ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 19-04-45

j.inst.Korean.electr.electron.eng.Vol.23,No.4,1430~1439,December 2019
http://dx.doi.org/10.7471/ikeee.2019.23.4.1430

FMI기반 co-simulation에서 step size control을 위한 Markov chain을 사용한 예측 방법

A Prediction Method using Markov chain for Step Size Control in FMI based Co-simulation

홍 석 준*, 임 덕 선*, 김 원 태**, 조 인 휘*★

Seokjoon Hong*, Ducsun Lim*, Wontae Kim**, Inwhee Joe*★

Abstract

In Functional Mockup Interface(FMI)-based co-simulation, a bisectional algorithm can be used to find the zero-crossing point as a way to improve the accuracy of the simulation results. In this paper, the proposed master algorithm(MA) analyzes the repeated interval graph and predicts the next interval by applying the Markov Chain to the step size. In the simulation, we propose an algorithm to minimize the rollback by storing the step size that changes according to the graph type as an array and applying it to the next prediction interval when the rollback occurs in the simulation. Simulation results show that the proposed algorithm reduces the simulation time by more than 20% compared to the existing algorithm.

요 약

FMI를 기반으로 하는 co-simulation의 마스터 알고리즘(MA)에서 시뮬레이션 결과의 정확도를 높이는 방법으로 zero crossing 포인트를 찾기 위한 Bisectional algorithm을 사용할 수 있다. 그러나 이 알고리즘은 많은 Rollback을 야기한다. 따라서 본 논문에서는 제안하는 MA는 Bisection algorithm을 통해 zero crossing 포인트를 검출하면서도 반복되는 구간 그래프를 분석하여 그 값을 Markov chain을 적용하여 다음 구간을 예측하여 이를 step size에 적용한다. 시뮬레이션에서 실제 Rollback이 발생했을 때 그래프 형태별로 변화되는 step size를 배열로 저장하고, 이룰 다음 예측 구간에 적용함으로서 Rollback을 최소화하는 알고리즘을 제안한다. 시뮬레이션 결과를 통해 제안하는 알고리즘이 기존 알고리즘에 비해 최대 20% 이상의 시뮬레이션 시간이 감소되는 것을 확인하였다.

Key words : Functional Mock-up Interface, co-simulation, zero crossing, rolllback, Markov chain

* Dept. of Computer Software, Hanyang University
** Dept. of Computer Science and Engineering, Koreatech University
★ Corresponding author
E-mail : iwjoe@hanyang.ac.kr, Tel : +82-2-2220-1088
※ Acknowledgment
This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2019R1I1A1A01058964)
Manuscript received Dec. 10, 2019; revised Dec. 27, 2019; accepted Dec. 30, 2019.

Ⅰ. 서론

Cyber Physical System(CPS)는 계산, 제어 기능 및 통신 기능을 가지는 복잡성을 가진 지능형 엔지니어링 시스템이다[1]. CPS의 최대 목적은 제어가 가능하고, 신뢰성을 가지며, 확장 용이 및 상호 운용을 가능하게 하는 효율적인 시스템을 구축하는 것이다. CPS는 우리의 삶을 유용하게 해주는 분야의 기술적 측면을 향상 시킬 수 있으며, 지능형 운송 시스템, 스마트 그리드, 의료 기기, 스마트 제어 시스템, 무기 시스템, 스마트 제조 시스템을 포함하여 다양한 분야에서 적용되고 있다[2]-[5].

Co-simulation은 여러 sub-system을 분산 방식으로 모델링하고 시뮬레이션하는 기술을 말한다. Co-simulation은 자동차[6], 전력 시스템[7], 항공[8] 등 여러 분야에서 연구되고 있는 주제이다. 각 sub-system은 시뮬레이터로서 값을 입력을 받고 이를 출력할 수 있는 블랙 박스라는 넓은 의미로 정의된다.

Functional Mock-up Interface(FMI) 표준[9]은 시뮬레이션 모델에서 인터페이스의 방법을 표준으로 정의하여 접근 가능성을 높이려고 한다. 이는 세부적으로 FMI는 동적 모델(dynamic model)의 Co-simulation과 Model-exchange를 모두 지원하는 독립적인 표준이며, FMI는 시뮬레이터가 함께 동작할 수 있도록 하기 위해 표준 API(application programming interface)를 정의하며, 이를 구현하는 모델은 Functional Mock-up Unit(FMU)로 정의 된다.

우리는 여러 CPS 시스템의 시뮬레이션 결과를 관찰하면서 시뮬레이션 그래프가 주기적으로 같은 패턴이 있는 경우가 종종 있음을 확인할 수 있었다. 이것은 많은 경우, CPS 시뮬레이션을 하는데 있어서 입력 데이터 값으로 특정 주기를 갖는 데이터를 사용하기 때문이다.

특정 주기를 갖는 입력 값으로 인해 다른 시스템의 요소에 대한 시뮬레이션 그래프가 시간에 따라 양수와 음수의 시뮬레이션 값을 갖는다면 zero crossing이 빈번히 나타나게 될 것이다. 언제 정확히 0의 값을 갖게 되는지 혹은 보다 정확한 zero crossing 포인트를 찾기 원한다면, Bisection algorithm과 같은 알고리즘[10]을 사용하여 시뮬레이션할 수 있다. 이 알고리즘은 어떤 모델의 시뮬레이션 진행 중에 zero crossing이 발생되면 zero crossing으로 인한 오차를 확인해서 만약 오차가 수용할 수 있을 정도의 크기가 아니라면 그림과 같이 이전 스텝으로 돌아간 후 simulation step size를 절반으로 줄이고 시뮬레이션하는 것을 계속 반복하여 가장 0에 가까운 지점을 찾는 것이다.

따라서, Bisection algorithm과 같은 zero crossing detection 알고리즘을 사용하여 시뮬레이션의 정확도를 높일 수 있는 반면에 동일한 주기를 갖고 반복되는 패턴의 시뮬레이션에서 zero crossing일 발생할 경우, bisection algorithm을 사용한다면, 시뮬레이션은 비효율적이고 또한, 시뮬레이션 시간도 늘어나게 될 것이다. 하지만, 만약 주기적으로 zero crossing을 반복하는 경우를 정확히 예측해서 미리 step size를 조정해서 시뮬레이션을 할 수 있다면, 여러 번 Rollback을 함으로서 낭비되는 시간을 줄일 수 있을 것이다.

본 논문에서는 Markov chain 기반으로 다음 Rollback 구간을 예측하고 실제 Rollback이 일어났을 때 변화되는 step size를 분석하고 이를 저장하여 다음 예측 구간에서 이를 적용함으로써 roll back을 최소화하는 알고리즘을 제안한다. 또한, 이를 통해 제안한 알고리즘을 기존 알고리즘과 비교하여 성능 및 효율성을 평가하였다.

이 논문의 나머지는 부분은 다음과 같이 구성되어 있다. 2장에서는 FMI와 FMI를 위한 master algorithm (MA)와 Rollback 기능에 관련된 기존 연구들에 대해서 설명하며, 3장에서는 제안하는 알고리즘에 대해서 설명하였다. 4장에서는 시뮬레이션 모델과 시뮬레이션 결과를 통해서 제안하는 알고리즘의 성능을 평가하였다. 5장에서는 본 논문의 결론을 맺는다.

Ⅱ. 관련 연구

1. The Function Mock-up Interface

FMI 표준은 MODELISAR project 연구를 통해 개발된 독립적인 interface tool 표준으로 Modelica Association가 관리하고 있으며, C 코드와 xml 파일을 사용해 서로 다른 시뮬레이션 환경에서 설계된 동적 모델 간의 표준화된 데이터 교환을 지원한다[9]. FMI의 주요 목적은 다른 모델링 도구로 설계된 모델에서 component 또는 하위 시스템(sub system)을 교환하고 상호 운용할 수 있도록 개선

하는 것이다. FMI 표준은 두 가지 주요 부분으로 구분되며, 이는 Model Exchange(ME)와 Co-simulation (CS)이다. 이 둘의 single FMU는 다음과 설명될 수 있다.

- ME에서의 FMI : 모델은 솔버(solver)없이 export되며, 이는 여러 개의 다른 simulation tool에서 component를 교환하기 위한 common format을 제공한다.
- CS에서의 FMI : 모델은 솔버(solver)와 함께 export하며, 전체 시스템은 master와 slave로 구성된다. 마스터는 오직 제어 만을 담당하며, 모든 slave는 동기화뿐만 아니라 데이터 교환을 통해 각자 독립적으로 해결한다.

본 논문에서는 앞에서 언급한 두 가지의 형태 중에서, 모델과 시뮬레이터를 함께 export하는 FMI co-simulation type의 FMU를 사용하는 co-simulation 알고리즘에 대해서 설명한다.

Co-simulation에서 Solver는 모델에서 제시된 문제를 해결하기 위해 이벤트 처리 및 통합 알고리즘이 포함된 소프트웨어 component이다. FMI 표준에는 구성요소(component)인 FMU가 준수해야 하는 Application Programming Interface(API)가 도입되어 있다. 우리는 시뮬레이션을 동작 시킬 때 master와 slave 간의 통신을 허용하는 co-simulation API의 필수적인 기능 중 일부를 소개한다.

- fmiDoStep : 이 기능은 Co-simulation에서 advancing time를 위해 FMI가 제공하는 방법이다. fmiOK 호출이 반환되면 step이 성공적으로 진행된 것이며, 진행할 수 없으면 fmiError를 반환한다. 만약 slave가 FMIDiscard를 반환했다면 communication step의 일부만 계산한 것이며, 이 때 알고리즘은 더 작은 communication step size로 다시 시도해야 한다.
- fmiSetXXX and fmiGetXXX : MA는 FMU에서 제공하는 fmiSetXXX를 호출하여 입력데이터를 FMU에게 제공하며, FMU로부터 출력 값을 검색하기 위해 fmiGetXXX를 호출하여 출력에서 ID와 저장할 위치에 대한 포인터를 인수로 제공한다('XXX'는 입력 및 출력의 데이터 유형으로 정의된다.).
- fmiSetFMUState and fmiGetFMUState : 이 기능들을 통해 master는 slave의 전체 상태를 저장하고 복원할 수 있다. fmi2SetFMUstate는 FMU의 내부 상태를 복원한다. 이는 FMU 상태에 대해서 이전에 복사된 입력 포인터가 필요하며, FMU의 현재 상태를 복사된 것으로 대체한다. fmi2GetFMUstate는 FMU의 내부 상태를 저장한다. FMU 상태에 관련된 포인터를 입력으로 받아서 현재 FMU 상태를 복사한다.

FMI에서 다양한 API가 존재하여 FMU의 상태를 읽고 쓸 수 있지만 보강해야할 사항들이 몇 가지 사항들이 있다. 첫째로 상태를 저장하고 복원하면 오버헤드가 발생할 수 있기에 이를 줄일 수 있는 방법을 고려하여 설계를 해야 한다. 둘째, Rollback이 발생하지 않아야 할 구간에서 발생하는 것을 예방하는 설계를 하기에는 지원하는 API가 충분하지 않다. 이에 본 논문에서는 FMI 표준에서 지원하는 API 준수하여 Rollback 기능을 구현하고 FMU에서 지원하도록 하였다. 알고리즘 설계에 있어 이러한 절차들을 어떤 순서로 호출해야 하는 것이 핵심적인 사항이다.

2. The Master Algorithm for FMI and Rollback functionality

FMI를 기반으로 하는 co-simulation에서 master algorithm(MA)은 FMI를 통해 FMU를 시뮬레이션하는 알고리즘이다. 몇몇의 기존 연구들에서 여러 FMU를 co-simulation하기 위한 MA을 제안하였다. [11]에서는 여러 개의 FMU에 대해서 먼저 기본 step size로 doStep을 통해 시뮬레이션을 수행해보고 만약 실패가 난 FMU가 있는 경우. 먼저 에러가 나기 이전 시점으로 복귀한 후(Rollback) 모든 FMU들 각각이 진행할 수 있는 최대 step size들을 확인해서 이중에 최소값으로 진행하도록 하였다. 반면에 다른 연구에서는 여러 개의 FMU를 이용해서 co-simulation할 때 먼저 기본 step size(h)로 시뮬레이션을 진행하고(T_2=T_0+h) 만약, 어느 하나의 FMU라도 invalid state이거나 error flag가 발생하면, 전체 시뮬레이션을 이전 시뮬레이션 시간(T_0)으로 롤백 후 전체 FMU들의 step size를 이전에 비해 작은 step size(h')로 다시 조정한 후 시뮬레이션하는 방식 반복하여 유효한 상태에 이르기까지 시뮬레이션하는 step revision 알고리즘을 제안하였다[12].

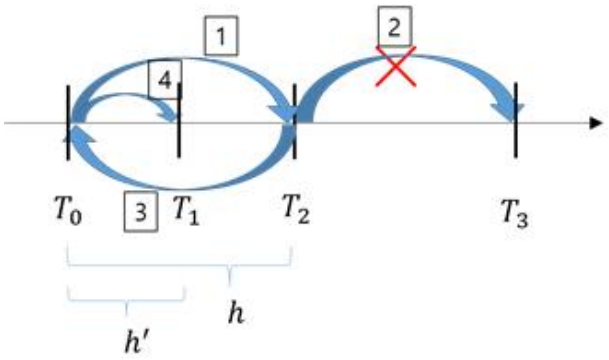

Fig. 1. Step revision algorithm with Rollback.
그림 1. 롤백과 스텝 개정 알고리즘

III. Markov 예측 기반의 효율적인 Zero crossing 검출 알고리즘

본 논문에서는 주기적으로 zero crossing이 반복되는 시뮬레이션 모델에 대해서 처음 zero crossing이 발생한 경우, Bisection 알고리즘에 의해서 Rollback이 되면서 변경된 step size값들을 시뮬레이션 값과 함께 저장해놓음으로서 다음에 같은 형태의 zero crossing이 발생될 것이 예상되는 경우, simulation step size를 이전 저장된 값들을 사용하여 시뮬레이션함으로서 Rollback을 최소화하여 효율적으로 시뮬레이션하는 알고리즘을 제안한다. 이때 각 시뮬레이션 그래프가 어떤 형태로 반복되는지에 대해서 예측하기 위해서 시뮬레이션이 수행될 때의 각 시점에서의 상태 값을 구하기 위해서 아래 그림과 같이 시뮬레이션 값과 기울기를 토대로 각 상태를 만든다.

또한, 시뮬레이션 값의 범위 상태(range state, i)와 기울기 상태(direction state, j)의 각각 2개의 상태의 조합을 통해 총 4개(1~4)의 최종 상태 값(S)을 아래와 같은 공식(1)에 의해 계산한다.

$$S = 2 \times (i-1) + j \tag{1}$$

시뮬레이션이 진행되는 동안 알고리즘은 시뮬레이션 시간에 따른 위 상태 값을 계속 업데이트하게 된다. 그리고 이 상태 값이 바뀔 때만 현재 상태 값으로 저장함으로서 상태가 어떻게 바뀌었는지 알 수 있게 된다.

제안하는 예측 알고리즘은 위와 같이 상태를 구분한 후, 시뮬레이션의 상태가 이전 상태와 다른 경우에 계속 저장하고 이를 통해서 다음 상태를 예

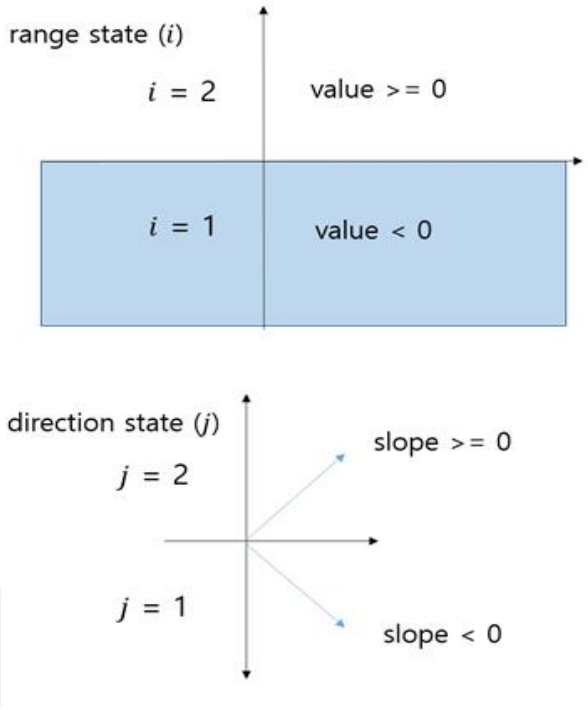

Fig. 2. Range and direction state of simulation value.
그림 2. 시뮬레이션 값의 범위 상태와 기울기 상태

측하기 위해 Markov chain을 이용한다.

하나의 Markov chain은 랜덤 변수들 X_1, X_2 , ..., X_n와 Markov 특성의 sequence이다. 만약 조건부 확률이 다음과 같이 정의될 경우, 다음 상태로 이동할 확률은 오직 현재 상태의 값에 의존한다는 것이다. 즉, Markov chain은 transition probabilities p_{ij} 의해 서술된다. 언제든 현재 state가 i가 될 경우, 다음 state가 j가 될 확률을 p_{ij} 로 표현할 때, Markov property에 의해 다음의 식이 만족된다. 이때 S는 유한한 state space이며 S = {1, … , m}이고 m은 양의 정수이다[13].

$$\begin{aligned} &P(X_{n+1}=j|X_n=i, X_{n-1}=i_{n-1},..., X_0=i_0) \\ &= P(X_{n+1}=j|X_n=i) = p_{ij}, \quad i,j \in S \end{aligned} \tag{2}$$

이전 연구들에서는 Delay Tolerant Network(DTN) 환경에서 Meeting Time span 혹은 노드의 방향과 속도를 상태로 구분해서 다음 상태를 예측하는데 Markov를 사용했다[14],[15]. 본 연구에서는 시뮬레이션 그래프의 추이를 예측하기 위해서 시뮬레이션 그래프의 현재 값과 기울기를 통해서 상태를 총 4가지 상태(S={1, 2, 3,4}) 값으로 구분하여 일정 주기 동안의 상태 값들을 저장 하고 변환 확률 행렬(transition probability matrix)을 만들어서 다음

상태를 예측하는 방법을 통해 zero crossing이 일어날 확률을 구하는 방법을 제안하였다.

시뮬레이션 그래프의 상태에 대한 The transition probability matrix P는 이전 시뮬레이션 그래프의 상태 sequence들을 이용해서 만들어진다. 시뮬레이션 그래프에서의 P는 다음과 같이 정의되어진다.

$$P = \begin{bmatrix} p_{11} p_{12} p_{13} p_{14} \\ p_{21} p_{22} p_{23} p_{24} \\ p_{31} p_{32} p_{33} p_{34} \\ p_{41} p_{42} p_{43} p_{44} \end{bmatrix} \quad (3)$$

여기서 p_{ij} 는 현재 상태가 i 일 때, 다음 상태가 j 인 경우의 확률이다. 시뮬레이션 그래프에서의 p_{ij} 는 다음과 같이 계산되어질 수 있다.

$$p_{ij} = \begin{cases} 0, & i = j \\ \frac{N_{ij}}{N_i}, & i \neq j \end{cases} \quad (4)$$

먼저 p_{ij} 값을 결정할 때, 현재 상태(i)와 다음 상태(j)가 같은 경우에는 0값을 갖도록 한다. 이것은 동일한 상태로 바뀌는 경우에 대해서는 고려하지 않는다는 것을 의미한다.

다음으로 N_i는 다음 상태를 고려하지 않고 상태 i 에서 다른 상태들로 변화된 횟수이고, N_{ij}는 상태 i 에서 상태 j로 변화된 횟수이다. 또한, P를 구하기 위해서 공식 (3)을 적용해서 각 p_{ij}를 계산하여 업데이트하는 기간은 시뮬레이션 시작 후 시뮬레이션 그래프의 최소 반복 주기인 T_p 까지로 한다. 즉, T_p 가 시뮬레이션 그래프의 주기라면, T_p 이후에는 동일한 상태 변화가 반복될 것이므로 다시 계산할 필요가 없다.

시뮬레이션이 진행되다가 zero crossing 포인트를 지나게 되면, Bisection 알고리즘에 의해 시뮬레이션 값이 Signal Threshold이하 값이 될 때까지 계속 step size를 반으로 줄이고 Rollback하여 다시 진행하는 것을 반복한다. 이때 제안하는 알고리즘은 step size를 줄이고 진행하는 과정에서 앞으로 진행된 step size들만을 그 시점의 시뮬레이션 값과 기울기 값과 함께 별도의 Array에 저장해 놓는다. 이후 다시 시뮬레이션이 진행되면 Markov 예측에 의해서 다음 상태 값이 zero crossing이 일어나는 경우, 현재 시뮬레이션 값과 Array에 저장된 이전 시뮬레이션 값들을 비교해보면서 완전히 같은 값은 아니더라도 미리 정한 오차 범위보다 작은 경우에는 Array에 저장된 step size를 적용해서 진행하도록 한다.

또한, 연속된 시뮬레이션 모델에서는 state가 3에서 1로 바뀌거나(down-crossing) 2에서 4로 바뀔 때 (up-crossing)의 2가지 경우의 zero-crossing일 발생되기 때문에 각각 Array1과 Array2에 저장하여 비교하도록 한다.

이후 시뮬레이션이 진행되면서 down/up crossing의 발생이 예측되고, 각각의 array에 값이 존재하는 경우, 현재 값과의 비교를 하면서 일정 오차 범위에 들어오면, array에 저장되어 있던 step size를 적용하면서 Rollback을 방지하도록 한다.

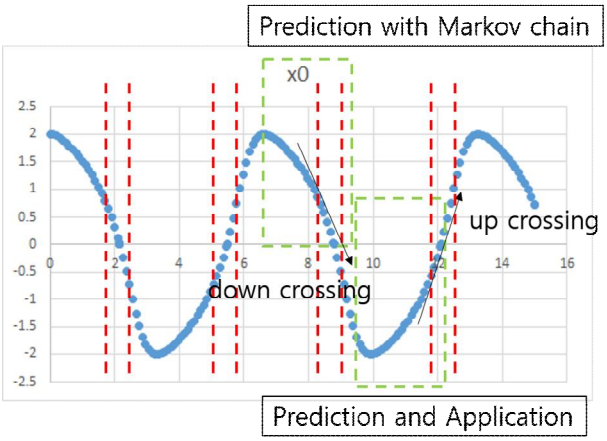

Fig. 3. Predict, and apply down/up crossing array.
그림 3. Down/up crossing array 예측과 적용

위 그림은 각각 어떻게 down/up zero crossing할 때의 값들이 array에 저장되고, 다시 Markov 예측과 값 비교를 통해서 step size가 적용되는지를 보여준다.

그림 4의 코드는 어떻게 zero crossing이 일어난 것을 감지해서 Rollback이 일어나고 이와 동시에 down crossing 혹은 up crossing의 경우 각각의 array에 값을 어떻게 저장하는지를 보여주는 코드이다. 먼저 시뮬레이션이 수행되는 동안에는 계속해서 getFMUState 함수를 호출하여 현재 fmuState 변수에 현재 상태값을 저장해놓는다. 이후 현재 FMU에서 시뮬레이션하고자 하는 상태 혹은 scalar 변수값을 getReal함수를 통해 읽어온다. 이 예제 코드에서는 시뮬레이션 상태 값을 계산하기 위해서 시뮬레이션 값 curr_r과 그 값에서의 기울기로 미

```
while (time < End)
{
fmu->getFMUState(c, &fmuState);
fmu->doStep(c, time, hh);
fmu->getReal(c, &vr, 1, &curr_r);
fmu->getReal(c, &vr2, 1, &curr_v);
curr_state = checkCurrState(curr_r, curr_v);
if(time < Tperiod)
        makeMarkovMatrix(curr_state);
if((prev_r * curr_r) < 0) //zero crossing
{
   if( fabs(curr_r) > SIGNAL_THRESHOLD)
   {
        duringRollback = 1;
        hh /= 2; //bisection algorithm
        fmu->setFMUState(c, fmuState);
        continue;
} else {
        duringRollback = 0;
        hh = h; //recover to original time step.
  if((prev_state == 3) && (curr_state == 1))
                AddToArray1(prev_r, prev_v, hh);
  else if((prev_state == 2) && (curr_state == 4))
                AddToArray2(prev_r, prev_v, hh);
}
} else {
 if( duringRollback == 1)
 {
     if((prev_state == 1) && (curr_state == 3))
        AddToArray1(prev_r, prev_v, hh);
     else if((prev_state == 4) && (curr_state == 2))
        AddToArray2(prev_r, prev_v, hh);
 }
}
/ * predict next zero crossing */

prev_r = curr_r;
prev_v = curr_v;

time += hh;
}
}
```

Fig. 4. Saving process when rollback occurs.
그림 4. 롤백이 일어나는 경우의 저장 과정

분값인 curr_v을 사용했다. 또한, 시뮬레이션 그래프의 상태 측정 및 Markov transition matrix를 만들기 위해 checkCurrState 함수와 makeMarkovMatrix함수를 호출한다. 이때 makeMarkovMatrix함수는 시뮬레이션 그래프가 반복되는 최소 주기 동안만 업데이트를 하면 된다.

다음으로 시뮬레이션의 이전 값(prev_r)과 현재 값(curr_r)을 곱한 결과의 부호를 통해서 zero crossing이 일어났는지를 판단할 수 있다. 즉, 결과가 음수인 경우 zero crossing이 발생한 것이다. 또한, zero crossing이 일어난 경우 현재 시뮬레이션의 절대값이 특정값(SIGNAL_THRESHOLD)값보다 작은지를 판단해서 만약 작지 않으면, setFMUState를 하고 다시 doStep을 함으로서(이 과정을 Rollback으로 부르기도 한다.) Bisection algorithm을 수행한다. 이때 처음 Rollback을 시작하는 경우에 during Rollback 값을 1로 함으로서 현재 rollback 중임을 나타낸다. 만약 시뮬레이션의 절대값이 SIGNAL_THRESHOLD값보다 작은지를 판단해서 작으면 이 경우는 zero crossing이 일어났지만, 시간에 대한 오차가 Theshold보다 작기 때문에 더 이상 Rollback하지 않고 다음 시뮬레이션 시간으로 진행된다. 이 때 진행되는 시뮬레이션의 step size는 원래 시뮬레이션의 기본값인 h로 변경되어 수행된다. 그리고 zero crossing을 통과한 점은 이전 상태(prev_state)와 현재 상태(curr_state)의 값을 통해서 down crossing인 경우 Array1에 위치 및 기울기 그리고 step size 정보를 포함한 array에 저장되고, up crossing인 경우 Array2에 저장한다.

다음으로 그림 5와 6은 그림 4의 코드는 Markov 예측을 통해서 시뮬레이션 값의 상태를 어떻게 예측하고 step size(hh)가 계산되어질 수 있는지를 보여준다. 또한, 그림 6은 현재 시뮬레이션 값을 이전에 저장해놓은 Array(down crossing인 경우, Array1, up

```
while (time < End)
{
//doStep
if((prev_r * curr_r) < 0) //zero crossing
{
//rollback
//save to Array
}
next_state = markov_prediciton();
if((curr_state == 3) && (next_state == 1) && (IsArray1()
== TRUE))
        hh = predictStepSize1(curr_r, curr_v);
else if((curr_state == 2) && (next_state == 4) &&
(IsArray2() == TRUE))
hh = predictStepSize2(curr_r, curr_v);

prev_r = curr_r;
prev_v = curr_v;

time += hh;
}
```

Fig. 5. Step size(hh) calculation process using Markov.
그림 5. Markov 예측을 통한 step size(hh) 계산 과정

```
k = 0;
while (k < ArraySize1)
{
if( (curr_r - Array1[k].r >= 0) && (curr_v - Array1[k].v
>= 0)){

gap_r = fabs((curr_r - Array1[k].r) /Array1[k].r);
gap_v = fabs((curr_v - Array1[k].v)/Array1[k].v);

            if(k == 0) {
            if((gap_r <= 0.1) && (gap_v <= 0.1))
                        predic_hh = Array1[k].hh;
                        break;
            }else{
            if( (Array1[k].hh < (prev_hh / 2)) && (curr_r +
prev_hh * v /2) > 0){
                        predic_hh = prev_hh / 2;
            } else {
                        predic_hh = Array1[k].hh;
}
break;
}
}
k++;
```

Fig. 6. predictStepSize1 function.
그림 6. predictStepSize1 함수

crossing인 경우 Array2)와 어떻게 비교해서 step size를 결정하는지를 보여준다. predictStepSize2함수는 Array1 대신 Array2로만 대체하면 나머지는 동일하다.

Ⅳ. 시뮬레이션 모델과 성능 평가

본문에서 제안한 알고리즘의 성능 평가를 위해서 VanDerPol FMU를 FMU SDK[16]로 시뮬레이션을 수행하여 시뮬레이션 결과를 확인해보았다. 그림 7은 FMU SDK에서 제공하는 VanDerPol FMU를 기본 MA으로 시뮬레이션한 결과이다.

vanDerPol 그래프의 2개의 continuous states (x_0, x_1)에 대한 시뮬레이션 그래프의 transition probability matrix P는 아래와 같이 구해질 수 있다. 따라서, 이 매트릭스에 의해서 만약 현재 상태가 1이라면 다음 상태는 2가 될 수 있음을 알 수 있다.

$$P=\begin{bmatrix}0&1&0&0\\0&0&0&1\\1&0&0&0\\0&0&1&0\end{bmatrix}$$

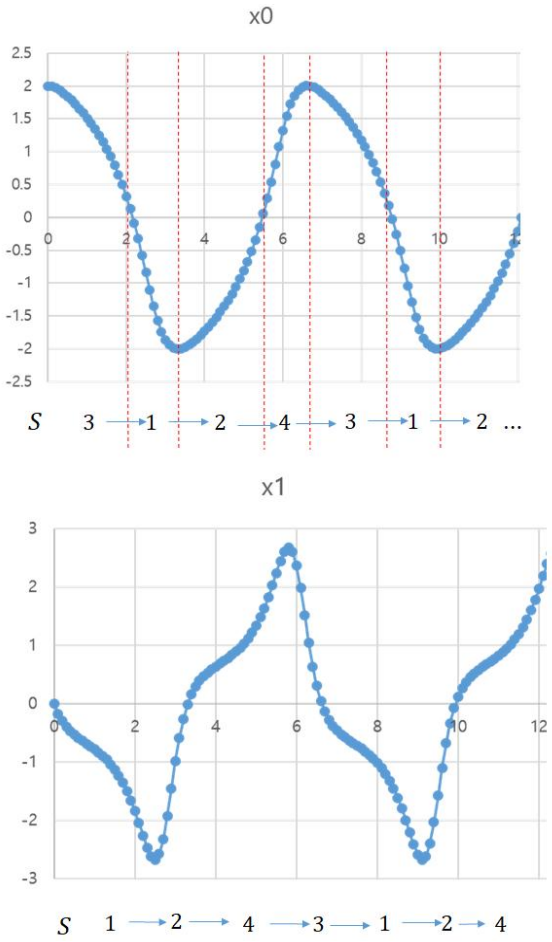

Fig. 7. VanDerPol(x_0,x_1) simulation graph
그림 7. VanDerPol(x_0,x_1) 시뮬레이션 그래프

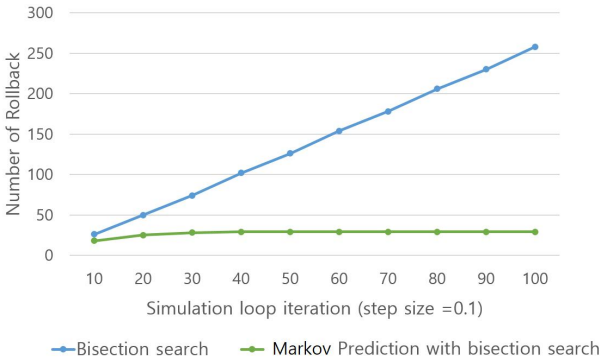

Fig. 8. Number of Rollback comparison.
그림 8. 롤백 횟수의 비교

그림 8의 결과는 vanDerPol을 FMU로 해서 시뮬레이션했을 때, 시뮬레이션 loop를 증가시키면서 MA이 Bisection search만을 사용한 경우와 Markov예측을 같이 사용한 경우의 결과를 나타낸다. 기존 알고리즘의 경우 동일한 패턴으로 zero crossing이

발생되어 Rollback이 계속 증가하는 것을 볼 수 있다. 반면 제안하는 알고리즘의 경우, 처음의 한 주기만 Rollback이 여러번 일어나게 되고, 이후에는 Markov 예측을 통해 동일한 Rollback은 예측해서 step size를 조정함으로서 추가적인 Rollback이 거의 일어나지 않는 것을 확인할 수 있다.

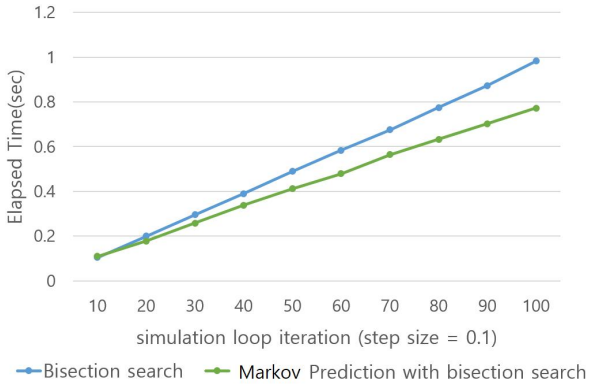

Fig. 9. Simulation time comparison.
그림 9. 시뮬레이션 시간의 비교

그림 9는 시뮬레이션 loop의 총 횟수를 증가시키면서 측정한 시뮬레이션 시간이다. 제안하는 알고리즘이 시뮬레이션 loop가 증가함에 따라 점점 더 빨리 시뮬레이션 되는 것을 볼 수 있는데 이것은 그림 8에서 알 수 있었던 것처럼 불필요한 Rollback을 계속해서 줄일 수 있었기 때문이다. 시뮬레이션 시간 비교를 통해 Markov 예측을 같이 사용한 경우가 Bisection 알고리즘만 사용한 경우에 비해 최대 20%이상 시뮬레이션 시간을 줄일 수 있음을 확인하였다.

Ⅴ. 결론

본 논문에서는 CPS 시뮬레이션을 하는데 있어서 주기적인 시뮬레이션 그래프 패턴을 가지는 경우가 있음을 착안하여 만약 주기적인 시뮬레이션 그래프에서 zero crossing이 자주 일어나는 경우에는 zero crossing 감지를 위한 롤백 과정이 반복해서 일어나기에 전체 시뮬레이션이 비효율적이게 된다.

따라서 본 논문에서는 이전 시뮬레이션 수행 시 시뮬레이션 그래프 상태 값들을 이용해서 Markov matrix를 만들고 Markov 예측을 통해서 다음 시뮬레이션 구간에서 동일한 패턴이 나오는 경우, 이전 롤백을 수행하면서 시뮬레이션에 적용되었던 step size를 적용함으로서 최소한의 Rollback이 될 수 있도록 하는 알고리즘을 제안하였다.

그리고 FMU에 적용할 수 있는 예측 알고리즘들을 사용했을 때의 결과 비교를 통해 제안하는 알고리즘이 Rollback 횟수를 최대한 줄임으로서 전체 시뮬레이션 속도가 증가될 수 있음을 확인하였다. 또한, 시뮬레이션 결과를 통해 제안하는 알고리즘이 기존 알고리즘에 비해 최대 20% 이상의 시뮬레이션 시간이 감소되는 것을 확인하였다.

References

[1] J. Eidson E.A. Lee S. Matic S. A. Seshia J. Zou “Distributed Real-Time Software for Cyber-Physical Systems,” *Proceedings of the IEEE (special issue on CPS)* pp.45-59, 2012.
DOI: 10.1109/JPROC.2011.2161237

[2] X. Guan B. Yang C. Chen W. Dai Y. Wang “A comprehensive overview of cyber-physical systems: from perspective of feedback system,” *IEEE/CAA Journal of Automatica Sinica* vol.3 no.1, pp.1-14, 2016. DOI: 10.1109/JAS.2016.7373757

[3] A Sharma, G Rathee, R Kumar, H Saini, V Vijaykumar, Y Nam, N Chilamkurti. “A Secure, Energy-and SLA-Efficient (SESE) E-Healthcare Framework for Quickest Data Transmission Using Cyber-Physical System,” *Sensors*, Vol.19, No.9, 2019. DOI: 10.3390/s19092119

[4] D. Roy L. Zhang W. Chang S. K. Mitter S. Chakraborty “Semantics-preserving cosynthesis of cyber-physical systems,” *Proceedings of the IEEE*, vol.106, no.1, pp.171-200, 2018.
DOI: 10.1109/JPROC.2017.2779456

[5] Z. Yu L. Zhou Z. Ma M. A. El-Meligy “Trustworthiness modeling and analysis of cyber-physical manufacturing systems,” *IEEE Access,* vol.5, pp.26076-26085, 2017.
DOI: 10.1109.ACCESS.2017.2777438

[6] W. Li X. Zhu J. Ju “Hierarchical braking torque control of in-wheel-motor-driven electric vehicles over CAN,” *IEEE Access,* vol.6, pp. 65189-65198, 2018.

DOI: 10.1109/ACCESS.2018.2877960
[7] W. Yu Y. Xue J. Lou et al. "An UHV grid security and stability defense system: considering the risk of power system communication," *IEEE Trans. Smart Grid,* vol.7, no.1, pp.491-500, 2016. DOI: 10.1109/TSC.2015.2392100
[8] M. B. Kamal G. J. Mendis J. Wei "Intelligent soft computing-based security control for energy management architecture of hybrid emergency power system for more-electric aircrafts," *IEEE J. Sel. Topics Signal Process,* vol.12, no.4, pp.806-816, 2018. DOI: 10.1109/JSTSP.2018.2848624
[9] Functional mock-up interface for model exchange and co-simulation, Version 2.0, Information Tech for European Advancement, Tech. Rep., Modelisar, 2012. http://www.fmi-Standard.org/downloads/
[10] T. Bui S. Chaudhuri T. Leighton M. Sipser "Graph Bisection Algorithms with Good Average Case Behavior," *Proc. 25th Int'l Symp. Foundations of Computer Science,* pp.181-192, 1984. DOI: 10.1007/BF02579448
[11] D. Broman C. Brooks L. Greenberg E. A. Lee M. Masin S. Tripakis M. Wetter "Determinate Composition of FMUs for Co-Simulation," *Proceedings of the International Conference on Embedded Software (EMSOFT 2013),* 2013. DOI: 10.1109/EMSOFT.2013.6658580
[12] F. Cremona M. Lohstroh D. Broman M. D. Natale E. A. Lee S. Tripakis "Step revision in hybrid co-simulation with FMI," *2016 ACM/IEEE International Conference on Formal Methods and Models for System Design MEMOCODE 2016,* pp.173-183, 2016. DOI: 10.1109/MEMCOD.2016.7797762
[13] DP Bertsekas, JN Tsitsiklis, *Introduction to probability*, Athena Scientific, 2002.
[14] I. Jeon K. Lee "A dynamic Markov chain prediction model for delay-tolerant networks," *International Journal of Distributed Sensor Networks,* vol.12, 2016. DOI: 10.1177/1550147716666662
[15] E. Wang, Y. Yang, B. Jia, "The DTN routing algorithm based on Markov meeting time span prediction model," *International Journal of Distributed Sensor Networks,* vol.9 2013. DOI: 10.1155/2013/736796
[16] "QTronic. FMU SDK (FMU Software DevelopmentKit)," http://www.qtronic.de/de/fmusdk.html

BIOGRAPHY

Seok-Joon Hong (Member)

2006 : BS degree in Electronics and Computer Engineering, Hanyang University.
2008 : MS degree in Electronics and Computer Engineering, Hanyang University.
2017 : PhD degree in Electronics and Computer Engineering Engineering, Hanyang University.
2017~: Post. Doc in Hanyang University.

Duc-Sun Lim (Member)

2008 : BS degree in Computer Engineering, Hanyang Cyber University.
2014~ : MS and Ph.D degree in Computer and software, Hanyang University.

Won-Tae Kim (Member)

1994.2 : BS degree in Electrical Engineering, Hanyang University.
1996.2 : MS degree in Electronics Engineering, Hanyang University.
2000.8 : PhD degree in Electronics Engineering, Hanyang University.
2001.1~2005.2 : CTO, Rostic Technologies Inc.
2005.3~2015.8 : CPS Team Manager, ETRI
2015.9~ : Assistant professor, Koreatech University

In-Whee Joe (Member)

1983.2 : BS degree in Electronics Engineering, Hanyang University.
1995.2 : MS degree in Information and Communication, Arizona University.
1998.2 : PhD degree in Electrical and Computer Engineering, Georgia Institute of Technology University.
1985~1992 : Research Engineer, DACOM Research Institute
1998~2000 : Researcher, Oak Ridge National Laboratory
2000~2002 Research Scientist, Bellcore Lab
2002~ : Professor, Hanyang University