OpenHPL — Suggested modification and fixes

Bjarne Børresen
bjarne.borresen@multiconsult.no
Multiconsult
P.O.Box7184 Majorstuen
N-0307 Oslo
Norway

April 20, 2023

Preamble

The following note must be considered an input to a discussion and dialog and does not reflect absolute positions. Hopefully, a fruitful discussion accompanied by experimentation and rigorous testing future releases of OpenHPL will be even better and more useful for the community.

1 Introduction

OpenHPL is an open source Modelica package for simulation of hydropower plants and similar installations. The package was originally developed by Liubomyr Vytvytskyi as part of his PhD thesis [3] and later extended by xxx and yyy. From January 2023 the OpenHPL package was released under the Mozilla Public License v2.0 [2]. The purpose of this note is summarize some suggested modifications and improvements in order to further develop the usefulness of the library.

2 Basic design principle

The Zen of Python [1] is a collection of statements trying in concise way to summarize what is considered the *pythonic* approach to solving programming tasks. There are to date no such eloquent and humorous summary of the *modelica way* to design and implement packages. In the following some ideas and suggestions for the best practices are summarized. These are

- The most common usage is the default behavior
- Minimize necessary input to create a runable model
- Hide complex options, only show normal usage as default
- Use inheritance to ensure consistent use and easy maintenance
- Use commmon Modelica best practice for handling properties
- Avoid side effects of models/elements
- All elements should have test function/test module (unit tests)
- Examples package should only contain workable, realistic cases (move function testes into separate package)

In the following some more discussion regarding implications and suggestions related implementation and possible changes are listed in some more detail. The main goal should be to develop a library that is easy to use for the most common cases and at the same time permits extension to complex, tailor made models when considered convenient.

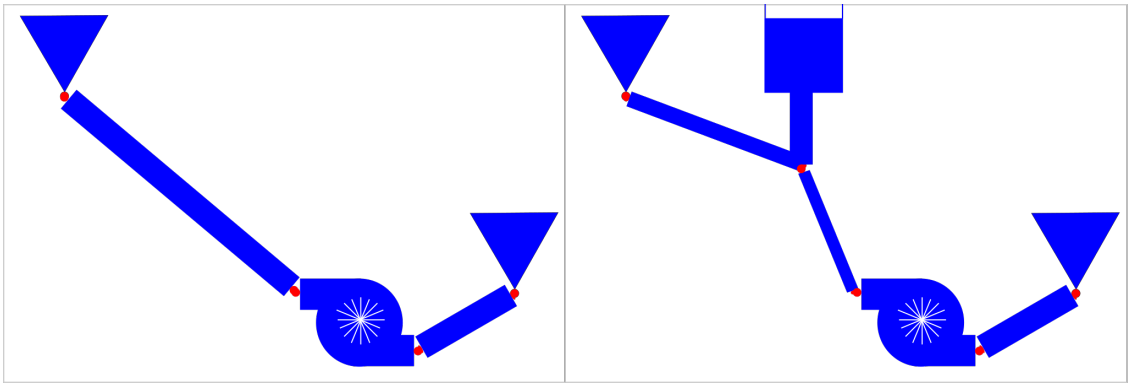

Figure 1: Example of two common, simple configurations for powerplants.

2.1 The most common usage is the default behavior

There are many examples of models where the most common cases covers more than 90% (or maybe 99%) of expected instances. Examples of this are

- Constant level reservoirs
- Constant area tunnels and pipes
- Constant area channels
- Models ignoring singular losses
- "Standard" turbine types (Pelton, Francis and Kaplan)

2.2 Minimize necessary input to create a runable model

This point is closely related to the above statement. The goal should be be able to set up a working model with limited amount of input, while at the

Component	**Parameters**
Reservoir	Absolute water level
Tunnel and pipes	Length, Area, Friction factor (or type)
Channels	Length, Total area, Friction factor
Turbines	Type, Head, Power

Table 1: Summary of suggested minimum (default) input for models

Steady state start should be the default behavior For most transient simulation the starting point will be a steady state condition. Examples of such cases are

1. Sudden load rejection form steady operating point
2. Sudden valve closure

To ensure this, the dynamic part of component models should define the initial equation $\frac{dy}{dt} = 0$ for all y's in the model. In order for this to work properly it is important that the initialization is tested for all components.

$$\Delta p = f \cdot \rho \cdot \left(\frac{L}{D}\right) \frac{v^2}{2}$$

Complex network

2.3 Hide complex options, only show normal usage as default

Listing 1: Dialog with option can be used to hide complex behaviour

```
ModelicaReference.Annotations.Dialog
```

2.4 Use inheritance to ensure consistent use and easy maintenance

One important design choice related to the previous point is how to hide or implement complexity. For most users it may be sufficient to hide the complexity in the GUI by using

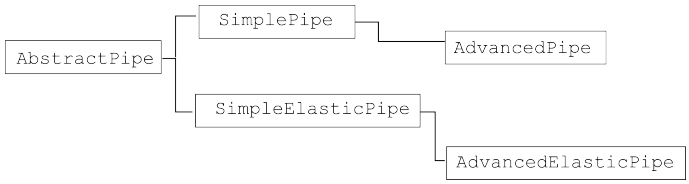

Figure 2: Example of inheritance diagram for the different pipe models.

2.4.1 Structurally identical models should be implemented on the same basis

This is related to the

3 Treatment global data

Global data is currently handled by the `inner` and `outer` construct.

Listing 2: Calling of

```
model ExamplePlant
    inner OpenHPL.Data data(SteadyState = true, Vdot_0 = 75)
```

```
    equation
end ExamplePlant;

within OpenHPL.Waterway;
model Pipe "Model of the pipe"
    outer Data data "Using standard data set";
extends OpenHPL.Icons.Pipe;

end Pipe;
```

Listing 3: Global data in OpenHPL is implemented as a record.

```
within OpenHPL;
record Data "Provides a data set of most common used settings"
extends Modelica.Icons.Record;
    parameter SI.Acceleration g = Modelica.Constants.g_n "Gravity constant"
    annotation (Dialog(enable=false, group = "Constants"));
    parameter Real gamma_air = 1.4 "Ratio of heat capacities at constant pressur
    annotation (Dialog(enable=false, group = "Constants"));
    parameter SI.Pressure p_a = 1.013e5 "Atmospheric pressure"
    annotation (Dialog(group = "Constants"));
    parameter SI.MolarMass M_a = 28.97e-3 "Molar mass of air at STP"
    annotation (Dialog(group = "Constants"));
    parameter SI.Density rho = 999.65 "Water density at T_0"
    annotation (Dialog(group = "Waterway properties"));
    parameter SI.DynamicViscosity mu = 1.3076e-3 "Dynamic viscosity of water at
    annotation (Dialog(group = "Waterway properties"));
    parameter SI.Height p_eps = 5e-2 "Pipe roughness height"
    annotation (Dialog(group = "Waterway properties"));
    parameter SI.Compressibility beta = 4.5e-10 "Water compressibility"
    annotation (Dialog(group = "Waterway properties"));
    parameter SI.Compressibility beta_total = 1 / (rho*1000^2) "Total compressib
    annotation (Dialog(group = "Waterway properties"));
    parameter Boolean SteadyState=false "If checked, simulation starts in steady
    annotation (choices(checkBox = true), Dialog(group="Initialization"));
    parameter SI.VolumeFlowRate Vdot_0=19.077 "Initial volume flow rate through
    annotation (Dialog(group="Initialization"));
    parameter SI.Frequency f_0 = 50 "Initial system frequency"
end Data;
```

4 Treatment of elevation and pressure

When calculating a hydropower system and other hydraulic systems, both the global pressure and pressure difference as well as the local values are of interest. Consider the two systems illustrated in figure 3. Assuming that the elevations and pipe parameters are identical, the two models are dynamically equivalent. However, the local pressure at the inlet of the pipe will strongly depend on the local elevation z_1. The total hydraulic pressure or piezometric pressure p_h can be written as

$$p_h = p_s + \rho g z$$

where p_s is the (local) static pressure and z the local elevation. By storing the local elevation z at the contact nodes (ports) it is easy to convert between the various forms (see listing below). In addition the difference of elevation can be removed from various components, such as pipe.

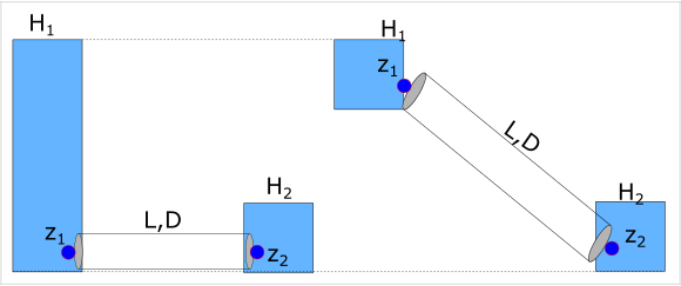

Figure 3: Two tanks are connected by a pipe. Assuming that the elevations and pipe parameters are identical the two models are dynamically equivalent.

Listing 4: Suggested interface Contact with phyiscal elevation added

```
within OpenHPL.Interfaces;
connector Contact "Water flow connector"
    SI.Pressure p "Contact pressure";
    SI.Lengthn z "Pysical elevation of node";
    flow SI.MassFlowRate mdot "Mass flow rate through the contact";
end Contact;
```

5 Abstract classes

The use of abstract classes (partial classes in Modelica terminology) is a useful approach for collecting common parts of multiple detailed models. As an example,

AbstractPipe basis class for both rigid water column pipe as well as elastic water pipe

AbstractValve basis class for different types of valves

AbstractTurbine basis class for different turbine types

6 Waterway

6.1 Reservoir

Normally when considering transients, changes in reservoir levels is not considered as the time scale of the reservoir changes is much larger than the typical time scales of the transient. Thus default mode should be an infinite reservoir.

```
    parameter Boolean infiniteReservoir=true "If checked, waterlevel is kept con
```

Question – can symbols/iconography on the model be made conditional?

References

[1] Pep 20 - the zen of python, 2004.

[2] Mozilla public licenseversion 2.0, 2012.

[3] Liubomyr Vytvytskyi. *Dynamics and model analysis of hydropower system.* PhD thesis, Univeristy of South-Eastern Norway, 2019.