특 집

제어시스템에서의 시뮬레이션 기법과 로봇공학에의 응용

李 柱 張, 金 聖 祐, 李 鍾 權

韓國科學技術院 電氣 및 電子工學科

I. 서 론

일반적으로 시뮬레이션이라 함은 현실세계의 여러 측면에 대한 연구가 용이하도록 다루기 쉬운 숫자나 기호로써 그 대상을 나타내는 과정을 말한다. 따라서, 공학뿐만 아니라 사회, 경제, 의학, 운송 등 여러 학문 및 연구분야에서 유용하게 이용되고 있다. 특히, 시뮬레이션은 실 시스템(real system)의 복잡성에서 기인하는 수학적 분석의 어려움을 극복하고, 여러 가지 제약에 의해 실제의 실험이 불가능하거나 또는 실험의 위험부담이 크고 소요비용이 많이 드는 경우에 아주 유용하다.

시뮬레이션을 위해서는 물리적인 실제시스템과 동작환경을 효과적으로 나타내는 모델이 필요하다. 이렇게 얻어지는 모델이 실제 시스템의 동작을 잘 모사하도록 만든다면, 우리는 (1) 특정한 동작환경 하에서 시스템의 성능을 예측하고, (2) 시스템을 테스트하고 평가하며, (3) 시스템에서 보다 깊은 연구가 필요한 부분을 알아낼 수 있는 방법을 갖게 된다. 시뮬레이션만의 장점이라고 할 수 있는 것은 고장이나 성능저하 등의 하드웨어적인 문제와 주된 관심사인 시스템설계 문제를 분리시킬 수 있는 능력이다. 특히 시스템의 동작(behavior)이 완전히 파악되지 않은 경우에는 시뮬레이션의 이러한 장점은 더욱 부각된다.[1]

본고에서는 특별히 제어시스템에 대한 시뮬레이션의 응용에 대해서 살펴본다. 넓은 의미에서 제어시스템은 여러 가지 형태를 가질 수 있겠으나, 미분방정식(differential equation)이나 차동방정식(difference equation) 등의 운동방정식(dynamic equation)으로 표현되어지는 제어시스템에 대한 시뮬레이션을 다루어 보고자 한다.

또한, 최근들어 로봇공학 및 컴퓨터 그래픽스 분야에서 실제 환경을 모사(simulate)하기 위한 많은 노력들이 경주되어 왔다. 이와 같은 시뮬레이션(simulation ; 모사, 모의실험)은 컴퓨터 디자인(CAD)을 이용하여 실제 요구되는 대상 및 환경을 모델링, 설계, 검증하기 위한 것으로서, 실제 대상과 환경에 대한 물리적인 특성의 이해 및 통찰력을

제공하여 준다.

일반적으로 CAD는 엔지니어링 도면, 건축 도면 등의 설계 등에서 광범위하게 사용되어 왔고, 최근에는 로봇공학에도 응용되어 로봇의 설계 및 분석 등에 이용되어 왔으나, 이들의 대부분은 정적인 움직임만을 대상으로 하고 있기 때문에, 실제 로봇이 움직이면서 작업하는 경우의 해석에는 다소 부적합하였다. 따라서 동적인 움직임을 해석하고 설계하는 수단으로서 시뮬레이션이 로봇공학의 산업계, 교육계, 연구계 각분야에 걸쳐 활발히 연구되고 있다.

II. 제어시스템에서의 시뮬레이션 기법

1. 제어시스템과 시뮬레이션

서론에서도 언급했듯이 본고에서는 운동방정식으로 표현되는 제어시스템에 대해서 다루고자 한다. 즉, 시스템의 모델이 미분방정식이나 차동방정식으로 표현되는 제어시스템을 말한다. 이러한 운동방정식은 곧 바로 제어시스템에 대한 상태방정식(state equation)으로 변형시킬 수 있게 되므로, 시뮬레이션을 하기 위한 아주 용이한 형태가 된다.

먼저 제어시스템에 대하여 간단히 살펴보면, 일반적으로 제어시스템은 신호(signal)의 형태에 따라 연속시간 제어시스템(Continuous-time Control System)과 이산시간 제어시스템(Discrete-time Control System)으로 나눌 수 있게 된다. 연속시간 제어시스템에 대한 모델은 보통 다음과 같은 1차 벡터 미분방정식의 형태를 취한다.

$$\dot{x}(t) = f[t, x(t), u(t)], \forall t \geq 0,$$

여기서, x(t)는 시간 t에서의 상태변수(state variable)를 나타내며, u(t)는 시스템의 제어입력을 나타낸다. 한편, 이산시간 제어시스템에 대한 모델은

$$x_{k+1} = f_k(k, x_k, u_k), k = 0, 1, 2, 3, \cdots,$$

의 1차 벡터 차동방정식의 형태이다. 이와 같은 1차 벡터 미분(혹은 차동)방정식은 곧 연속시간(혹은 이산시간) 제어시스템에 대한 상태방정식(state equation)으로 볼 수 있다.

제어시스템에서 제어의 목적은 간단히 말해서, 적절한 제어입력 u를 시스템에 가함으로써 시스템의 출력 또는 상태변수 x를 원하는 대로 만들어 주는 것이라 할 수 있다. 이러한 제어목적 아래서 피드백 시스템(Feedback System)과 같은 형태로 전체 시스템을 구성하고 적절한 제어기를 설계하는 것이 제어공학자들이 하는 일이라고 말할 수 있을 것이다. 그런데, 다른 시스템과 마찬가지로 제어시스템에 있어서도 이렇게 설계한 제어기가 과연 원하는 성능을 나타낼 것인지, 특히 시스템이 거대하고 복잡한 경우에는 실제로 제어시스템을 구성하지 않고 제어기의 성능을 예측하는 것이 필요하게 된다. 이러한 필요에 따라 시뮬레이션은 시스템의 특성 분석 뿐만 아니라, 제어기 설계에 있어서도 반드시 거치는 중요한 과정으로서 자리를 잡고 있다.

시뮬레이션을 수행하는 두 가지 방법으로써 아날로그 컴퓨터에 의한 아날로그 시뮬레이션과 디지탈 컴퓨터에 의한 디지탈 시뮬레이션이 있다. 아날로그 컴퓨터는 적분기(integrator), 미분기(differentiator), 가산기(summer) 등의 전자회로 요소로 이루어져 있어서 이러한 요소들을 시스템의 상태방정식에 따라 구성하여 시뮬레이션을 수행한다. 그러나, 이 방법은 디지탈 컴퓨터가 발달한 후로는 거의 쓰이지 않게 되었다. 디지탈 컴퓨터로 시뮬레이션을 수행하기 위해서는 시스템에 대한 수학적인 모델을 가지고 프로그래밍하여 상태변수의 값을 구해내는 과정을 거친다. 이때 아날로그 시뮬레이션과는 달리 디지탈 시뮬레이션에서는 적분이나 미분 연산을 하는 특별한 연산자가 없기 때문에 수치해석적인 방법에 의해서 이들을 구현해야 하는데, 이 부분에 대해서 자세히 알아보기로 한다.

1) 디지탈 컴퓨터에서 수치해석적 방법에 의한 적분 연산

적분의 수치해석적인 방법으로 여러 가지 적분

공식이 가능하며 적절한 적분공식을 선택하는 것이 시뮬레이션에 있어서도 결정적이 될 수 있다. 가장 간단한 형태로는 Euler formula가 있다. (i+1)번째의 값은 i번째의 값과 그 때의 기울기에 의해서 다음과 같이 추정된다.

$$x(t+\Delta t)=x(t)+\frac{dx(t)}{dt}\cdot\Delta t \text{ or } x_{i+1}=x_i+\dot{x}\cdot\Delta t$$

수치해석적인 적분을 하는데 있어서 바로 전 단계의 변수값에만 의존하는 여러 가지 효과적인 방법들도 있지만, 여러 단계 전의 값까지 필요로 하는 방법들도 있으며 적분 단계 크기(integration step size)가 변수값이 변함에 따라 조정되는 방법들도 있다. 앞에서 말한 Euler method는 적분 단계 크기를 매우 작게 하지 않는 한 오차가 크게 발생하기 때문에 실제로는 거의 쓰이지 않고 있다. 시뮬레이션에서 많이 쓰는 적분공식으로 Simpson's rule(1/3 rule, 3/8 rule), trapezoidal rule, Runge-Kutta formula(2nd, 3rd, 4th order) 등이 있다. 특히 제어시스템을 위한 시뮬레이션에서는 4th order Runge-Kutta 방법을 많이 사용한다. 이 방법에서는 각 적분 단계마다 4번의 연산을 수행하여 다음 단계의 값을 보다 정확히 얻어내므로 Euler method에 비해서 오차가 매우 적다. 반면, 각 단계마다 연산을 많이 하므로 수행시간이 오래 걸린다는 단점도 있다. 다음은 4th order Runge-Kutta 공식이다.[2,3]

$$u_1=h\cdot f(t_i,y_i)$$
$$u_2=h\cdot f(t_i+h/2,y_i+u_1/2)$$
$$u_3=h\cdot f(t_i+h/2,y_i+u_2/2)$$
$$u_4=h\cdot f(t_i+h,y_i+u_3)$$
$$y_{i+1}=y_i+\frac{1}{6}\cdot(u_1+2u_2+2u_3+u_4)$$

여기서, $h=\Delta t$이고 f는 각 점에서의 y의 1차 미분계수이다.

2) 디지탈 컴퓨터에서 수치해석적 방법에 의한 미분 연산

아날로그 컴퓨터에서 미분기는 잡음(noise)의 문제를 안고 있다. 디지탈 시뮬레이션에서의 미분연산에도 이에 준하는 문제가 있는데, 시스템으로부터라기 보다는 시뮬레이션 방법으로부터 기인하는 발진성의(oscillatory) 불안정한 응답이 바로 그것이다. 수치해석 분야에서 이루어진 연구를 통하여 보면 수치적인 미분연산은 미분방정식의 풀이에 적합하지 않다는 사실을 알 수 있다. 이것은 곧 시뮬레이션에도 적합하지 않다는 것을 의미한다. 따라서 시뮬레이션에서는 미분연산을 거의 사용하지 않는 것이 바람직하다. 그러나 어쩔 수 없이 사용해야만 하는 경우(예를 들어, 미분 제어기)에는 다음과 같은 간단한 형태로 미분연산을 수행할 수 있다.

$$x(n)=\frac{dy(t)}{dt}\cong\frac{y(n+1)-y(n)}{\Delta t}$$

3) 수치적인 적분 vs. 시뮬레이션

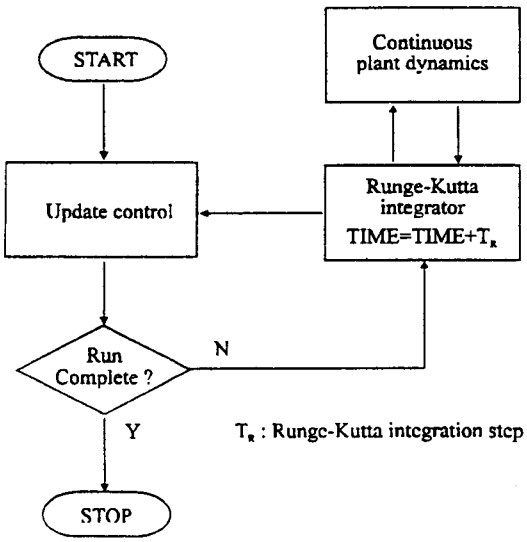
앞에서 살펴본 바에 의하면, 제어시스템에 대한 시뮬레이션은 미분방정식의 해를 수치해석적으로 구하는 과정에 불과하다고 생각할 수 있다. 그렇다면 수치적인 적분연산과 시뮬레이션의 차이는 무엇일까? 그 차이는 그리 크지 않겠지만 다음과 같은 시뮬레이션의 특징이 있다.[3]

(i) 시뮬레이션에서는 각 단계에서의 계산 결과를 그 순간의 시스템의 상태변수값으로 해석하는 것이다. 즉, 시뮬레이션이라는 것은 본질적으로 시간에 따라 시스템의 상태변수값을 구성하는 과정이다(time histories of state). 반면, 단순한 연산(computation)은 최종 결과를 염두에 둔 것으로써, 계산 과정에서의 각 단계의 값은 의미가 없다.

(ii) 시뮬레이션에서는 주어진 미분방정식을 전체 과정 안에 있는 각 단계 중의 하나로 보는 반면에 수치연산에서는 단지 그 방정식을 적분하는 것이 목적이 된다.

2. 연속시간 제어시스템(Continuous-time Control System)의 시뮬레이션

앞에서도 언급했듯이 연속시간 제어시스템에서는 시스템의 상태방정식이 미분방정식의 형태로



〈그림 1〉 연속시간 제어시스템 시뮬레이션 과정

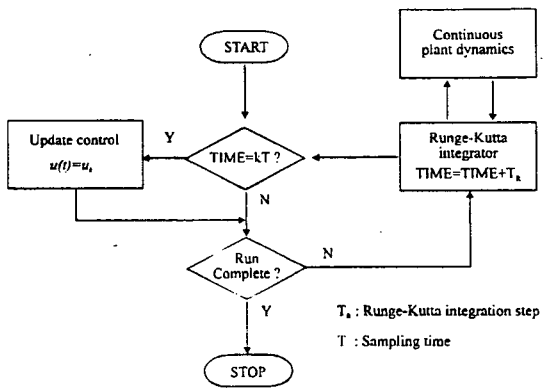
나타내어 진다. FORTRAN이나 C같은 프로그래밍 언어로서 시뮬레이션을 수행하려면 2장 1절에서 설명한 수치적인 적분공식에 의해 상태변수의 값을 구하는 과정을 서브루틴 등을 구성하여 사용자가 모두 처리해 주어야 한다. 〈그림 1〉은 연속시간 제어시스템의 시뮬레이션 과정을 흐름도(flow-chart)형식으로 나타낸 것이다.

DYNAMO, CSMP, ACSL 등으로부터 MATLAB, SIMNON, MATRIX$_X$, Program CC에 이르기까지 대부분의 시뮬레이션 소프트웨어 패키지에서는 시스템에 대한 미분방정식 또는 전달함수 형태의 모델이 주어지면, 사용자는 복잡하고 시간이 오래 걸리는 프로그래밍 과정을 거치지 않고도 신속하게 시스템의 시뮬레이션을 수행할 수 있게 해 주고 있다.[6~9] 사용자는 이미 소프트웨어 안에 들어 있는 Runge-Kutta 서브루틴 등을 호출하여 적분 단계 크기(integration step size), 수행시간 등을 지정하고, 상태변수의 초기값을 설정해 주기만 하면 된다. 그리고, 대부분의 이러한 소프트웨어들은 graphical tool까지 포함하고 있으므로, 결과에 대한 확인이 용이하고 보다 좋은 성능을 나타내도록 제어기를 다시 설계하는 과정이 쉽게 반복될 수 있다.

3. 이산시간 제어시스템(Discrete-time Control System)의 시뮬레이션

이산시간 제어시스템에서는 제어기를 포함한 전체 제어시스템이 이산시간시스템이 된다. 즉, 제어기를 제외한 원래 시스템(original system)은 연속시간 제어시스템에서와 마찬가지로 연속시간에 따라 동작하는 연속 시스템(Continuous System)이다. 이러한 연속 시스템에서의 상태변수 또는 출력을 Sample-and-Hold하여 이산시간에서의 값에만 관심을 두고 이에 따라 이산시간 제어기를 구성하는 것이 이산시간 제어시스템에서의 설계 과정이라고 할 수 있다. 따라서, 이산시간 제어시스템에 대한 시뮬레이션은 두 가지 방법으로 나눌 수 있는데, 하나는 차동방정식으로 주어진 상태방정식에 따라 전체 제어시스템을 iteration함으로써 시뮬레이션을 수행하는 것이고, 또 다른 방법은, 원래 시스템(original system)은 연속시스템이므로 미분방정식으로 모델링된 것을 수치적인 적분을 통하여 시뮬레이션을 하면서 그 시스템에 가해지는 제어입력은 이산시간마다 갱신(update)하는 것이다. 〈그림 2〉는 두번째 방법으로 시뮬레이션을 할 때의 흐름도이다.[10]

한편 적응제어시스템과 같이 복잡한 시스템의 경우에는, 상태방정식으로 주어진 연립 차동방정식의 시뮬레이션을 위해서 상태변수들 간의 순서재배치(reordering)와 시간축 이동(time-shifting)이 필요하며, 적절한 초기값을 설정해 주어야 한



〈그림 2〉 이산시간 제어시스템 시뮬레이션 과정

다. 최근에 R. D. Ellis[11] 등은 이러한 문제에 대한 수학적인 접근과 동시에, 자동으로 상태변수의 순서 재배치와 시간축 이동을 해 주는 알고리즘을 고안하여 ISIM이라는 시뮬레이션 패키지를 개발하였다. 이 패키지를 이용하면 사용자는 주어진 차동방정식을 있는 그대로 입력해 주기만 하면 된다고 한다.

III. 로봇 시뮬레이션

이 장에서는 로봇공학에 시뮬레이션을 적용하는데 있어서 시뮬레이션이 성공적으로 수행되기 위하여는 어떤 요소들이 필요한가를 살펴보기로 한다.

1. 로봇 시뮬레이션의 요소

먼저, 사용되는 시뮬레이션 꾸러미(package)를 자유로이 사용할 수 있는 사용자가 있어야 하며, 소프트웨어를 뒷받침할 수 있는 실시간 하드웨어가 필요하다. 현재 GUI(graphic user interface)의 발전으로 사용자는 더욱 쉽게 소프트웨어를 사용할 수 있으며, 하드웨어의 성능도 최근 수년간 비약적으로 개량되어 현재는 PC상에서도 복잡한 작업공정의 시뮬레이션이 가능하게 되었다.

다음으로는 CAD 시스템에 대한 전반적인 이해를 토대로 실제 작업용 로봇 및 워크셀(workcell)들의 정확한 모델이 CAD 장비를 사용하여 기구학적으로 용이하게 모델링될 수 있어야 한다. CAD 모델링은 기존의 많은 CAD 장비들로 가능하고, 시뮬레이터는 이들과의 데이타를 공유함으로써 상호 정보를 교환할 수 있다. 이러한 모델들은 항상 최신의 것으로 유지되어야 하며, 실제 작업공정과의 차이가 없도록 캘리브레이션(calibration)을 거쳐야 한다.

세번째로는 워크셀의 설계이다. 로봇을 시뮬레이션하는 경우에, 이것은 비단 로봇만이 시뮬레이션되는 것이 아니라 로봇의 엔드-이펙터, 컨베이어, 파트 피더(part feeder), 공작기계, AGV (autonomous guided vehicle) 등도 함께 시뮬레이션되어야 한다. 이를 위하여 워크셀의 설계는 다음의 작업들을 포함한다.

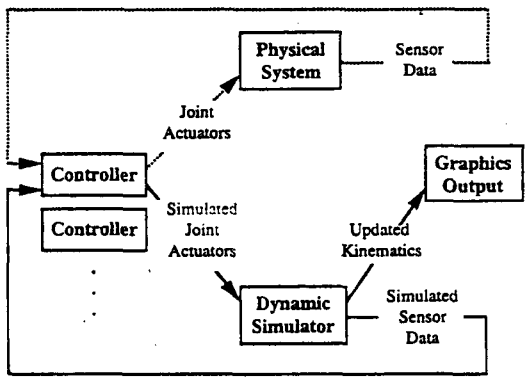
- 워크셀 구성원의 배치 및 수정
- 로봇에 엔드-이펙터 부착
- 워크셀 구성원의 작업 내용 결정
- 로봇과 워크셀들의 수행도 검증

워크셀의 기능검증은 3차원 그래픽을 이용하여 이루어지며, 로봇을 중심으로 작업이 가능한 영역내에 있는가, 충돌을 방지하거나 회피할 수 있는가, 개략적인 사이클 타임은 어느 정도인가 등이 반복적인 설계를 통하여 최적화된다. 필요에 따라서는 모델의 변경을 요구하기도 한다.

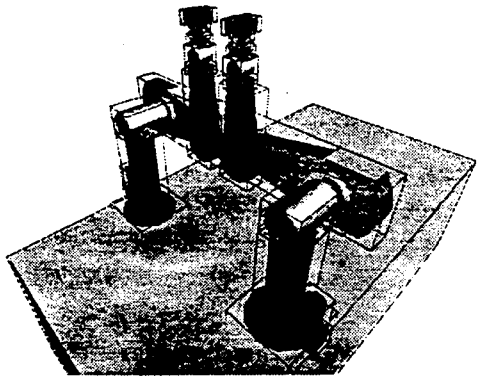
네번째로 로봇의 동작계획 및 동작 시뮬레이션이다. 로봇의 동작계획은 로봇과 작업환경사이에서 상호적으로 이루어져야 한다. 이러한 일련의 과정이 〈그림 3〉에 나타나 있다. 다이나믹 시뮬레이터는 조인트 제어기로부터 제어입력을 받아들여 각 대상물의 운동방정식을 이용하여 시뮬레이션된 센서 데이타를 다시 제어기로 피드백한다. 한편 기구학적인 해는 업데이트되어 그래픽 출력으로 사용자에게 보여주는 과정을 반복한다.

이러한 로봇의 동작계획 및 동작 시뮬레이션의 주요작업으로는 :

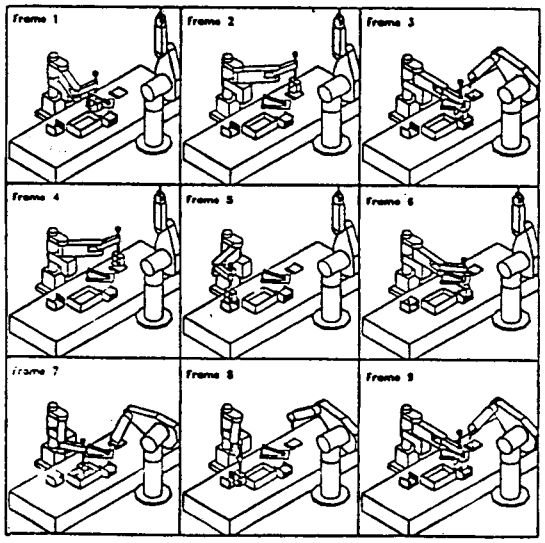
- 로봇과 작업환경, 또는 로봇끼리의 충돌/접촉 감지/예측(그림 4)
- 로봇이 그 작업영역을 벗어나는 가에 대한 감지



〈그림 3〉 다이나믹 시뮬레이션의 전체 구조도



〈그림 4〉 두 대의 로봇이 작업하는 경우의 충돌회피 시뮬레이션



〈그림 5〉 동작 시뮬레이션을 통한 로봇 움직임과 작업의 최적화 및 동작 분석

- 로봇의 작업경로의 최적화
- 로봇이 작업수행중에 제한된 속도/가속도/토크들을 벗어나는 가를 감지
- 로봇 동작의 유연화, 최적화 및 동작 분석(그림 5)

등을 포함한다.

다섯번째로 필요한 작업이 캘리브레이션(calibration)이다. 이 작업은 시뮬레이션과 실제 환경간의 차이를 보정하는 단계로서, 차이가 나는 원인을 규명하고, 시뮬레이션 모델을 정기적으로 보정하는 데 사용된다. 차이가 나는 근원은 주로 워크셀의 실제 배치의 차이, 로봇의 기하학적/기구학적 모델의 차이, 로봇의 반복작업으로 인한 위치오차의 누적 등을 들 수 있다.[17]

2. 시뮬레이션의 적용 사례

수년전만 하더라도 로봇 및 작업환경의 시뮬레이션은 주로 워크스테이션급이상의 중·대형 컴퓨터에서 수행되어 이를 위한 설치비 또한 상당한 비용으로 작용하였었다. 하지만 이제는 컴퓨터의 비약적인 발달로 PC상에서도 유용하게 쓸 수 있는 시뮬레이터들이 속속 개발되고 있다. 그 중의 대표적인 예가 영국의 Robot Simulation사가 개발하여 국내에서도 시판중인 WORKSPACE이다.[18]

이러한 시뮬레이터의 주요특징은 3차원 CAD를 기본으로 입체기하학, Bezier surfaces, dimensioning, 그리고 산업표준형 DXF, IGES interface 등을 포함하며, 모든 모델은 wireframe, hidden-line, shaded로 변경이 가능하다. 또한 로봇 프로그래밍 언어로 KAREL, VAL, VALII, AS, PDL/2 등외에도 많은 종류의 언어가 사용가능하고, 용이하게 오프-라인프로그래밍으로 로봇의 동작계획, 동작 시뮬레이션 및 캘리브레이션을 수행할 수 있다. 보다 자세한 특징은 참고문헌 [18]에서 찾아볼 수 있다.

설치사례로는 터키의 CIMTAS사의 대형 철근 'I' 빔의 아크용접분야에서 작업환경의 10축 움직임을 모의 실험함으로써 로봇선택의 기준을 마련하고, 로봇의 작업범위 내에서 충돌을 방지하는데 사용되어 설치에 따른 시간과 자금을 줄일 수 있었다. 이외에도 영국의 국방부, GLAXO, 미국의 크라이슬러와 포드 자동차 회사, ABB ROBOT사, 일본의 NTT사에의 여러 분야에서 그 성능을 인증받고 있음이 보고되어 왔다.

IV. 결 론

시뮬레이션은 많은 학문 및 연구분야에서 사용되는 과학적 문제 해결 기법이다. 특히 제어공학에서는 제어 대상 시스템 분석, 제어기 설계 등의 목적을 달성할 수 있는 아주 유용한 도구가 된다. 그러나 시뮬레이션을 단순히 간단한 문제해결을 위한 도구로서 사용하기보다는 시뮬레이션 소프트웨어 패키지의 다양한 기능을 이용하여 시스템에 대한 보다 깊은 이해와 물리적 의미까지도 얻을 수 있음을 간과해서는 안될 것이다. 또는 프로그래밍 언어를 이용해서 직접 시뮬레이션 프로그램을 작성하는 것도 제어시스템에 대한 이해를 돕는데 기여를 할 것이다.

시뮬레이션이 제어시스템을 설계하는데 있어서 부수적인 도구가 되고 있는 것이 사실이지만, 한편으로는 특별히 제어시스템을 시뮬레이션하는 데서 발생하는 어려움들을 해결해 나가는 것도 중요한 연구주제가 될 수 있을 것이다.

한편 로봇공학에의 응용에서 살펴본 것처럼 시뮬레이션의 장점은 직접 사람과 로봇, 그리고 작업환경이 움직이는 대신에 컴퓨터로서 발생가능한 문제점들을 감지하고, 원하는 설계 방식에 따라 오프-라인 프로그래밍으로 최적화할 수 있으므로 시간과 장비, 예산, 원가 절감의 종합적인 성능향상을 꾀할 수 있게 되었다는 점이다. 더우기 편리한 소프트웨어와 하드웨어의 비약적인 발전은 싼 값으로 이러한 시뮬레이션의 구현이 가능하게 되었다.

앞으로 이러한 시뮬레이션에 관한 연구는 지능형 로봇과도 연계되어 더욱 발전되어 나갈 것으로 기대된다.

참 고 문 헌

〔1〕 A. M. Colella, M. J. O'Sullivan, and D. J. Carlino, "Systems Simulation : Methods and Application", D.C. Health and Company, 1974.

〔2〕 R. J. Kochenburger, "Computer Simulation of Dynamic Systems", Prentice-Hall, 1972.

〔3〕 N. Deo, "System Simulation with Digital Computer", Prentice-Hall, 1983.

〔4〕 G. A. Korn, "Interactive Dynamic System Simulation", McGraw-Hill, 1989.

〔5〕 K. Ogata, "Discrete-Time Control Systems", Prentice-Hall, 1987.

〔6〕 "ACSL-Advanced Continuous Simulation Language : Referencd Manual", Mitchell and Gauthier Associates, 1986.

〔7〕 A. L. Pugh III, "DYNAMO User's Manual", 5th ed., The MIT Press, 1976.

〔8〕 F. H. Speckhart and W. L. Green, "A GUIDE TO USING CSMP-The Continuous System Modeling Program : A Program for Simulating Physical Systems", Prentice-Hall, 1976.

〔9〕 H. Elmqvist, K. J. str m, T. Sch nthal, and B. Wittenmark, "SIMNON User's Guide for MS-DOS Computers", SSPA Systems, 1990.

〔10〕 F. L. Lewis, C. T. Abdallah, and D. M. Dawson, "Control of Robot Manipulators", Macmillan Publishing Company, 1993.

〔11〕 R. D. Ellis, R. Ravikanth, and P. R. Kumar, "Automating the Simulation of Complex Discrete-Time Control Systems : A Mathematical Framework, Algorithms, and a Software Package", IEEE Trans. Automat. Contr., vol. 39, pp. 1795~1801, Sep. 1994.

〔12〕 서석환, "Robotics와 CAD", 정보과학회지, 제8권 제3호, 1990년 6월, pp.23~31

〔13〕 F. Nagashima, Y. Sato, M. Hirata, M. Ueki, T. Maruyama and T. Uchiyama, "ロボトシミュレタ-FIRST-の開發", 第12

回日本ロボト學會學術講演會, 1994년 11월, pp. 813~814

〔14〕 A. M. Eydgahi, "Using simulation for designing robotic environment," in Proc. Int. Conf. Automation, Robotics and Computer Vision (ICARCV'94), 1994, pp. 1756~1760.

〔15〕 G. K. Adam and E. Grant, "QMTOOL-A qualitative modelling and simulation CAD system for designing automated workcells," in Proc. 1994 IEEE Int. Conf. Robotics and Automation, pp. 1141~1146.

〔16〕 P. U. Lee, D. C Ruspini, and O. Khatib, "Dynamic simulation of interactive robot environment," in Proc. 1994 IEEE Int. Conf. Robotics and Automation, pp. 1147~1152.

〔17〕 R. Judd and A. Knasinski, "A technique to calibrate industrial robots with experimental verification," IEEE Trans. Robotics and Automation, vol. 6, no. 1, Feb. 1990, pp. 20~30.

〔18〕 John P. Owens, 천세훈, "로봇에는 왜 시뮬레이션이 필요한가", 제9회 로보틱스 및 자동화 연구회 논문집, 1995년, pp. 49~54

저 자 소 개

李 柱 張

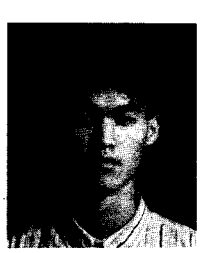
1948年 11月 14日生

電子工學會誌 第 20卷 第 3號 參照

金 聖 祐

1967年 11月 26日生
1990年 2月 한국과학기술원 전기 및 전자공학과 졸업(공학사)
1992年 3月~현재 한국과학기술원 전기 및 전자공학과 박사과정 재학중

1990年~1991年 삼성전기 연구원



李 鍾 權

1972年 11月 27日生
1994年 2月 한국과학기술원 전기 및 전자공학과 졸업(공학사)

1994年 3月~현재 한국과학기술원 전기 및 전자공학과 석사과정 재학중