Linearization for Analysis of a Hydropower Model using Python API for OpenModelica

Liubomyr Vytvytskyi Bernt Lie

Department of Electrical engineering, Information Technology and Cybernetics, University of South-Eastern Norway, Porsgrunn, Norway, {Liubomyr.Vytvytskyi,Bernt.Lie}@usn.no

Abstract

Even though almost all processes in the real world are described by nonlinear models, nonlinear theory for analysis of these models is far less developed than the theory for linear models. Therefore model linearization is important in order to make efficient analysis tools for these models.

This paper describes the possibility of automatic linearization in Python for a hydropower system modeled in OpenModelica using our in-house hydropower Modelica library *OpenHPL*. Linearization is made using a Python API. Simple uses of the linearized model for analysis and synthesis are indicated.

Keywords: linearization, hydropower, Python API, OpenModelica

1 Introduction

1.1 Background

A transition towards more renewable energy sources is currently taking place in Europe and all over the world. This situation leads to increase in the use of flexible hydropower plants to compensate the highly changing production from intermittent energy sources such as wind and solar irradiation. A high head hydropower system is considered for this study, because it takes up the main part of all hydropower plants in Norway.

The possibility for modelling and simulating this hydropower system takes an important role in order to make efficient analysis tools for testing a designed controller for stability and performance in different operating regimes. One of such analysis tools can provide automatic linearization; an approximate linear model allows for the use of linear theory for analysis and synthesis which is much better developed than nonlinear theory.

1.2 Previous Work

Basic mathematics and control theory needed to model, analyze, and design feedback systems are provided in (Åström and Murray, 2010). Based on these methods, the linearized hydropower model can be further tested and analyzed for control purposes using a Python[1] package — *python-control* (The Python Control Systems Library)[2]. *python-control* is a Python module, where basic operations for analysis and design of feedback control systems are implemented.

A Python API[3] for OpenModelica[4] already exists that provides possibilities for controlling simulations of the OpenModelica models via Python (Lie et al., 2016). Python in turn gives much wider possibilities for plotting, analysis, and optimization (e.g., using Python packages *matplotlib*, *numpy*, *scipy*, etc.).

Some work on modeling a waterway for the high head hydropower system together with a generator, the Francis turbine, and a governor, has already been done using OpenModelica (Vytvytskyi and Lie, 2017, 2018). Unit models have been assembled in our in-house Modelica[5] library *OpenHPL*[6].

1.3 Overview of Paper

In this paper, the main contribution is investigation of how modern computer tools can make the workflow of analysis and design, including linearization and linear control analysis/design. This is the first paper that demonstrates how linearization can be done using the Python API for "non-academic" hydropower models of different complexity. Model implementation is done in OpenModelica using the *OpenHPL* library.

The paper is structured as follows: Section 2 gives a system description of a high head hydropower system. Section 3 gives an overview of the modeling tools and a presentation of the hydropower model. Then automatic linearization and a simple PI controller design are described in Sections 4 and 5. Finally, discussion and conclusions are given in Section 6.

2 System Description

High head plants typically collect and store water in reservoirs in mountains, with tunnels leading the relatively small flow of water down a considerable height difference to the aggregated turbine and generator. The electricity, produced by the generator, is then transferred through power lines to consumers. A typical structure for the high

[1] https://www.python.org

[2] https://goo.gl/MtbYtf

[3] https://goo.gl/Qyjqq2

[4] https://openmodelica.org

[5] https://www.modelica.org

[6] Open Hydro Power Library is developed by the first author within his PhD study.

https://doi.org/10.3384/ecp18153216

Proceedings of The 59th Conference on Simulation and Modelling (SIMS 59), 26-28 September 2018, Oslo Metropolitan University, Norway

head hydropower plant is depicted in Figure 1 (Vytvytskyi and Lie, 2017).

For simulations in this paper, the data from the Sundsbarm hydropower plant in Telemark, Norway is used with data provided in (Winkler et al., 2011), see Table 1 and 2.

Table 1. The waterway geometry of Sundsbarm hydropower plant.

Waterway unit	*Height difference, m*	*Length, m*	*Diameter, m*
Reservoir	48	—	—
Conduit	23	6600	5.8
Penstock	428.5	600	3
Surge tank	120	140	3.4
Discharge race	0.5	600	5.8
Tail water	5	—	—

Table 2. The turbine geometry of Sundsbarm hydropower plant.

Turbine type	*Nominal head, m*	*Nominal flow rate, m^3/s*	*Nominal power, MW*
Francis	460	24.3	104.4

3 Modeling

3.1 Modeling Tools

All modeling is done in OpenModelica, which is an open-source Modelica-based modeling and simulation environment intended for industrial and academic usage[7].

For modeling the hydropower system, library *OpenHPL* is used. This is an in-house hydropower library, where different parts of the waterway components, such as reservoir, conduit, surge tank and turbine, have been assembled. In this library, different waterway components of the hydropower system are described by both mass and momentum balance, and could include compressible/incompressible water or elastic/inelastic pipe walls. A better overview of the mathematical models and methods used in this library is giving in (Vytvytskyi and Lie, 2017; Splavska et al., 2017).

In addition, our hydropower library can also be connected with other open source Modelica libraries such as *OpenIPSL*[8] (Open-Instance Power System Library), where a much wider variety of power system components are presented. Together, the *OpenHPL* and *OpenIPSL* libraries give a possibility to develop a model for the whole hydropower system that starts from the water in the reservoir and ends with the different electrical loads. Linearization also works for more complex/detailed models than used here (e.g., a model for the whole hydropower system), but space limitations restrict our presentation to simpler cases.

[7]Some tutorials exist for Modelica — `http://book.xogeny.com`, and OpenModelica — `https://goo.gl/76274H`

[8]`http://openipsl.readthedocs.io/en/latest`

3.2 Model Presentation

In this study, two cases of complexity for this system are considered:

1. Simplified system with incompressible water and inelastic pipe.

2. More complex system that includes water compressibility and pipe shell elasticity in the penstock.

Both these cases are straightforward to implement in OpenModelica using the *OpenHPL* library. A block diagram that is relevant for both cases of the hydropower system is presented in Figure 2. For simplicity, the water levels in reservoir and tail water are considered to be constant.

In both cases, the model has one input — turbine gate opening — u_{tr}, and one output — turbine volumetric flow rate — $\dot{V}_{tr}$.

4 Linearization

4.1 Overview

The Python API (Lie et al., 2016) provides a linearization function that allows approximation of nonlinear DAE models in OpenModelica to linear state space models in Python.

First, the Modelica model is instantiated in Python using the *OMPython* package and the following command:

```
hps_s=ModelicaSystem("OpenHPL.mo","
    OpenHPL.Tests.HPLiniarization","
    Modelica") // for simpler model
hps_kp=ModelicaSystem("OpenHPL.mo","
    OpenHPL.Tests.HPLiniarizationKP","
    Modelica") // for complex model
```

After this, the input signal and simulation options are set in Python for the simulation. Before linearization, the model parameters are set to steady state values. Automatic linearization is done from Python, where the matrices for the general state-space representation of a linear system are given using the following command:

```
As,Bs,Cs,Ds = hps_s.linearize() // for
    simpler model
Akp,Bkp,Ckp,Dkp = hps_kp.linearize() // for
     complex model
```

4.2 Simple Model

First, the simple hydropower model is linearized. Through linearization, the state-space matrices A, B, C and D are generated:

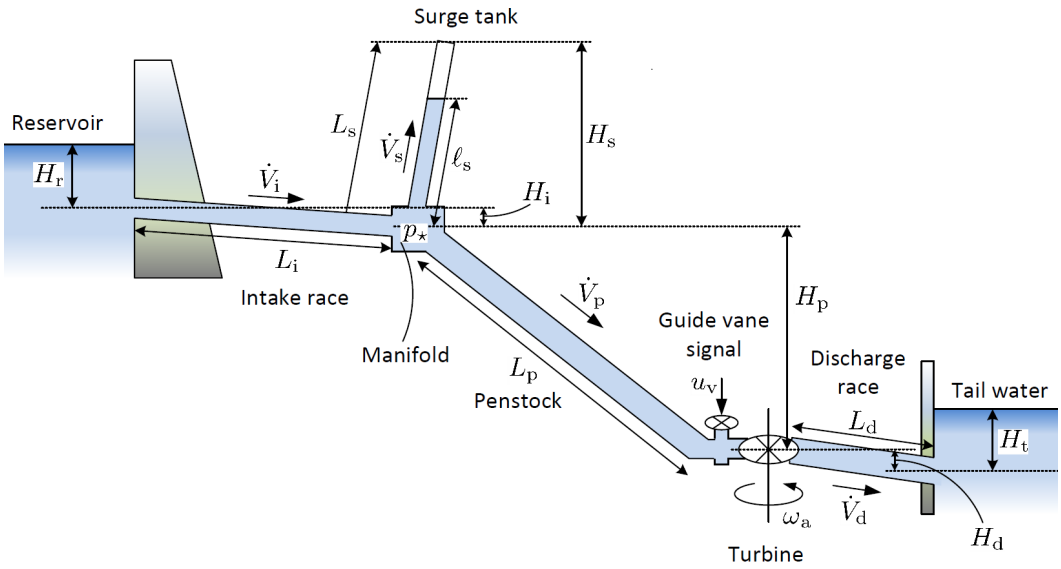

Figure 1. Structure of the high head hydropower plant.

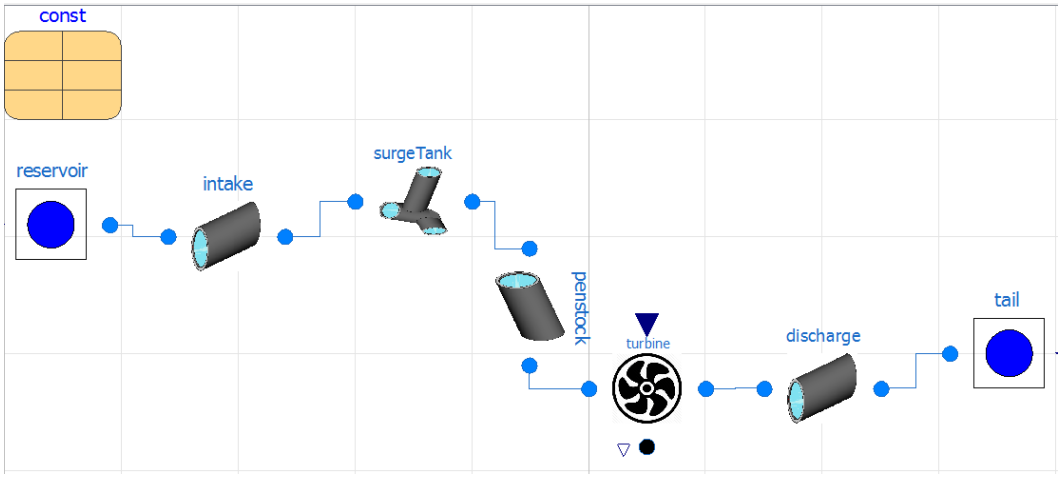

Figure 2. Model of the hydropower system.

$$A = \begin{bmatrix} -4.4 & 3.4 \cdot 10^{-4} & 7.7 \cdot 10^{-6} \\ 4.2 & -4.0 \cdot 10^{-3} & -1.1 \cdot 10^{-5} \\ 0.0 & 997 & 0.0 \end{bmatrix} \quad (1)$$

$$B = \begin{bmatrix} 110.45 \\ -106.62 \\ 0.0 \end{bmatrix} \quad (2)$$

$$C = \begin{bmatrix} 1.0 & 0.0 & 0.0 \end{bmatrix} \quad (3)$$

$$D = \begin{bmatrix} 0.0 \end{bmatrix} \quad (4)$$

Information about the state, input and output variables and their order for the linearized simple model can checked in Python using the following commands:

```
hps_s.getLinearInputs() // for inputs
hps_s.getLinearOutputs() // for outputs
hps_s.getLinearStates() // for states
```

As mentioned above, this model has one input $u = u_{tr}$ and one output $y = \dot{V}_{tr}$. The linearization algorithm gives a state vector with 3 elements: $x = [\dot{V}_p, \dot{V}_s, m_s]^T$. Here, $\dot{V}_p$ and $\dot{V}_s$ are the volumetric flow rates in the penstock and surge tank respectively, m_s is the water mass in the surge tank. The linearization algorithm has actually considered two more states (the water masses in the reservoir and tail water). However, due to assumption of the constant water level in those compartments, their rows in the A matrix are zero vectors and can be neglected.

As seen, for this simpler case the linear model is of low order. It is also known that the system is asymptotically stable if all eigenvalues of the A matrix have negative real parts. Using the following command from the *numpy* package, we find the eigenvalues:

```
linalg.eig(As)
```

The eigenvalues of A matrix are as follows:

$$eig(A) = \begin{bmatrix} -4.367 \\ -0.003 + 0.06j \\ -0.003 - 0.06j \end{bmatrix} \quad (5)$$

4.3 Complex Model

Next, the more complex hydropower model has been linearized in the same way as was presented for the simpler case. The state-space matrices are presented below in simplified form due to their shape:

$$A = \begin{bmatrix} 4.17 & \dots & 0.0 \\ \vdots & \ddots & \vdots \\ 0.0 & \dots & 0.0 \end{bmatrix} \in \mathbb{R}^{22\times 22} \quad (6)$$

$$B = \begin{bmatrix} 0.0 \\ \vdots \\ 0.0 \end{bmatrix} \in \mathbb{R}^{22\times 1} \quad (7)$$

$$C = \begin{bmatrix} 0.0 & \dots & 0.0 \end{bmatrix} \in \mathbb{R}^{1\times 22} \quad (8)$$

$$D = \begin{bmatrix} 0.036 \end{bmatrix} \quad (9)$$

The inputs, outputs and states for the linearized complex model are also provided using the following commands:

```
hps_kp.getLinearInputs() // for inputs
hps_kp.getLinearOutputs() // for outputs
hps_kp.getLinearStates() // for states
```

The input and output are the same as for the simpler case. However in this case, the model consists of 22 states that make it more space demanding, $x = [\dot{m}_{p,i}, p_{p,i}, \dot{V}_s, m_s]^T$. Here, two states are also relevant for the surge tank: the volumetric flow rate, $\dot{V}_s$, and the water mass, m_s. On the other hand, the penstock now is described by 20 equations — 10 for the mass flow rate — $\dot{m}_{p,i}$) and 10 for the pressure — $p_{p,i}$ (here, i is a cell number in range from 1 to n, where n is a number of discretization points of the penstock). This is due to using the Finite Volume method for the discretization of the more complex model with compressible water and elastic pipe walls (the penstock is divided in ten cells here).

In the same way as it was done for the previous simpler case, the eigenvalue analysis of A matrix could be performed. We found that this more complex system is also asymptotically stable.

4.4 Bode Plot Comparison

After the hydropower model has been linearized and the (A,B,C,D) matrices for the general state-space representation are defined for the two cases, some further analysis for the linearized system might be done. For control purposes, the frequency response of a system (Bode plot) can be interesting.

To plot this frequency response, the following commands from the *python-control* package in Python can be used:

```
sys = ss(A,B,C,D)
mag, phase, omega = bode_plot(sys, dB=True)
```

As an alternative, the transfer function $H(s)$ of the system could be found from:

$$H(s) = C(sI - A)^{-1}B + D \quad (10)$$

Here, s is the Laplace operator and for the frequency response, define $s = j\omega$, where ω is frequency in radians. After this, the Bode plot for the linearized hydropower system can be plotted. The Bode diagram for the two cases of the linearized hydropower model are shown in Figure 3.

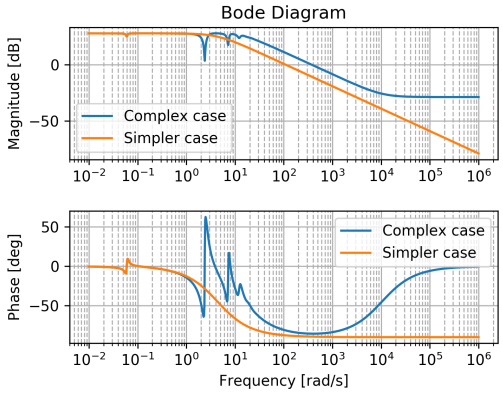

Figure 3. Comparison of Bode plot for two cases of the hydropower system.

5 Design of PI Controller

Using the *python-control* package in Python, a simple PI controller for the linearized hydropower models could be designed and tuned.

First, the step response of the control signal for the linearized hydropower model is found for the two cases, using the following command:

```
sys_s = ss(As,Bs,Cs,Ds) // simple system
youts, Ts = step(sys_s)
sys_kp = ss(Akp,Bkp,Ckp,Dkp) // complex
    system
youtkp, Tkp = step(sys_kp)
```

The results of the control signal step response for both the simple and the complex linearized models are shown in Figure 4.

After this, a PI controller $C_r(s) = \frac{K_p s + K_i}{s}$ is tuned.

Then, the controller transfer function is defined in Python using the control package and connected to the hydropower system via feedback using the following commands:

```
CrPI = tf([[[Kp, Ki]]],[[[1., 0.]]])
Trs = feedback(sys_s*CrPI,1)
youts, Ts = step(Trs)
Trkp = feedback(sys_kp*CrPI,1)
youtkp, Tkp = step(Trkp)
```

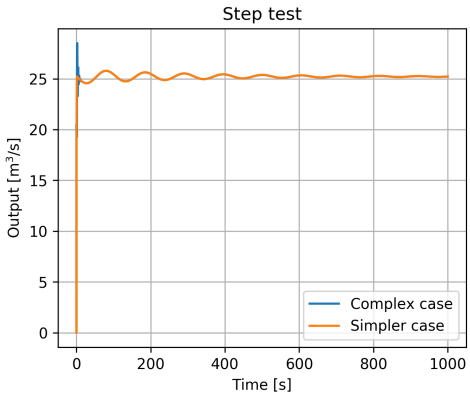

Figure 4. Comparison of the step response for the simple and complex linearized models.

The results of the step response for the reference value for the PI controller that control the hydropower system for the two cases are shown in Figure 5 and Figure 6. For the two cases, the step response is done for two sets of controller parameters.

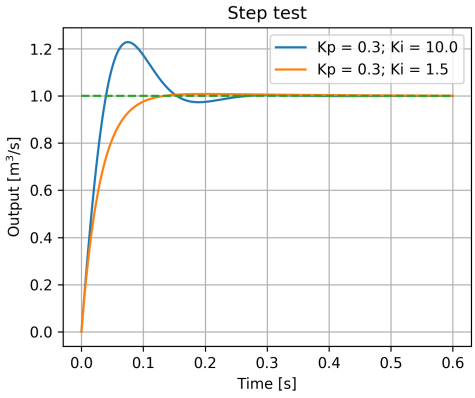

Figure 5. Step response for the simpler model with PI controller.

Finally, the designed and tuned PI control could be checked on the original (nonlinear) hydropower model in OpenModelica. The results of the step test for the output and input are shown in Figure 7 — for the simpler case and Figure 8 — for the more complex case.

6 Discussion and Conclusions

The possibility of automatic linearization of OpenModelica models through Python using the Python API has been presented in this paper.

Two cases with different model complexity for the hydropower system have been linearized in order to show the linearization capability of the Python API. Despite the model complexity, the linearization algorithm finds the state space matrices A, B, C, D.

After linearization, linear theory could be further used for the model analysis and synthesis. Examples of analysis has been presented by creating a Bode plot and designing

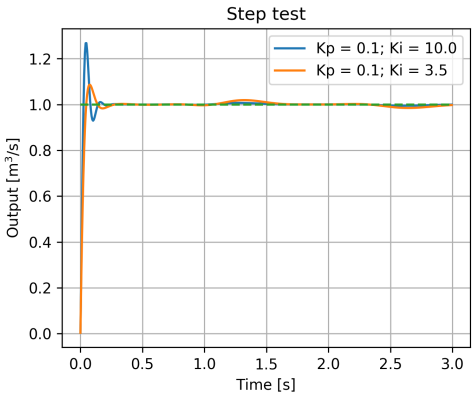

Figure 6. Step response for the complex model with PI controller.

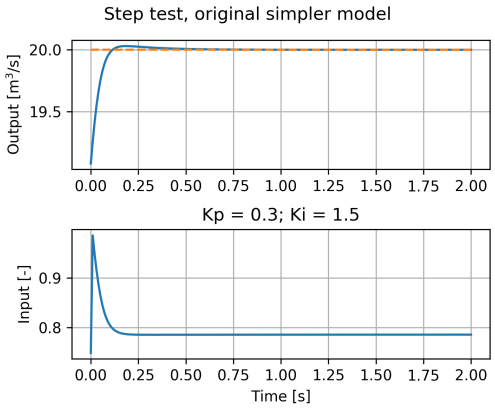

Figure 7. Step test for the nonlinear hydropower model, simpler case.

a simple PI controller, using the already exist packages in Python (*numpy* or *python-control*). The Bode diagram has been plotted for two cases of the hydropower system in order to show the frequency response of the models. Then the possibility of PI controller design has been shown for the two cases. The designed PI controller has also been tested for the original (nonlinear) models in OpenModelica.

Besides the presented examples of linear analysis, many more other possibilities for analysis and synthesis of the linearized model might be used, such as sensitivity or stability analyses, etc.

References

Karl J. Aström and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2010.

Bernt Lie, Sudeep Bajracharya, Alachew Mengist, Lena Buffoni, Arun Kumar, Martin Sjölund, Adeel Asghar, Adrian Pop, and Peter Fritzson. API for Accessing OpenModelica Models From Python. *Proceedings of EuroSim 2016, Oulu, Finland*, 2016.

Valentyna Splavska, Liubomyr Vytvytskyi, and Bernt Lie. Hy-

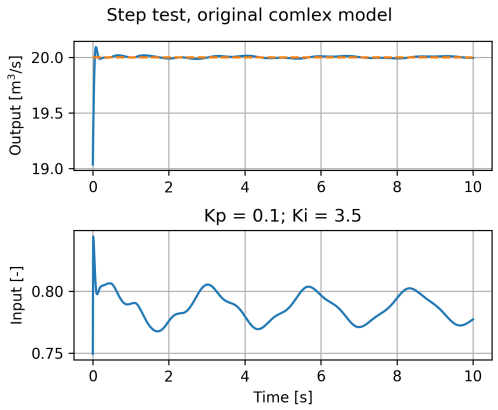

Figure 8. Step test for the nonlinear hydropower model, complex case.

dropower Systems: Comparison of Mechanistic and Table Look-up Turbine Models. *Proceedings of 58th SIMS Conference, Reykjavik, Iceland*, 2017.

Liubomyr Vytvytskyi and Bernt Lie. Comparison of elastic vs. inelastic penstock model using OpenModelica. *Proceedings of 58th SIMS Conference, Reykjavik, Iceland*, 2017.

Liubomyr Vytvytskyi and Bernt Lie. Mechanistic model for Francis turbines in OpenModelica. *Proceedings of Mathmod conferance, Wien, Austria*, 2018.

Dietmar Winkler, Hege M. Thoresen, Ingvar Andreassen, Magamage A. S. Perera, and Behzad R. Sharefi. Modelling and Optimisation of Deviation in Hydro Power Production. In *Modelica Conference*, 2011.