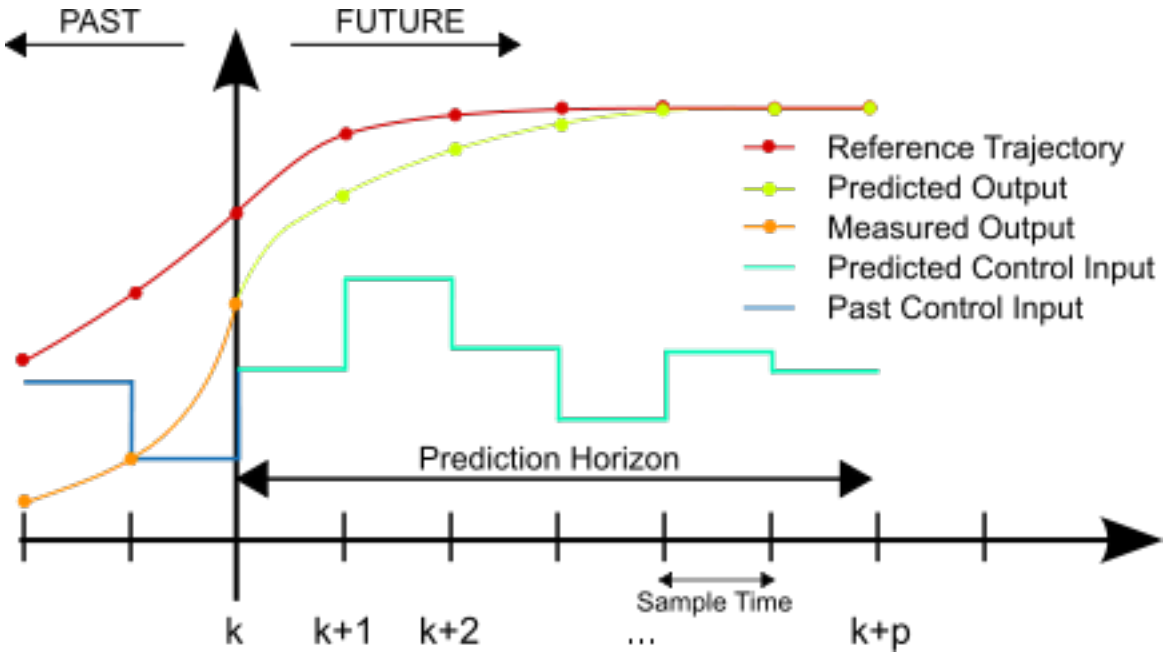
Brazil: Decision support for Tres Marias dam (CEMIG)

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



Deltares

Model predictive control

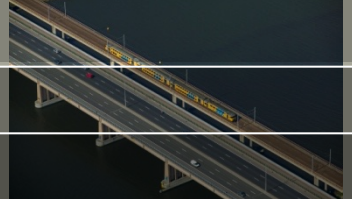


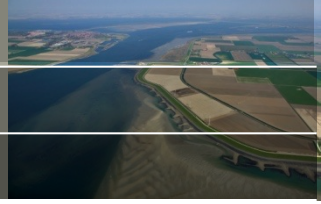
Source: Wikipedia. CC BY-SA 3.0.

- 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

Deltares

Prediction model

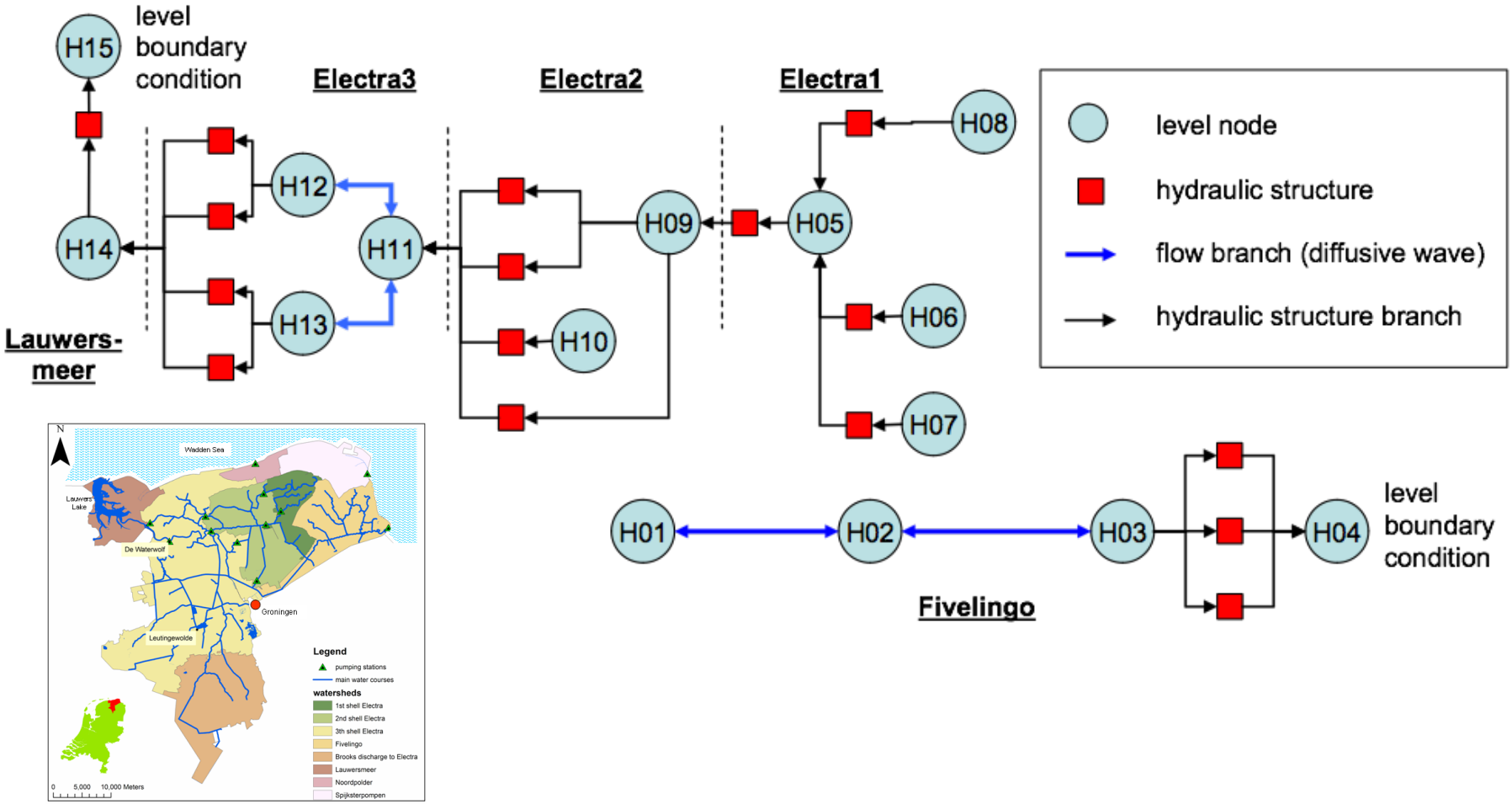




A good prediction model satisfies several requirements:

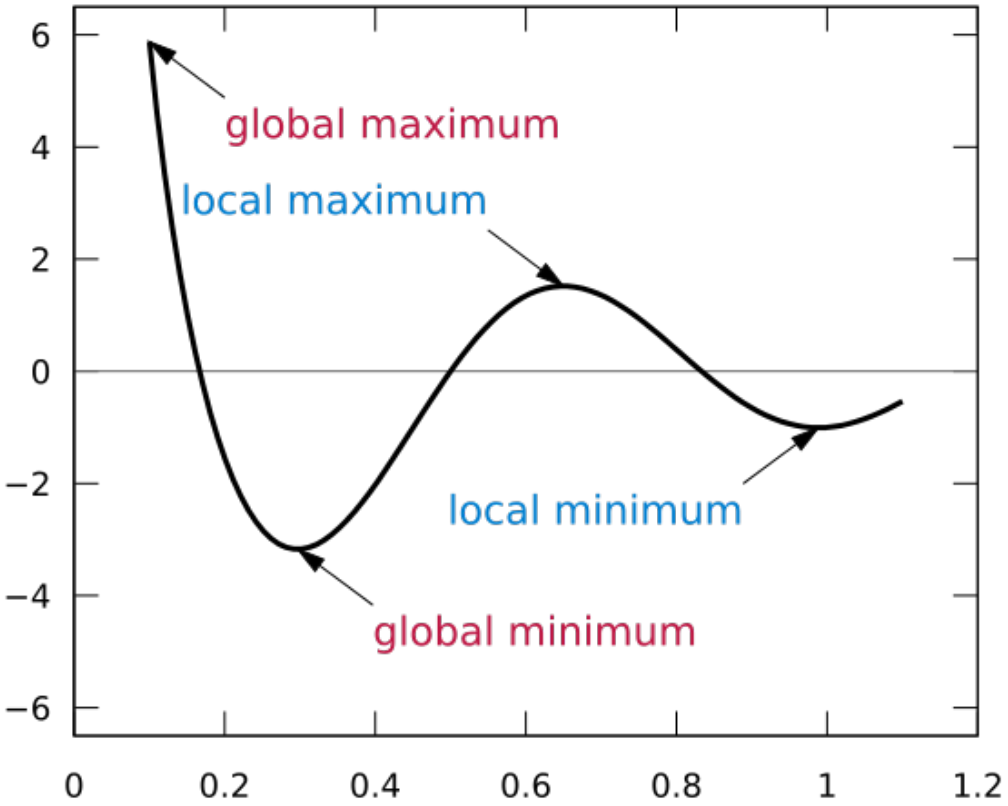
- *Accurate*: It captures the relevant physical processes with sufficient accuracy.
- *Simple*: It focuses on the essential processes. Details are left out. Optimizing for details is a bad idea, considering the inaccuracies inherent in any inflow forecast. Less = more.
- *Quick*: As it will need to be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

Prediction model



Deltares

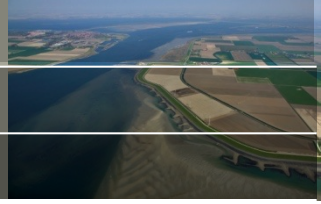
Local and global optima



Source: Wikipedia. GFDL 1.2.

Deltares

Key to reliable operational optimization



Idea: Formulate optimization problems that only admit global minima.

The mathematical term for such formulations is that they are convex.

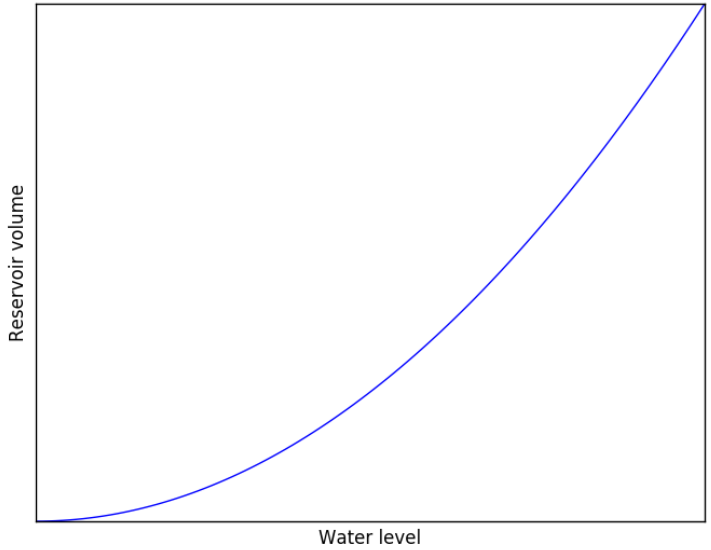
Other packages step around the issue by using linear models.

RTC-Tools goes a step further by finding formulations that accommodate fundamental nonlinearities in the system.

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.



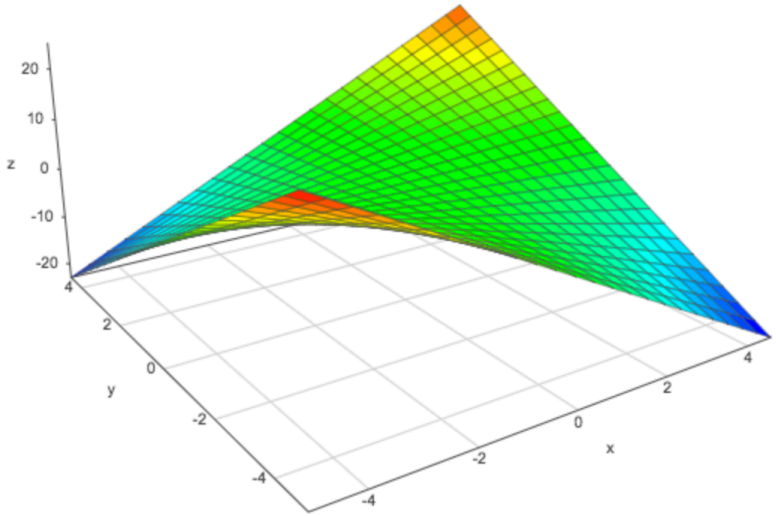
- However, accounting of volumes is linear:

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

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

Nonlinearity #2: 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

Interdisciplinary modeling: Modelica

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

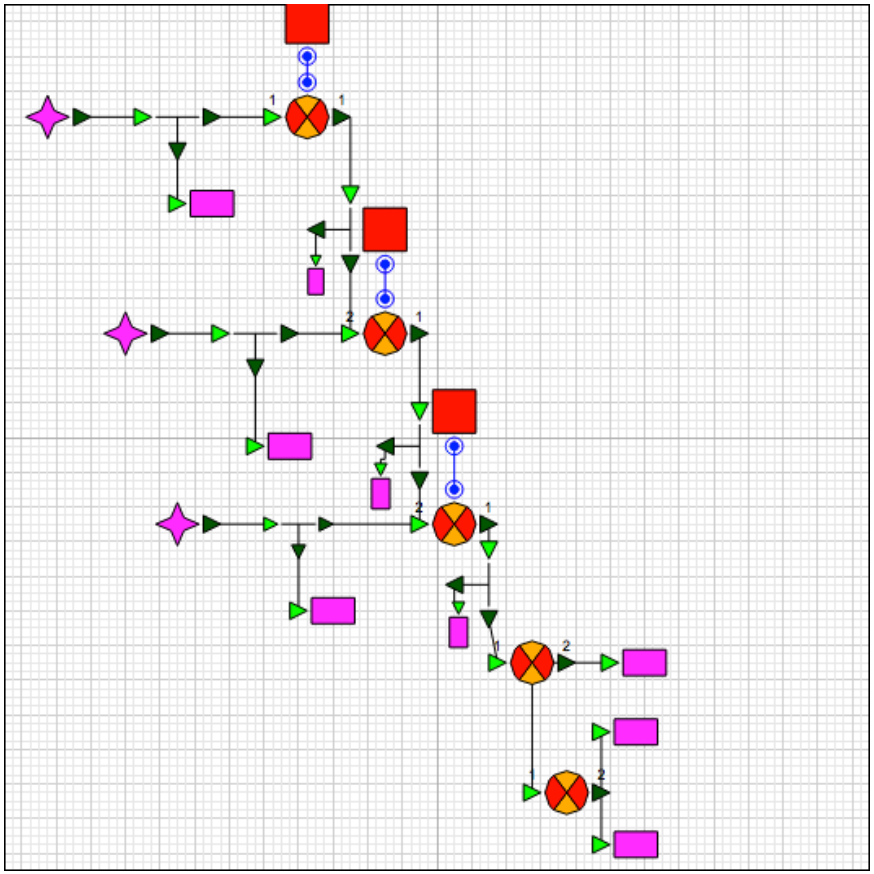
- Open standard
- Independent of application domain
- Widely used in industry

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

N.B. Steady-state init.

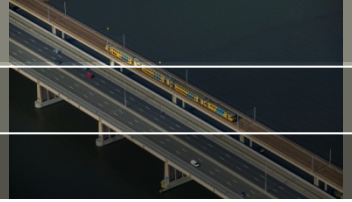
Water allocation in Citarum basin, Indonesia

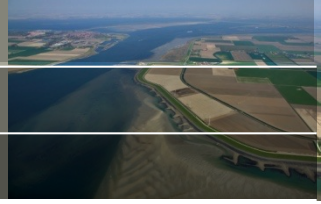
```
model citarum
  import SI = Modelica.SIunits;
  input SI.VolumeFlowRate inflow_Saguling_Q;
  input SI.VolumeFlowRate inflow_Cirata_Q;
  input SI.VolumeFlowRate inflow_Jatiluhur_Q;
  input SI.VolumeFlowRate lateralLoss_Saguling_QLat1;
  input SI.VolumeFlowRate lateralLoss_Cirata_QLat1;
  input SI.VolumeFlowRate 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]) ;
```



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?

Deltares

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

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

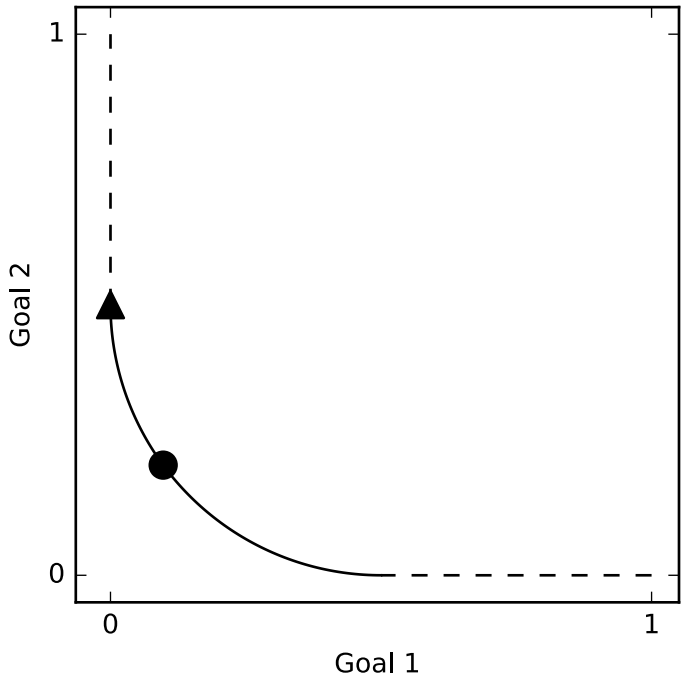
and for all $i \neq j$

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

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

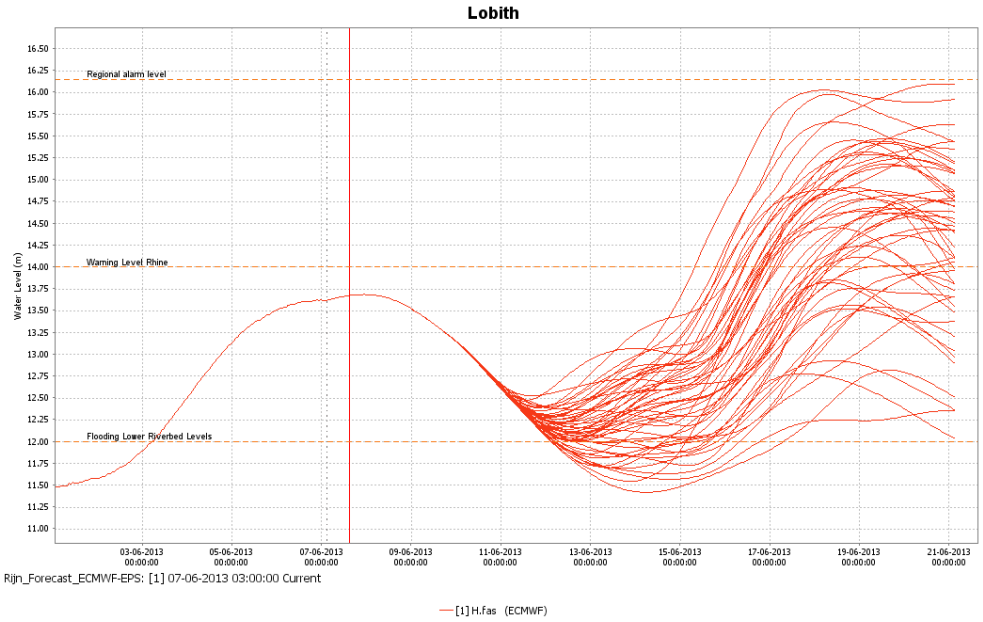
Pareto front

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



Deltares

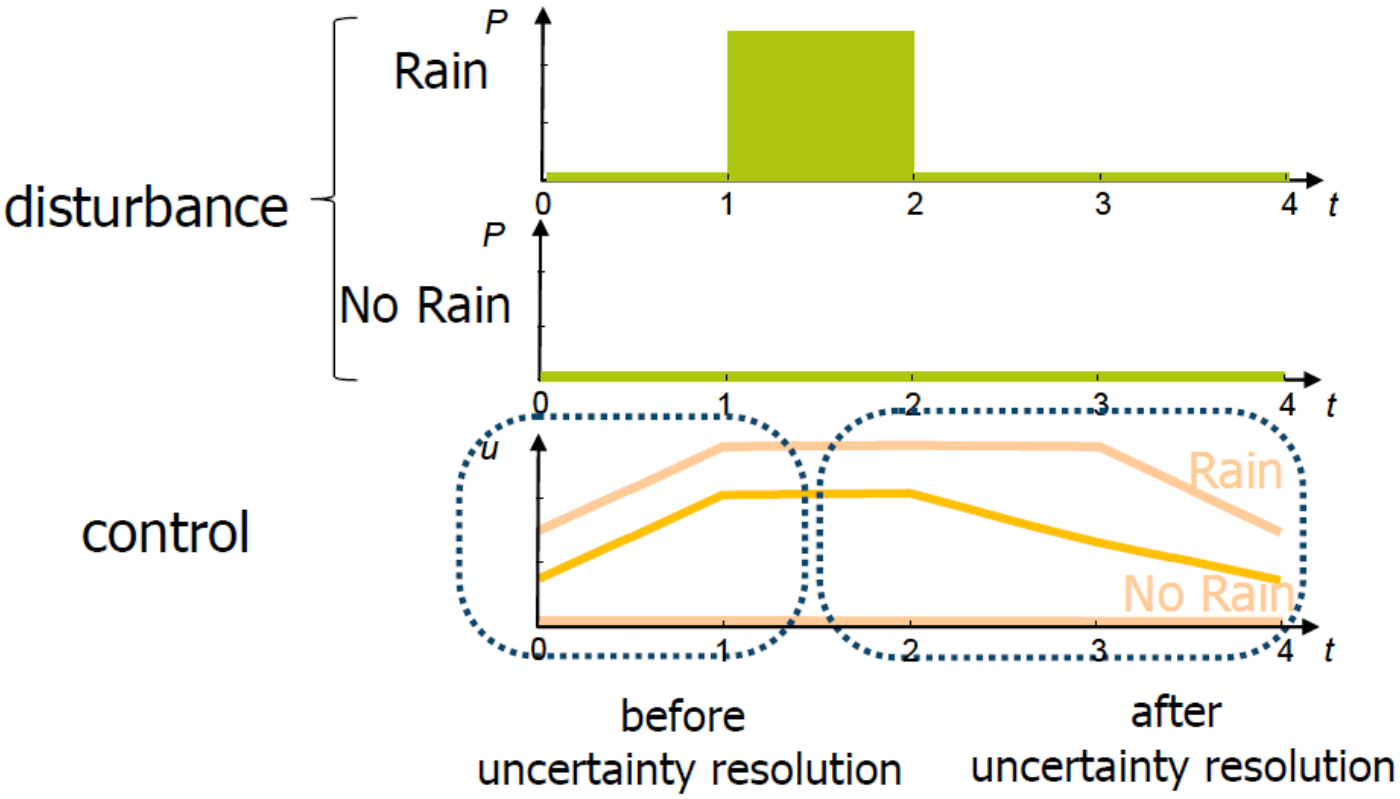
Ensemble techniques



Deltares

Multi-stage Stochastic Oprimization

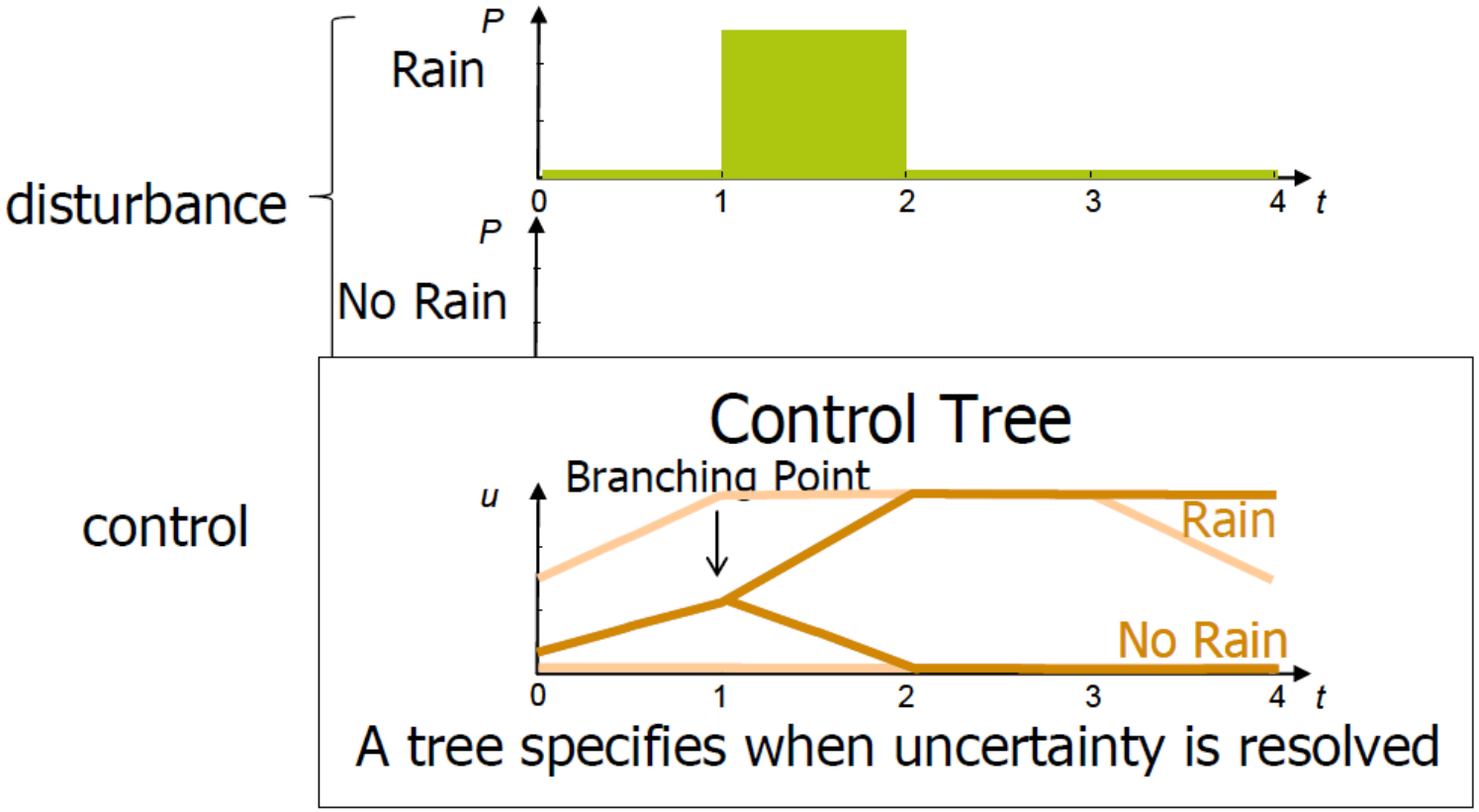
Decision Uncertainty Resolution Decision



Deltares

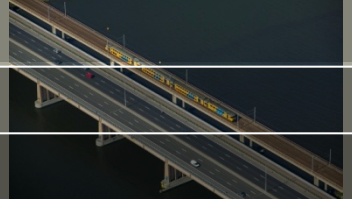
Multi-stage Stochastic Oprimization

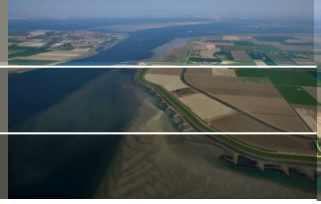
Decision Uncertainty Resolution Decision



Deltares

N.B. ….

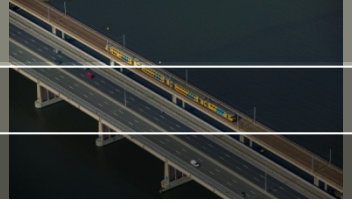
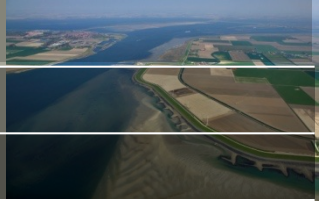




… RTC supports stochastic MPC with multiple model parametrizations or even multiple models in parallel.

All models are tied into the stochastic control tree.

Deltares

Thank you for your attention!

Feel free to contact me any time at

jorn.baayen@deltares.nl

Deltares