Development of OpenModelica interface for external control

Kunihiro Matsuzawa [1)] Hajime Sato [1)] Takashi Yamashita [1)]

1) AdvanceSoft Corporation
4-3 Kanda Sugugadai, Chiyoda, Tokyo, 101-0062, Japan (E-mail: kmatsu@advancesof.jp)

KEY WORDS:**vehicle development, computer aided engineering, design optimization,** tool, machine learning [B2]

OpenModelica is an open source modeling and simulation environment for the Modelica language suitable for 1DCAE and model-based development.
OpenModelica is a simple modeling and simulation environment, but without the advanced features of commercial MBD tools.
By developing a Python interface library that can control OpenModelica from the outside and combining it with other software, we believe that we can acquire advanced new functions.
In order to control OpenModelica externally, we have developed a new Pyhton interface library OpenModelicaCompilerForPython.
It allows the OpenModelica process to be called through an interface library to externally control the simulation, store the simulation results externally, and read / write model parameters.

Introducing a case study by Bayesian optimization.
We externally controlled OpenModeica from the Bayesian optimization library BayesianOptimization, which can be used free of charge with open source software, and verified it.
For the spring-mass damper model explained in Fig. 1, a search was performed by Bayesian optimization with the spring constant and damper constant as unknown variables (objective variables).
With the correct values of the spring constant and damper constant, which are the objective variables, set to 2.0 [N], 100 cases of "time change of mass position" data were generated by adding random noise to the amplitude of the external force. The objective function was set to minimize the sum of squares of the residuals between the result data at each search point of Bayesian optimization and the data of 100 cases.
Fig. 2 shows the results of Bayesian optimization. After the number of searches is about 70, the value close to the correct answer value is reached. By the number of searches 258 times, the spring constant 1.98593 (correct answer value 2.0) [N / m] and the damper constant 1.974402 (correct answer value 2.0) [N / m] were obtained.

Introducing a case study by reinforcement learning.
We externally controlled OpenModeica from the reinforcement learning frameworks OpenAI Baselines and OpenAI Gym, which can be used free of charge with open source software, and verified it.
Fig. 3 shows the results of reinforcement learning. In the figure on the left, up to about 3000 learning times, the number of trials is large and the rewards and achievements are small, but as the number of learning progresses, the number of trials will be reduced to one or two, and the reward will also be. It has been shown that more will be available. The figure on the right shows the distribution of the spring constant and damper constant selected in the verification when learning was performed 1000 times, 5000 times, and 15000 times. There are more plots as the number of trials increases from the one with less learning. In addition, it can be seen that the spread of distribution narrows as learning increases.

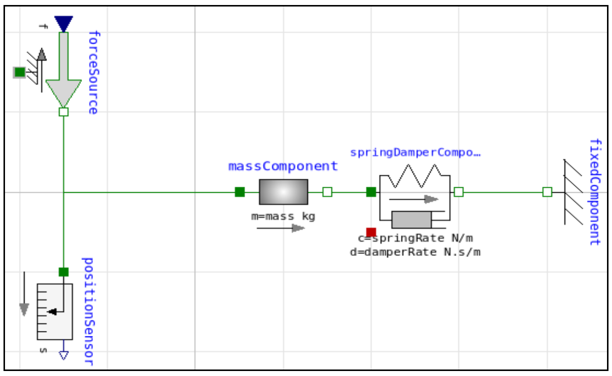

Fig.1 Simple spring-mass-damper model

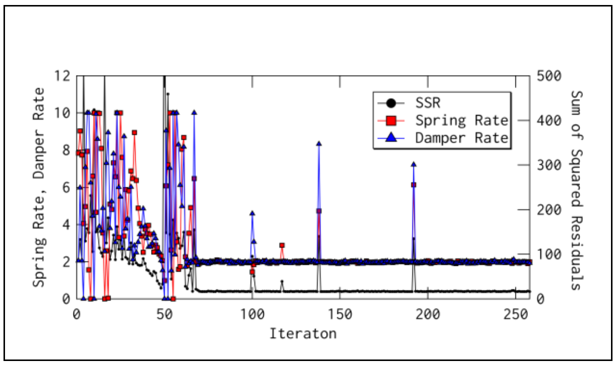

Fig.2 Result of Bayesian optimization

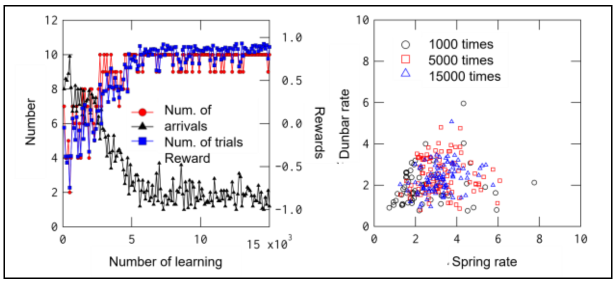

Fig.3 Result of reinforcement learning

By using this Python external control interface, it becomes possible to combine it with other software, and it becomes a highly versatile framework that can acquire advanced enhancements. It is expected to be used in the future, such as improving the efficiency of design and analysis.