RIBASIM2RTC-Tools

programmed by Cheryl van Kempen and Bernhard Becker

documented by Bernhard Becker

# What the toolbox does

The toolbox converts parts of a RIBASIM model into an RTC-Tools 2 model.

Note that manual configuration work needs to be done to get a working RTC-Tools 2 model.

The toolbox contains the following:

* ModelBuilder.py: A Python script that converts a RIBASIM shapefile to an RTC-Tools Modelica stub.
  + Status: ready
  + To do: add an annotation for each object with the coordinates such that the network can be displayed in the OpenModelica Editor.
  + An Example data set is ready.
* A conversion tool for time series
  + Status: to do

# Versions used

RTC-Tools 2.0

RIBASIM 7

Python 2.7 (?)

Code and examples are stored in the OpenEarth repository: https://svn.oss.deltares.nl/repos/openearthtools/trunk/python/applications/RIBASIM2RTC-Tools

# ModelGenerator.py

## Purpose

ModelGenerator.py converts a RIBASIM shapefile to a Modelica model file.

The RIBASIM shapefile is a shapefile that is exported from RIBASIM with the help of the netter feature “export shapefile”.

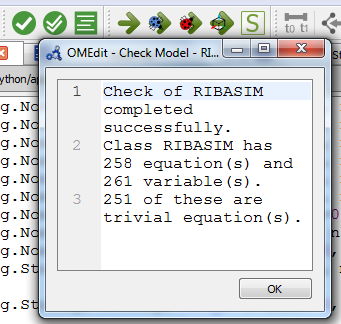
The columns of the shape file are aligned to a so-called Modelica shapefile standard. The Modelica shapefile is a pre-standard for RTC-Tools networks designed by Peter Gijsbers. The Modelica shape file has been designed to generate Modelica input files with the help of a python tool shp\_to\_mo. RIBASIM2RTC-Tools makes use of the Modelica shape file standard and the Python tool shp\_to\_mo, but shp\_to\_mo has been adjusted.

## Input

Input is

* Two shapefiles of RIBASIM standard. These two shapefiles represent the network of the RIBASIM model.
  + One shapefile contains nodes
  + the second shapefile contains the links.
* Two Modelica-shapefiles. Only the column names of these two shapefiles are used.

## Output

* The output is a modelica file \*.mo, which is written into a folder “model”, according to the RTC-Tools folder structure.
* The number of variables and the number of equations does not match. Equations for the reservoir release must be added by the modeler.
* Output can be tested with the Check Model tool from the OpenModelica Connection Editor. The difference between number of variables and number of equations should be explicable, e. g. by the number of reservoirs, which adds one variable for Q\_release per reservoir.   
  

## Source files

* ModelGenerator.py is the main programme
* shp\_to\_mo.py is a programme by Peter Gijsbers that generates a Modelica file based on a Modelica shape file. The Modelica shape file has been designed to generate a Modelica model from. It has been modified for the purpose of RIBASIM shape file conversion:
  + Reservoir submodel has been added
  + in declaration, the el.name property is used, and not the el.id. The reason is that RIBASIM has not necessarily unique ids – the same ID can be used for a node and a branch at the same time. So we assign prefixes and store the new ID under name. The IDs are still used in order to generate the connections, so the ID must not be changed.

# Working with RIBASIM2RTC-Tools

## Preface

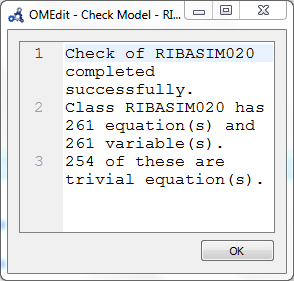
The working steps are explained with the help of the example files in the OpenEarth repository RIBASIM2RTC-Tools.

## Install RTC-Tools

An installation guideline for RTC-Tools can be found in SharePoint:

https://deltares.sharepoint.com/:w:/t/dlt1006/EUKReKfrZ7JEv-OK\_FGBq1oBT\_FJu6cUHpuRG\_3zploaVw?e=DlKIGJ

## Create the network (Modelica model file)

* Export your RIBASIM model to a shape file. Store it under RIBASIM2RTC-Tools\Input\RIBASIM\shp\.
* Run ModelGenerator.py. The run can be carried out via a programming shell like Spyder. This Python script creates a Modelica file \*.mo under \RIBASIM2RTC-Tools\Output\model\. In the current example, the Modelica is RIBASIM.mo.
* Create a Modelica model file that extends the Modelica file that has been generated by ModelGenerator.py (current example: RIBASIM020.mo extends RIBASIM.mo). In this Modelica file (RIBASIM020.mo) add the missing equations such that the number of equations and variables is equal:
  + Time series for control of type fixed=false (because the control is determined by the optimizer)
    - Reservoirs need a time series for the release.
    - Diversion nodes need a time series for one of the outgoing branches.
  + Modify the attributes for the reservoir node
    - the nominal value helps the optimizer, it should be a typical value.
    - min and max should not be too restrictive, so we recommend setting min=0 and max as highest value ever expected. Default values are provided by the ModelGenerator.py.
* Check the Modelica file with the “Check Model” tool . The number of equations and variables must be equal:  
  
* Complete the RTC-Tools folder structure in \RIBASIM2RTC-Tools\Output\:
  + add a folder input
  + add a folder src.

## Prepare the time series (CSV file)

Approach: use TMS time series from RIBASIM and make a CSV file out of it with the help of Excel. An example file is CSV\_Input.xlsm.

Note that the CSV Mixin of RTC-Tools only supports equidistant time steps. For non-equidistant time steps, use the XML file format and PI Mixin.

* Create a time series for all input data under Modelica that have the fixed=true attribute.
* Make sure that the inflow data is in m³/s.

## Prepare the initial state (CSV file)

Create a CSV file and specify the initial state for all objects that need it:

* Volume in the reservoir
* Initial reservoir release
* Initial flow at diversion points

For the current example, the initial states file with states for all reservoir parameters has been prepared with the file Reservoirs.xlsx.

## Prepare the RTC-Tools Python script

Example file: RIBASIM2RTC-Tools\Output\src\RIBASIM.py

### Basic file

Start from the example and make a test run:

* Change the class name (RIBASIM).
* Change the model\_name such that it refers to the right Modelica model.
* Perform a test run. Use the batch file runRTC-Tools2\_Optimization.bat for the current example.
  + Check the log file if an optimal solution has been found.
  + Check the file timeseries\_export.csv output folder, if all expected timeseries are in it and if they have data.

### Volume ranges for reservoirs

* Physical limits are in Table 3.6 in the RIBASIM file Bin2Prt.log. Copy this data into the file Reservoirs.xlsx.
* Check the volume max and volume min attribute for all reservoir nodes in the Modelica file. This attribute is generated automatically by ModelGenerator.py. The physical limits should be within this range.
* Add the physical limits as RangeState Goal in RIBASIM.py.
* Perform a test run.
  + Check if the reservoir volume is within the physical limits. Use Veusz to plot the timeseries. An example file is given with RIBASIM.vsz

### Operational range for reservoirs between firm storage volume and flood control volume

* Use the flood control volume and firm storage volume values from Table 3.6 in Bin2Prt.log
* translate them into a time series (CSV\_Input.xlsm)
* Generate a CSV input file.
* Add RangeStateGoals for the time-variant operational range for the reservoir volume in RIBASIM.py.
* Perform a test run
  + check if the reservoir volume is within the operational range.
  + Check if the reservoir release is plausible. Release limits are given in the Modelica extension RIBASIM020.mo.

### Target volume for reservoirs

* Add the target volume time series to CSV\_Input.xlsm.
* Add RangeStateGoals for the target volume of the reservoir to RIBASIM.py.

### Low flow demands

* Add an output time series in RIBASIM020.mo and add the corresponding equation for low flow nodes.
* Check the Modelica files.
* Perform a test run with RTC-Tools.
* Add a timeseries for the minimum flow to CSV\_Input.xlsm and export it to the input folder for RTC-Tools as timeseries\_import.csv.
* Add a RangeStateGoal for the minimum flow on low flow nodes.
* Perform a test run and check the result. Vary the minimum flow and the priority.

# Optimizatoin results from the example

The example case is a RIBASIM network which represents the upper Nile in a simple way. The network is shown in Figure 1.

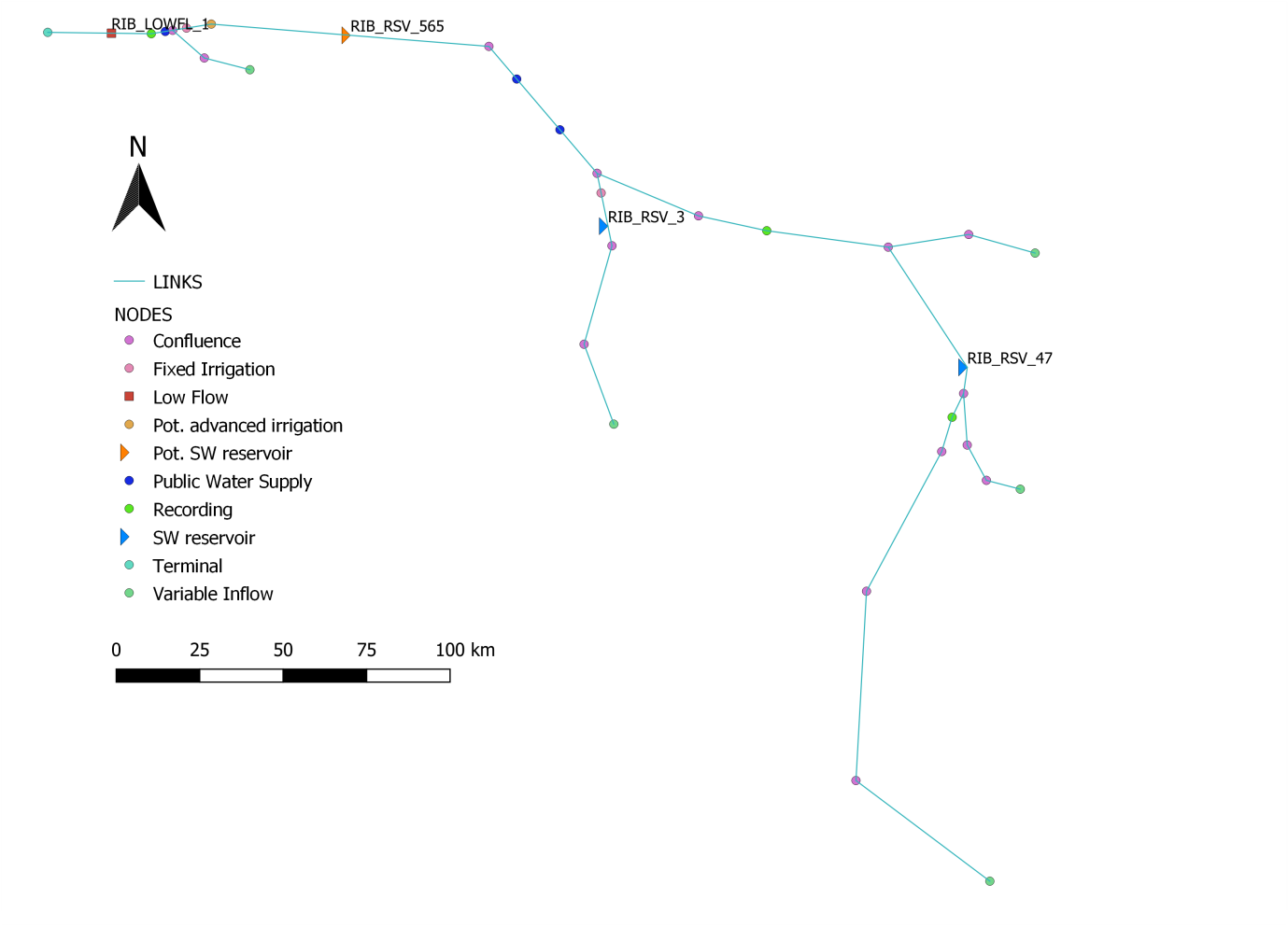


Figure 1: RIBASIM network

Figure 2 shows inflow, reservoir release and storage volume over time for the three reservoirs RSV 3, RSV 47 and RSV 565. The operational range for reservoir volume and the target volume, given as flood control volume and firm storage volume, are indicated as horizontal lines, where target volume is the same as the operational maximum volume. In Figure 3 the discharge for the low flow node LOWFL 1 is given over time.

Priorities are the following:

|  |  |
| --- | --- |
| Priority | Goal |
| 1 | Physical limits of reservoir volume (constant) |
| 20 | Operational range for reservoir volume: firm storage volume to flood control volume (time series) |
| 25 | Minimum and maximum discharge at the low flow node |
| 30 | Operational target for reservoir storage volume (time series) |
|  |  |

Figure 2 shows that the reservoirs operate within the operational range (with some small tolerance). the reservoir operational target is the same as the maximum operational level (flood control volume), so the volume is kept as high as possible. There is an inflow peak in the summer of the year 2011 in all reservoirs. All reservoirs anticipate this peak by reducing the reservoir volume. Release is kept to a virtual maximum such that the release of all three reservoirs does not add up to a discharge that exceeds the maximum discharge at the location LOWFL\_1. At LOWFLO\_1, the minimum and maximum discharge is obeyed at all times (Figure 3). In November 2010 the release is a little bit above the inflow in all reservoirs in order to ensure the minimum flow.



Figure 2: Reservoir inflow, reservoir release and reservoir volume with operational bounds and target (Rule curve) for the three reservoirs in the model.



Figure 3: Discharge and minimum discharge at node LOWFLOW\_1, a maximum has been set to 500 m³/s as well.

# Summary, conclusions and outlook

A tool to convert a RIBASIM schematization from shape file into an RTC-Tools schematization (Modelica file) has been developed. This tool can create an RTC-Tools model schematization automatically. Beside the schematization, time series and the configuration of an optimization problem (Python) are necessary. Excel sheets that assist the modeler and example Python code have been prepared. Having reached this status, the toolbox can be further developed in projects, and RTC-Tools can be used in water resources planning projects as an accompanying tool beside RIBASIM, when optimization is required.

With a test case it has been shown that the toolbox is able to prepare an RTC-Tools shematization and how it can manually extended such that it runs technically and produces plausible results. The system is optimized as whole: both spatially and temporal. The fact that reservoir releases of three reservoirs sum up such that minimum and maximum discharge at the low flow location, which is located downstream of the reservoirs, are always satisfied shows the spatial optimization. The reservoirs’ volume curves over time show that the model is able to anticipate on future inflow peaks. Especially this optimization over the whole temporal domain is an added value with respect to RIBASIM. It makes RTC-Tools suitable for operational systems already now, but can also be useful for system stress tests and strategic planning.

Recommended steps to continue are the following:

* Further develop the example case such that it compares to the RIBASIM model
  + Add a water demand to illustrate how reservoirs contribute to water allocation
  + Add missing time series and features such that a comparison to the original RIBASIM model
* Development of this toolbox
  + Automate the generation of time series
* Required development in RTC-Tools
  + Area-dependent evapotranspiration from a reservoir. The challenge is to formulate this dependency such that it is a convex formulation.
  + Allow non-equidistant time series in CSV input files. It is already possible to use non-equidistant time series in FEWS-PI format, but this format is quite cumbersome for the RIBASIM studies.
  + Capacity to optimize hydropower production. The challenge is also a non-convex formulation of the power generation function.