XXI IMEKO World Congress "Measurement in Research and Industry"
August 30 – September 4, 2015, Prague, Czech Republic

# HYBRID BACKWARD SIMULATOR TO DETERMINE CAUSAL STATE CHANGES

*Yukio Hiranaka, Shinichi Miura and Toshihiro Taketa*

Yamagata University, Yonezawa, Japan, zioi@yz.yamagata-u.ac.jp

***Abstract*** – We are developing a backward simulator, which determines unknown system input change or internal state change from the system output by using the system model. However, its processing time would increase unlimitedly if the simulation model requires multiple case branching. In some target application, we can use forward simulation module in the backward data flow with significant reduction of processing time. This paper shows an example of such application to determine heater operation from temperature measurement with successful simulation results.

***Keywords***: backward simulation, temperature measurement, inverse problem

## 1. INTRDUCTION

We are studying backward simulator to determine unknown internal state changes or input time function from the system output. The simulator solves inverse problems efficiently by using constraints about and among internal models, inputs and outputs [1,2]. In the preceding works, we showed applicability of backward simulation to dynamic system behavior [3] with case branching [4] and numerical models [5].

The backward simulator is hoped to be a rigorous tool to detect possibility of system bug, fault and undesirable behavior. In the case of analog backward simulation, we should use range signals defined by the minimum and the maximum values to limit the number of cases to be tested. We can control the processing time by setting the number of subdivisions to be appropriate.

Design of backward module is the most important thing and affects the efficiency of the simulation. We found a way to integrate forward simulation modules in the backward simulation. We call it as a hybrid simulator. A previous simulator may consumes huge processing time with multiple case branching, while our hybrid simulator processes in a straight way. We show in this paper principles of the hybrid simulator model, and the result of backward simulation to determine heater operation from the temperature change measured.

## 2. BACKWARD SIMULATION AND MODELS

Here, we explain the concept of backward simulation with an example application of heater operation inference. Fig. 1 shows a structure of forward simulation model for room temperature change caused by an electrical heater. Black dots denote branching points, circles with plus sign denote summation points, and arrows denote the direction of physical information flow. Blocks with $z^{-1}$ means one sampling time delay, which is a member of an integration loop. We adopted two integration loops because the room temperature still increases after the time of heater turned off. The first integration loop corresponds to the neighbor of the heater accumulating calorie locally, and the second loop corresponds to the whole room heat reservoir. The parameters A, B are decay factors due to heat transfer, C is the specific heat of the room converting calorie to temperature. The response can be expressed as $T=a(e^{-bt} - e^{-ct})$.

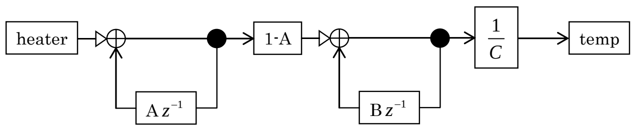

**Figure 1.** Forward simulation model of room temperature.

Figure 2 shows the implemented objects and connection diagram for the simulation of Fig.1. Each obect has corresponding function shown in Fig.1, heater for heating for specified time duration, sum for feedback summing, br for branching with the same output value, dly for one sampling time delay, co for coefficient multiplier, and temp for temperature recording. These objects are coded in Scala class and sum1 and sum2 are instance objects of the class sum, for example. All the objects have indipendent GUI windows which accept local settings and display status and values in each object.

Data which flow on the connection links consist of time and values. As an example, heating power 800W from "heater" at its output port "o" at the time of 10.0s is shown in a XML style UCF (universal communication format) message [6,7]

<sim><s>heater<s>o</s></s><t>10.0</t>800.0</sim>,

where sim is the simulation controller which redirects the message to "i" port of "sum1" as specified in the simulator's connection table [4]. The <s> tag indicates the source of the message. The source tag is nested in this case to mean "o" of "heater.".

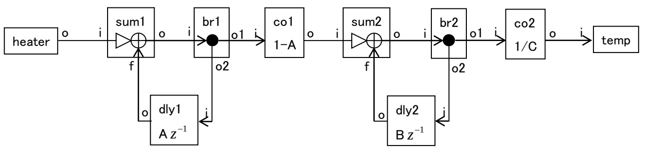

**Figure 2.** Forward simulation objects and connections.

In the backward simulation (Fig. 3), UCF messages flow in backward directions (dashed lines with reversed arrows). The block "temp" is the starting block which sends the time sequence temperature in the reversed order, packing time and temperature in each UCF message. The temperature in the backward simulation is expressed by a value range which expresses the minimum and the maximum temperature as "0.0, 10.0", which means the range from 0.0 inclusive to 10.0 exclusive [3].

By narrowing the range, we can control the resolution of simulation. Simulation parameter ndiv set by the starting block "temp" specifies the number of divisions. As an example, when the whole range is from 0.0 to10.0 and ndiv equals to 10, the value 4.5 is expressed in the UCF message as "4.0, 5.0" as the divided range width is 1.0.

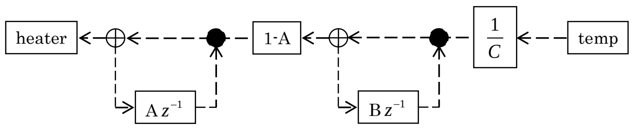

**Figure 3.** Backward simulation model corresponding to Fig.1.

## 3. HYBRID SIMULATOR AND IMPLEMENTATION

Two important features of our simulator, time synchronization and hybrid backward simulation (inclusion of forward simulation modules into the backward simulation) are explained in this section.

Fig. 4 simplifies a forward integration loop in Fig. 1. The summation node must gather the same time data from the two incoming links, the input x(i) and the feedback f(i)=y(i-1) to form the output y(i)=y(i-1)+x(i). Fully synchronized simulator will do the work. However, we intend to perform our simulation in a distributed processing environment, and we preferred asynchronous processing as long as it is possible. Then, the summation node is designed to have input record table which keeps time and value pairs received at the left input port (triangle arrowed port in Fig. 1-8). The data message from the feedback port (from the delay block) triggers summation process by picking up the data of the same time from the input record table.

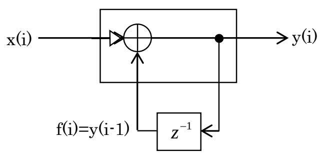

**Figure 4.** Forward integration loop.

In Fig. 5 which shows the backward version of Fig. 4, we need to calculate two outputs from a single input at the summing point satisfying y(i)=x(i)+f(i). The computing intensive solution is to simulate all the pairs, x(i) and f(i), matching the equation [4]. A practical solution is to perform simulation for finite number of divided range pairs, e.g. (x,f) for (0.0 to 2.5, 2.5 to 5.0) and (2.5 to 5.0, 0.0 to 2.5) to match the output of 4.5 [4]. The both solutions need a case branching mechanism.

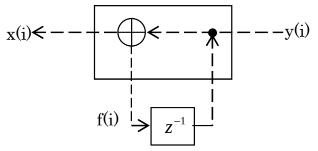

**Figure 5.** Backward integration loop.

However, if we apply another solution to use forward simulation module in the backward simulation as in Fig. 6, we can avoid the expansion of processing time caused by case branching. In Fig. 6 , we can formulate the process as x(i)=y(i)–f(i) and f(i)=y(i+1). The same input record table used in Fig. 4 can be utilized for making y(i) available to the inverse delayed feedback f(i)=y(i+1). The starting value y(i+1) remains in the table as the corresponding time feedback do not appear on the feedback input.

In Fig.6, the dot node sends received backward data in time reverse order to the both link to the sum node and to the delay node. And the delay y(i+1) will be treated as the feedback f(i). We have to describe the detailed processing of the sum node. In the backward simulation, data flows are expressed as a range (the minimum and the maximum). Then backward calculation must treat range information. If the range from the right is (a, b) and the range from the bottom is (c, d), the output through the left port should be calculated as (a-d, b-c) to cover the broadest value range. And x(i) must be larger than or equal to zero as it expresses calorie value. Then, if a-d or b-c is less than zero, they must be substituted by zero.

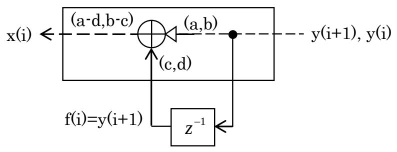

**Figure 6.** Hybrid backward integration loop.

Fig. 7 shows the resulting practical hybrid backward simulator, and Fig. 8 shows implemented objects and connections. Each object in Fig.8 performs backward processing, sum, dly and br for processing described with Fig.6, co for dividing by the coefficient. We coded it in Scala language which uses Java virtual machine. The simulator we made has forward and backward simulation functions with GUI interface.

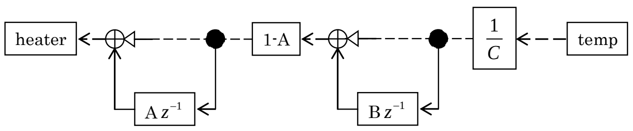

**Figure 7.** Hybrid backward simulation model.

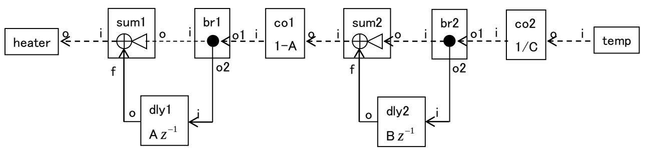

**Figure 8.** Hybrid backward simulation objects and ports.

## 4. MODEL SIMULATION

Fig. 9 is a result of forward simulation where the heater was turned on for the duration from 100s to 160s. The parameters A(=0.93), B(=0.994) and C(=4500) were estimated by the real measurement shown in the next section.

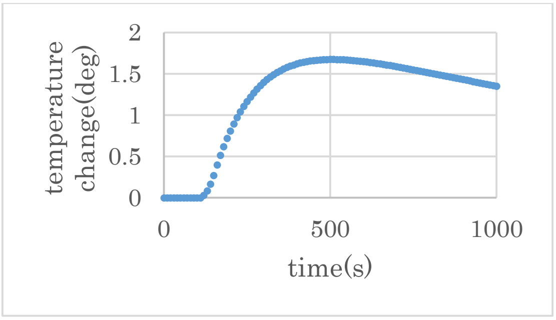

**Figure 9.** A model temperature change by heating(100-160s).

Figs. 10-12 are the results of backward simulation when the temperature change in Fig. 9 was fed by "temp" block in Fig. 8. The width of resultant ranges are shown by the difference between min and max values in those figures. The ndiv parameter was set 100, 1,000 and 10,000 for Fig. 10, 11 and 12, respectively. Although the smaller ndiv makes erroneous heater inference, larger ndiv reconstructs near original situation (Fig.12). As we used Float resolution for the temperature change data, we can successfully increase ndiv up to desired resolution.

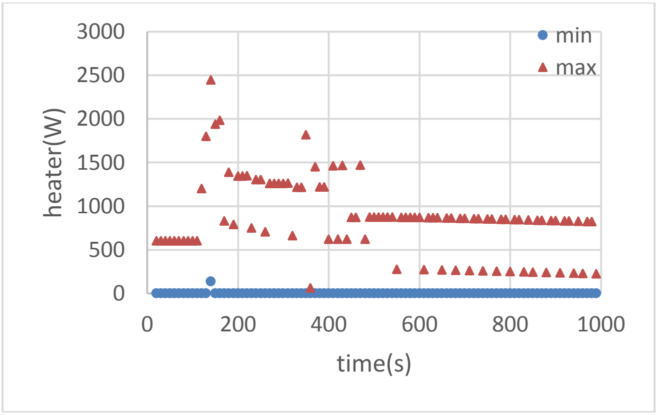

**Figure 10.** Backward simulation result for ndiv=100.

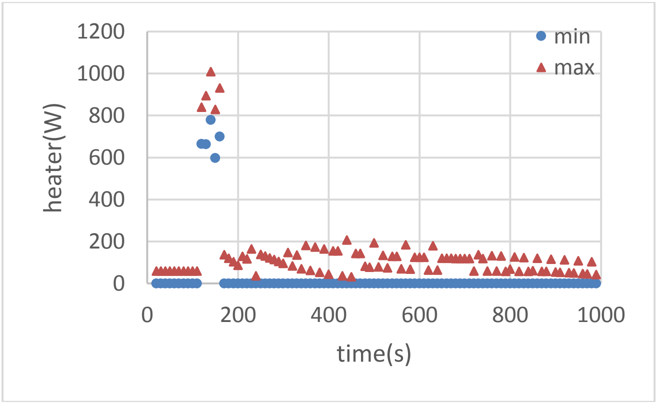

**Figure 11.** Backward simulation result for ndiv=1,000.

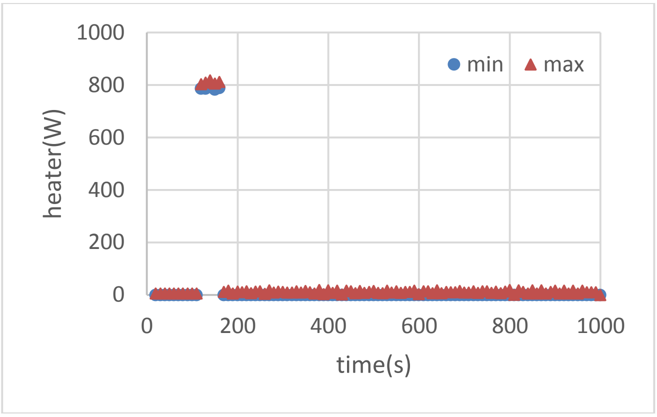

**Figure 12.** Backward simulation result for ndiv=10,000.

## 5. REAL DATA AND BACKWARD SIMULATION

We applied the backward simulator to real data in Fig.13, which is the room temperature response to 600W infrared heater turned on for 60 seconds in a tiny room of 3.6 m$^3$. The sensor is Sensirion's SHT71, which has 0.01 degree resolution, placed at 1m high in the room.

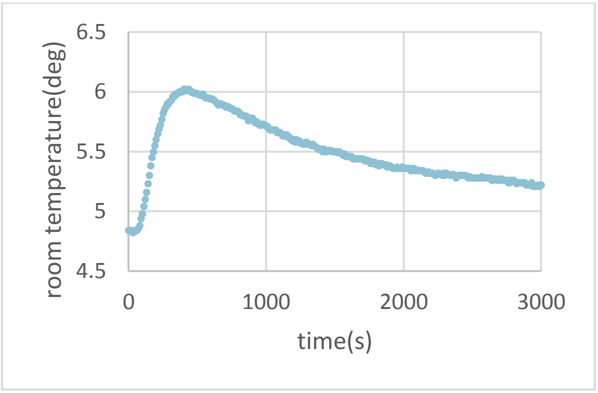

**Figure 13.** A measured temperature change caused by a real heater (ON for 0-180).

Figs. 14-15 are the results of backward simulation for the temperature change of Fig. 13. The ndiv parameter was set 100 and 150 for Fig. 14 and 15, respectively. The real heater operation (600W for starting 180 s) is residing inside the result range in Fig.14-15. Larger ndiv such as used in Fig. 11-12 makes more erroneously scattered heater estimation and the min values go up beyond 600W. The reason may be due to noise and model mismatch in the real measurement. In such cases, ndiv should not be set large.

Fig.16 is a naïve backward simulation which we changed the calculation described in relation to Fig.4 in order to suppress the widening of the result range in Fig.14-15. In that calculation we replaced backward result (a-d, b-c) with (a-c, b-d), more moderate estimation. In Fig.16, heating is more certain at the first 180s than Fig.14-15.

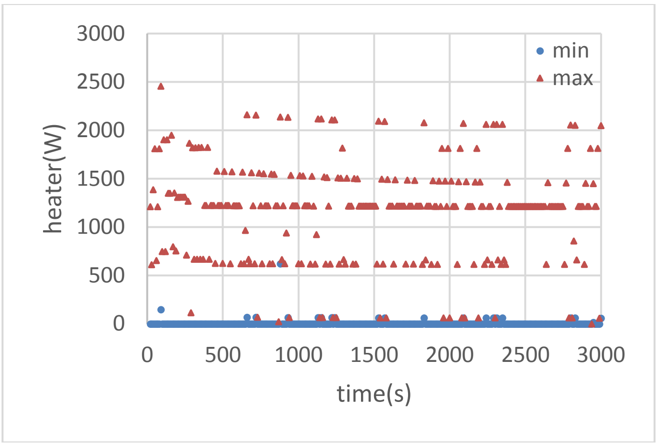

**Figure 14.** Backward simulation result for ndiv=100.

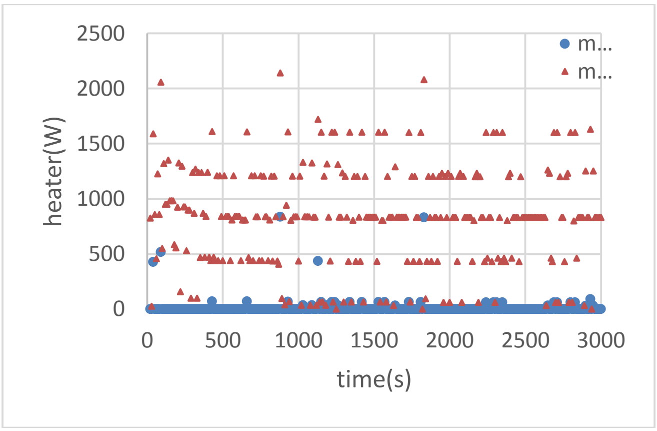

**Figure 15.** Backward simulation result for ndiv=150.

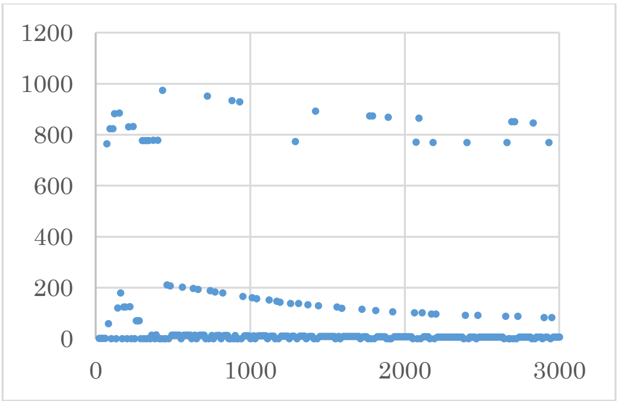

**Figure 16.** Naïve backward simulation result for ndiv=100.

## 6. DISCUSSION

The backward simulation for the model data without noise was successfully performed. In the case of real data, there is also a possibility of model discrepancy. It is important to keep enough tolerance to model mismatches and noise existence for practical applications.

As backward simulator does not need case branching and asymptotic processing, processing time is proportional to the number of simulation blocks and number of time samples. We used Scala's Actor (multi-thread) functionality suited for distributed processing, which may cause data arrivals out of order. Synchronous mechanism which assures the merit of distributed processing will be needed for the simulator.

## 4. CONCLUSIONS

We showed that backward simulation can be efficiently performed by using hybrid method (forward simulation modules in backward simulator). Causal state changes can be practically determined for simulated data and for real measured data. Internal state of any system can be determined by backward simulation if we define the system's model properly for backward simulation.

We have to study the relation among precision of simulation output, degree of model fitness and SNR of data for backward processing.

## ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Number 25540006.

## REFERENCES

[1] Chueng-Chiu Huang and His-Huang Wang, Backward Simulation with Multiple Objectives Control, Proc. IMECS (International MultiConference Engineers of Engineers and Computer Scientist), Hong Kong, 2009.

[2] Backward reasoning, Backward chaining, en.wikipedia.org.

[3] Y. Hiranaka and T. Taketa, DESIGNING, BACKWARD RANGE SIMULATOR FOR SYSTEM DIAGNOSES, Proc. XX IMEKO World Congress, 2012.

[4] Y. Hiranaka., T. Taketa and S. Miura, Case Branching Backward Simulator for Integer Factorization, Proc. 8th EUROSIM Congress on Modeling and Simulation, pp.259-264, 2013.

[5] Y. Hiranaka., H. Sakaki, K. Ito, T. Taketa and S. Miura, Numerical Backward Simulation Model with Case Branching Capability, Proc. 4th International Conference on Simulation and Modeling Methodologies Technologies and Applications (SIMULTECH 2014), pp.225-230, 2014.

[6] Y. Hiranaka and T. Taketa, Object-Oriented Framework for Cross-Layer Communication, Proc. ITSim2008 (International Symposium on Information Technology), pp.1640-1645, 2008.

[7] Y. Hiranaka, Y.Sato, T.Taketa and S.Miura, Smart Power Outlets with Cross-layer Communication, Proc. ICACT2011, pp.1388-1393, 2011.