# 객체지향 방식의 프로그래밍과 차량 파워 시스템

## Object-Oriented Programming and Automotive Powersystem

정 병 용, 조 동 일
(Byeong Yong Jeong and Dong-il Cho)

**Abstracts** : This paper represents a new powertrain simulation methodology using the object-oriented programming paradigm. The advantage of the object-oriented paradigm is the module interchangeability and simulation flexibility. Powertrain subsystems and controller modules are implemented using the MEX files in MATLAB Simulink in this paper, preserving module interchangeability. Currently, the required CPU time on a 75MHz Pentium PC is about three times the real time. It is anticipated that the "Automotive Powersystem Toolbox" being developed in this research would be of much utility in designing subsystem controllers as well as in designing subsystem mechanicals.

**Keywords** : object-oriented programming, automotive powersystem, simulation, simulink, MEX files

### Ⅰ. 서론

객체지향의 프로그래밍 기법은 복잡한 소프트웨어 시스템을 분석하고 실제 시스템을 상위 시스템에서 하위 시스템으로 모델링하는 데에 편리한 방법론이므로 CACSD (Computer-Aided Control Systems Design)에 유용하게 적용되어 질 수 있는 패러다임이다. 물리적인 시스템에 대한 시뮬레이션이 소프트웨어에 의한 실제 시스템의 표현이라는 점을 고려하면 차량 파워 시스템의 객체지향의 구현을 위한 환경으로 시뮬레이션의 방법으로 객체지향의 개념을 도입할 경우 많은 이점이 얻을 수 있음을 예상할 수 있다[2].

본 논문에서는 [3]의 실차실험을 통해 검증된 8개의 상태변수를 가지는 차량 파워 시스템 모델을 구현하였다. [3]에서 알 수 있듯이 하나의 상태변수 값을 계산하기 위해서는 다른 상태변수의 값들이 많이 필요하며, 서로 밀접하게 커플링되어 있다. 그러므로 중간 계산값을 공유하는 정도와 물리적인 의미에 따라서 경계를 나누어주면 계산상의 효율과 단위 시스템 수준에서의 재사용성을 동시에 보장할 수 있다. 본 논문에서는 이러한 조건을 만족시키는 경계로 차량 파워 시스템을 엔진, 트랜스미션, 드라이브트레인으로 나누어주었으며 연료분사 제어기와 클러치의 유압 제어기를 구현하였다.

차량 파워 시스템인 객체지향의 구현을 위한 환경으로는 Simulink를 사용하기로 한다. 그러나 Simulink의 기본 라이브러리만으로는 비선형성을 정의하기에는 부자연스럽고 계산 시간이 많이 소요되므로, 본 논문에서는 C MEX 파일을 사용하여 시스템의 비선형 동적특성을 정의하고, 사용자 인터페이스와 수치 적분 커널은 Simulink를 이용한다[5]. 이와 같이 정의된 시스템에 대해서 시스템의 변화가 생기는 경우에도 모듈 단위의 재사용이 가능하며, 연료분사 제어기로 두 종류의 제어기를 구현하여 제어기의 설계와 교체가 쉽게 이루어질 수 있음을 보인다.

### Ⅱ. 객체지향의 CACSD

이 장에서는 차량 파워 시스템의 객체지향적 구현의 환경으로 본 논문에서 사용한 MATLAB/Simulink의 여러 가지 구현방법을 비교하고, 실제의 구현에 사용한 방법과 유의할 점을 알아본다.

1. Simulink상에서의 시스템의 정의

MATLAB은 시스템의 정보를 행렬 형태로 저장하며, Simulink라는 시뮬레이션 툴을 가지고 있다. Simulink는 경계로 나누어진 단위 시스템을 자연스럽게 정의할 수 있고 시각적인 설계 환경을 가지고 있으며, 여러 설계 과정을 수행하는데 필요한 툴들을 갖추고 있으므로 차량 파워 시스템의 객체지향적인 구현의 환경으로 Simulink를 사용한다[1].

시스템의 입출력 특성을 구현하고 나머지 부분을 은닉시킴으로써 라이브러리화 하는 방법은 그림 1과 같이 masking과 grouping의 두가지 방법으로 나뉘어 진다. Grouping은 입출력을 제외한 나머지 부분을 단순히 은닉하는 것으로 가변적인 요소를 인자로 가지지 못하므로 변화를 수용하지 못하며, grouping은 가변적인 입력 인자를 가질 수 있으므로 상황의 변화를 수용할 수 있는 구조이다.

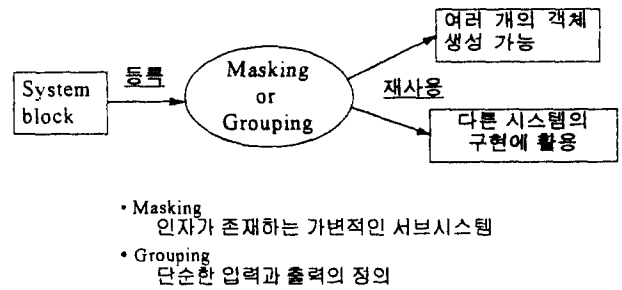



그림 1. 단위 시스템의 등록 및 재사용.
Fig. 1. Registration and reuse of unit systems.

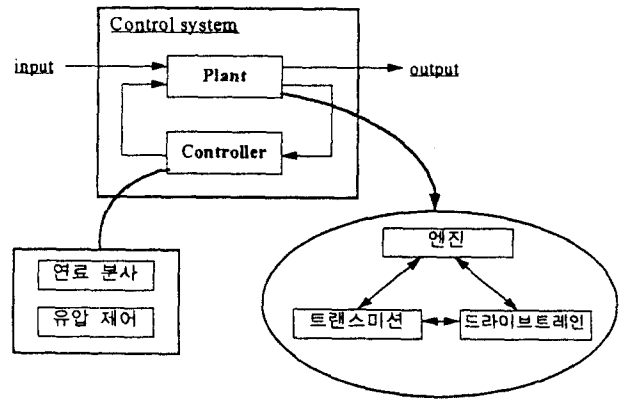



그림 2. 파워 시스템의 계층적 구조.
Fig. 2. Hierarchical structure of the automotive powersystem.

접수일자 : 1995. 12. 6., 수정완료 : 1995. 3. 2.
정병용, 조동일 : 서울대학교 공과대학 전기공학부
※ 본 연구는 과학재단(목적기초과제번호:95-0200-09-02-3)의 지원하에 수행되었고 이에 감사드립니다.

복잡한 시스템을 계층적인 구조로 표현하면 시스템의 구조를 쉽게 파악할 수 있으며 구현된 단위 시스템으로부터 전체 시스템을 쉽게 구성할 수 있는 장점이 있다. 3.2에서 언급하게 될 차량 파워 시스템의 객체지향적 구현을 위한 경계를 이용한 전체 시스템의 계층적 표현은 그림 2와 같고 이때 각각의 구성 블록들은 그림 6과 같이 이루어진다.

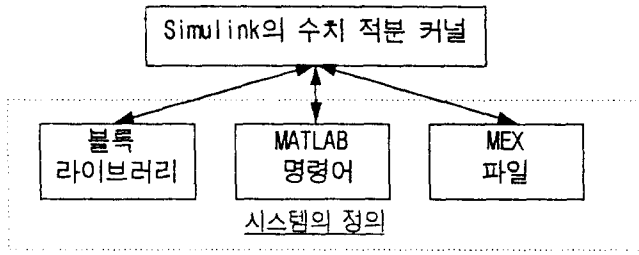



그림 3. Simulink의 계산 원리.
Fig. 3. Principle of simulink computation.

Simulink의 계산이 이루어지는 원리는 그림 3과 같이 시스템의 정의와 수치적분 커널이 분리되어 있는 구조이며, 시스템에 대한 정의는 기본 블록 라이브러리, MATLAB 명령어, 여러 툴 박스의 함수들, MEX 파일 등을 통하여 이루어 진다. 이와 같은 구조는 시스템의 정의를 변경하지 않고 수치적분 방법을 변경시키는 것을 가능하게 하여 준다. 그리고 이러한 방식의 시스템의 정의를 위해서는 시스템의 정의 부분과 수치적분 커널의 정보 교환을 위하여 정해진 프로토타입을 따라야 한다.

Simulink가 내부 명령어를 수행하기 위하여 거치는 단계는 다음과 같이 나뉘어 진다.

- 명령어를 읽어 해석(parsing)
- 연산자의 타입을 결정
- 연산을 위한 기억장소를 할당
- 연산의 수행

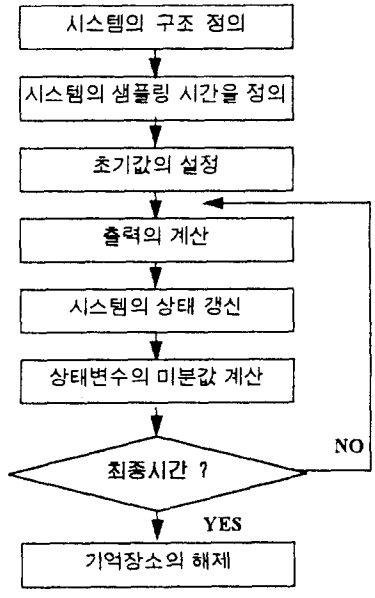



그림 4. MEX 파일의 흐름도.
Fig. 4. Flowchart of MEX file.

위와 같은 방식의 명령어 수행은 인터프리터의 특징을 가지며 시뮬레이션에는 부적합한 구조이다. 시뮬레이션에서는 각각의 시간에 대하여 거의 동일한 연산을 반복하기 때문에 위의 동작들 중 앞의 세단계를 매번 반복하는 것은 계산시간을 낭비하는 결과를 초래한다. 그러므로 프로그래밍 언어를 사용하여 시스템을 정의하고 이것을 컴파일하여 오브젝트코드의 수준에서의 실행을 함으로써 위와 같은 계산시간의 낭비를 해결할 수 있다. 이때 생성되는 코드를 MEX 파일이라 하며 앞서 언급한 바와 같이 계산의 커널과의 정보 교환을 위한 정해진 프로토타입을 갖추어야 한다. 그림 4는 이러한 과정을 도식적으로 나타내어 준다.

2. 구현 방법의 선택

Simulink상에서의 여러가지 구현방법들 중 차량 파워 시스템의 시뮬레이션을 위한 계산 방법을 선택할 때 고려되어야 할 사항은 다음과 같다.

- 서브 시스템 사이의 연결성
- 시각적인 계층성
- 계산시간
- 비선형성을 처리하는 효율성

단위 시스템으로 경계를 나누어 구현하고 이를 연결하여 전체 시스템을 구현하게되면 구현의 단위가 작아지므로 각각의 단위 시스템에 대해 변화의 가능성이 작아지게 된다. 이말은 재사용의 가능성이 높아지는 것을 의미한다. 그리고 입출력의 특성만을 파악하고 사용이 가능하게 하기 위해서는 시각적인 계층성을 갖추어야 한다. 계산의 효율성에 대한 고려가 필요한 경우는 조건의 분기에 의한 비선형성이 존재하는가에 의하여 좌우된다. 프로그래밍 언어의 경우에는 조건을 검사하여 만족되지 않으면 계산을 수행하지 않는다. 그러나 Simulink는 시간의 흐름에 따라 전체의 시스템을 거슬러 가며 계산을 하므로 모든 식에 대하여 계산을 한 후 조건에 맞는 값을 고르게 된다. 즉 조건을 만족시키지 않는, 필요없는 식에 대한 계산이 이루어진다. 계산시간의 비교를 위해 Van der pol 식을 모델로 하여 1000초 동안의 시뮬레이션을 수행하여 보았다. 계산은 Pentium 60 MHz에서 이루어졌으며 결과는 표 1과 같다. 계산시간을 비교하면 인터프리트의 과정을 필요로 하지 않는 오브젝트코드 수준에서의 구현과 Simulink의 기본 라이브러리를 사용하는 것이 빠름을 알 수 있다. 표 1은 위와 같은 고려 사항에 대하여 각각의 구현 방법을 비교한 것이다.

표 1. 구현방법에 따른 특성의 비교.
Table 1. Comparison of various implementational methods.

| 구현 방법 | 서브시스템의 연결성 | 시각적 계층성 | 계산시간 (초) | 비선형적 효율성 |
|---|---|---|---|---|
| Simulink+Block library | O | O | 0.55 | X |
| Simulink+ MATLAB function | O | O | 14.39 | O |
| Simulink+M-file | O | O | 22.52 | O |
| Simulink+C MEX file | O | O | 0.94 | O |
| M file | X | X | 11.81 | O |
| C MEX file | X | X | 0.27 | O |

표 1은 Simulink의 환경에서 C MEX 파일보다 블록 라이브러리가 더 빠르게 계산을 수행함을 알 수 있다. 그러나 Van der pol 식의 경우에는 조건의 분기가 존재하지 않으므로 비선형적인 효율성에 대한 영향을 볼 수 없다. 차량 파워 시스템에서는 토크 변환기나 기어의 위치를 결정하는 부분에서 조건의 분기에 의한 계산이 존재하므로, 블록 라이브러리에 의한 구현은 C MEX 파일에 의한 구현보다 계산이 느리게 된다. 그러므로 차량 파워 시스템을 구현하기 위한 방법으로 비선형성이 강한 부분에 대해서는 네번째의 방법을 적용하고, 조건의 분기가 없는 선형적인 부분에 대해서는 기본의 블록 라이브러리를 사용하여 시스템을 구현

하도록 한다.

## Ⅲ. 차량 파워 시스템의 객체지향적 구현

Simulink의 환경에서 차량 파워 시스템을 객체지향적으로 구현하기 위하여 고려해야 할 사항들을 알아본다. Simulink는 가변 시간간격의 방법으로 수치 적분을 수행하므로 계산속도가 빠르고 주어진 정확도를 만족시키는 이점이 있지만, 이로 인하여 비선형성이 강한 시스템의 구현시 시간간격에 대한 고려가 필요하다. 또 재사용성을 가지는 시스템의 구현을 위해서는 시스템 구현의 경계 문제를 생각해 주어야 하며 이 경계에 의한 입출력 변수의 정의를 살펴본다.

수치 적분 방법은 Simulink의 커널이 담당하는 역할이며 내정된 방법은 가변 시간간격 계산 방법이다. 계산 시간간격이 가변적임으로 인하여 시스템의 정의에 미치는 영향으로는 수치 적분 커널 인자에 대해 미치는 영향과 MEX 파일 안의 함수 구현에 미치는 영향으로 나누어 볼 수 있다. 특히 시간 지연 함수의 구현은 고정 시간간격일때에 비하여 크게 달라진다.

### 1. 가변 시간간격 시뮬레이션

가변 시간간격 시뮬레이션이란 정확도를 주어진 범위 안에서 유지하기 위하여 계산 시간간격을 가변적으로 조정하는 방법이며, 연속시간 시스템의 경우 미분함수의 시간에 따른 변화율에 의하여 완만하게 변하는, 변화율이 작은 시간구간에서는 시간간격을 크게 하고 변화율이 큰 구간에서는 시간간격을 작게 한다. 보통의 시뮬레이션 경우 계산의 초기에는 시간간격이 작아 계산 횟수가 많으나 정상상태에 이르게 되면 시간간격이 커짐으로 인하여 계산의 횟수가 현저하게 줄어든다. 그러므로 시간간격이 고정된 계산방법에 비해 전체적으로 계산의 횟수가 줄어들어 계산시간이 줄어드는 결과를 가져온다. 가변 시간간격 시뮬레이션을 하기 위하여 인자로 주어져야 하는 요소는 다음과 같다.

- 정확도
- 최대의 시간간격
- 최소의 시간간격
- 수치 적분 방법

가변 시간간격의 시뮬레이션의 경우에는 계산의 속도와 신뢰성 사이의 절충이 필요하다. 시간간격의 최소값이 작을수록 함수의 변화가 심한 부분에서는 많은 계산을 하게 되므로 주어진 정확도를 만족시킬 수 있지만 이에 따라 많은 계산횟수를 요하므로 계산의 속도는 현저하게 느려진다. 그러므로 원하는 정확도를 만족시키는 범위에서 가능한 큰 시간간격의 최소값을 정해 주는 것이 좋다. 또 한가지 고려되어야 할 점은 전체의 시스템을 함수적으로 분석하여 구현하는 것이 불가능하다는 것이다. 가변 시간간격의 시뮬레이션은 수치적분 커널이 임의의 상태변수 값, 입력 값 그리고 시간에 대한 상태변수의 미분값을 필요로 하므로 시스템의 정의가 연속시간, 이산시간 시스템 각각의 경우에 대하여 시간, 상태변수, 입력 변수를 입력으로 하여 미분값을 계산하는 형태의 함수가 되어야 한다.

### 2. 시간 지연 함수의 구현

차량 파워 시스템의 엔진 서브시스템은 엔진 도시 토크를 계산하기 위하여 시간 지연 함수의 구현을 필요로 하며 이를 위하여 큐를 사용한다. 이때 현재 계산시간의 새로운 값이 들어가고 가장 오래전 시간의 값이 버려지며, 실제의 값들이 처음 기억되어 유지되는 위치는 변하지 않으며 갱신될 값과 버려질 값에 대한 위치 정보만 변하게 된다.

Simulink는 가변 시간간격 시뮬레이션을 수행하므로 고정 시간간격 시뮬레이션에서와는 다른 시간 지연 큐의 구현이 필요하다. 고정 시간간격의 경우 지연된 값의 위치를 찾아내는 방법은 간단하다. 현재의 시간에 대한 값의 위치에서 지연된 시간을 시간간격으로 나누어준 값 만큼을 빼어준 위치가 지연된 값이 저장되어 있는 위치가 된다. 그러나 시간간격이 가변적일때는 지연된 값이 저장된 위치를 시간 지연만의 계산에 의해 알 수 없으며 큐의 저장된 값들을 일일이 찾아가며 시간을 비교하여 값을 찾아낸다. 또한 변수값이 저장될 때 시간도 같이 저장해 주어야 하므로 이중적인 기억장소가 필요하게 된다. 그러므로 이와 같은 탐색 알고리듬에 의해 큐의 처음부터 매번 찾는 것은 시뮬레이션시 비교에 의한 계산시간의 오버헤드를 초래한다. 이러한 오버헤드를 줄이기 위해서는 새로운 탐색 알고리듬이 필요하다.

시간 지연은 엔진의 회전속도에 대한 함수이고, 엔진의 회전속도는 시간에 대하여 연속적인 함수이다. 그러므로 시간 지연의 크기 역시 시간에 대한 연속적인 함수로 생각할 수 있다. 즉 시간간격이 작은 경우 전시간의 시간 지연 크기와 현재시간의 시간 지연 크기의 차이는, 충분히 작음을 의미한다. 이말은 전시간에 지연된 값을 찾은 위치를 알고 있다면 현재 시간의 지연된 값의 위치는 그로부터 충분히 가까운 위치에 있음을 뜻한다. 그러므로 시간 지연 큐의 구현시 전시간의 지연된 값을 찾은 위치를 기억하고 다음 번의 큐를 탐색할 때 그 위치에서 탐색을 시작한다. 이와 같은 탐색 알고리듬은 고정 시간간격 계산 방법의 시간 지연 큐의 구현에 비해 계산속도가 거의 떨어지지 않는다. 그림 5는 이상의 알고리듬을 도식적으로 나타내었다.

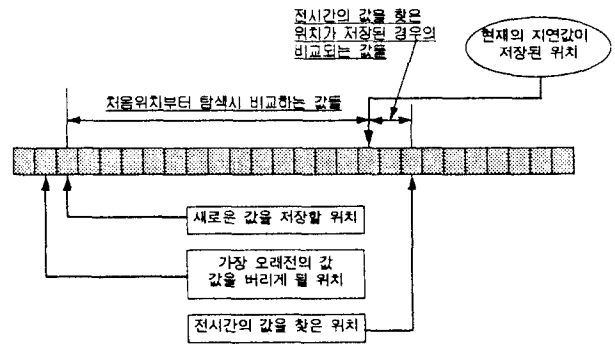



그림 5. 지연시간 큐의 구현.
Fig. 5. Implementation of the time delay queue.

### 3. 차량 파워 시스템의 고려 사항

가변 시간간격 계산 방법을 차량 파워 시스템에 적용할 때의 고려할 사항을 알아보자. 차량 파워 시스템은 강력한 비선형성으로 말미암아 초기조건에 의하여 계산의 정상상태에 이르는 시간이 결정되며 계산의 초기상태에는 시간간격이 작으므로 많은 계산이 수행된다. 실시간에 가까운 계산이 요구되는 경우에는 초기조건을 얼마나 살 잡아 주는가에 의해서 계산시간에 결정되기도 하며, 수렴과 발산을 결정한다[6].

연료분사 제어기는 엔진의 펄스에 의하여 구동된다. 이때 엔진의 회전속도에 의하여 발생 주기가 결정되는 펄스는 계산 시간간격이 최대로 커졌을 때에도 정해진 각도의 회전에 대해서 펄스의 발생 주기를 만족시킬 수 있어야만, 얻어진 연료분사 제어의 시뮬레이션를 신뢰할 수 있다. 즉 최대 엔진 회전속도의 펄스 발생 주기보다 최대 시간간격이 작아야 한다. 또한 시간간격의 최대 최소값은 제어기의 종류와 이득의 크기에 따라서 가변적으로 적용하여야 한다. 일반적으로 이득의 크기가 큰 경우에는 작은 시간간격으로 계산을 수행한다.

시스템의 입력이 되는 외란의 크기 역시 시간간격의 크기를 결정한다. 외란의 크기가 크다는 것은 함수의 시간에 대한 변화율이 큰 것을 의미하므로 시간간격을 작게 하여 계산을 수행하여야 한다[12]. 이밖에 트로틀 밸브의 각도, 부하토크, 점화의 선행정도, 유체의 시간상수 등은 입력함수의 시간에 따른 변화율을 결정하여 주는 요소로서 계산속도와 함수의 수렴성과의 절충을 위해서는 수렴과 신뢰성을 보

장하는 최대한 큰 시간간격을 지정한다.

4. 차량 파워 시스템의 경계 구분

이제 우리가 생각하여야 할 것은 차량 파워 시스템을 적당한 물리적인 의미를 가지며 자료의 은닉성을 보장하는 경계로 나누어 주는 일이다. 앞에서 언급된 것과 같이 시스템의 경계를 상태변수에 의하여 나누어 주는 것은 자료의 은닉성을 해치게 된다. 예를 들어 엔진 시스템을 상태변수별로 구현한다면 흡입 매니폴드의 공기의 질량의 변화율과 엔진의 회전속도, 실제로 분사된 연료의 질량 등은 모두 각기 하나의 시스템을 이루게 된다. 이 경우 만일 엔진의 회전속도의 미분값을 계산하려고 한다면 흡입 매니폴드의 공기의 질량의 변화율을 필요로 하며 이 값은 다른 서브시스템의 출력값이 아닌 시스템의 상태변수이므로 외부에서 참조할 수 없는 값이다. 물론 이 두 시스템은 별개의 독립된 시스템이 아닌, 우리가 편의상 경계를 나누어준 시스템이다. 그러나 이 시스템들이 나중 다른 시스템의 구현을 위해서 사용되었을 때는 다른 별개의 시스템이 될 수 있으며 이때는 다른 시스템의 상태변수를 참조하는 불가능한 일이 된다. 이러한 이유로 말미암아 시스템의 각각의 상태변수별로 다른 서브시스템에 구현하는 것은 자료의 은닉성을 해치게 된다. 그러므로 계산시 상태변수의 값을 공유하는 정도와 물리적인 의미를 가지는 정도를 고려하여 시스템의 경계를 구분하여야 한다.

본 논문에서는 엔진, 트랜스미션, 드라이브트레인 등과 제어기들을 차량 파워 시스템을 구현하는 서브시스템의 경계로 설정한다. 이와 같은 경계에 대해서 상태변수의 값들에 대한 참조가 경계의 밖에서는 별로 이루어지지 않으며, 경계 안의 시스템 그 자체로 물리적인 의미를 갖게 된다.

5. 단위 시스템의 입출력 설정

단위 시스템의 출력을 정하는데 고려하여야 할 점은 다음과 같다. 시스템의 다른 시스템과 연결되는 호환성을 높이기 위해서는 의미를 가지며 중복이 되지 않는 최대 개수의 변수들을 출력으로 내어주는 것이 좋다. 즉 독립성을 가지는 모든 변수의 값을 출력으로 내어 준다. 다음과 같은 예를 들어 보자. 엔진 시스템의 흡입 매니폴드의 공기의 질량의 변화율과 엔진의 회전속도를 출력으로 내어주는 경우 체적 효율 함수 등은 다시 계산되어 질 수 있는 값이다. 그러므로 체적 효율 함수의 값을 출력으로 내주는 것은 출력변수의 중복이다. 그러나 실시간에 가까운 계산속도가 요구되는 경우에는 중복을 허락하여 출력변수를 정할 수 있다. 예를 들면 토크 변환기에서 엔진으로 펌프의 토크 값을 넘겨주게 되는데 이것은 변속기의 속도와 중복되는 경우이다. 왜냐하면 펌프의 토크는 엔진의 회전속도와 변속기의 회전속도로부터 다시 계산되어질 수 있는 값이기 때문이다. 계산속도와 재사용성의 사이의 절충을 이루어 중복을 허락하여야 한다.

시스템 사이의 독립적인 구현을 위해서는 전체 시스템의 수준에서의 고찰로 입출력 변수를 정한다. 임의의 시스템이 다른 시스템 내부의 특수한 변수 값을 입력으로 요구하는 경우에는 다른 시스템이 그 값을 출력으로 내어주어야 하므로 다른 시스템의 출력에 대한 제한 조건이 되어 시스템 사이의 독립성이 적어지게 된다. 또는 요구되는 값을 계산하기 위한 시스템 사이의 변환 시스템이 필요하게 된다.

엔진, 트랜스미션, 드라이브트레인 그리고 제어기로 정해진 전체 시스템의 경계로부터 위와 같은 요구 조건을 만족시키는 서브시스템 단위의 입력과 출력은 표 2와 같다. 표 2에서 볼 수 있듯이 엔진 시스템이 요구하는 입력들 중 트로틀 밸브의 각도, 점화의 선행정도, 부하토크의 값 등은 외부에서 주어지는 순수한 외부의 입력이다. 그러므로 이 입력들의 변화에 따른 시스템의 응답을 알기 위해서는 시간에 따른 시나리오를 정하여 줄 수 있다. 그러나 펌프토크의 값은 변속기의 일부분으로 구현한 토크 변환기의 출력으로 차량 파워 시스템의 서브시스템 사이의 커플링을 보여 주는 예이다.

표 2. 서브시스템의 입출력 변수와 연결 상태.
Table 2. Input and output variables of each subsystems and connection status.

| | 엔 진 | 트랜스미션 | 드라이브트레인 |
|---|---|---|---|
| 입력 변수 | 트로틀 밸브의 각도<br>점화의 선행정도<br>부하토크의 값<br>펌프 토크의 값<br>연료분사량에 대한 명령 | 엔진의 회전속도<br>축의 토크의 합<br>클러치의 압력 | 리액션 캐리어의 회전속도 |
| 출력 변수 | 흡입 매니폴드의 압력<br>엔진의 회전속도<br>산소센서의 출력 | 변속기의 회전속도<br>리액션 캐리어의 회전속도<br>기어의 미끄러짐<br>터어빈의 토크<br>펌프의 토크 | 차량의 직진속도<br>양쪽 축의 토크<br>앞 바퀴의 회전속도<br>뒷 바퀴의 회전속도 |

각각의 단위 시스템을 구성하는 요소들을 살펴보자.엔진 시스템은 독립적인 시스템으로서의 상태변수 계산 이외에 연료분사 제어기를 위한 출력을 내 주어야 한다. 연료분사 제어기의 입력을 위하여 산소센서의 출력과 정해진 각도의 크랭크 축 회전에 대한 펄스를 발생시키는 블록을 포함한다. 이것은 사건 동기 방식의 계산을 위한 신호를 발생시켜 준다. 트랜스미션 시스템은 기어의 현재 위치에 의하여 미분방정식이 선택되는 가변적인 구조의 시스템이다. 이를 위해 주어진 입력을 이용하여 기어의 위치를 구하는 블록을 포함하며 토크 변환기를 서브시스템으로 가지고 있다. 드라이브트레인과 트랜스미션 시스템 역시 주어진 입력에 대하여 출력을 생성하는 독립적인 시스템이므로 상태변수를 계산하는 블록을 기본적으로 내장하고 있다. 제어기의 블록은 크게 연료분사를 위한 부분과 클러치의 유체 압력 제어를 위한 부분을 포함하며, 적당한 입력의 연결만으로 다른 제어기 모듈을 첨가하는 것이 가능하다. 그림 6은 이와 같은 단위 시스템들의 구조와 상호 작용을 나타낸다.

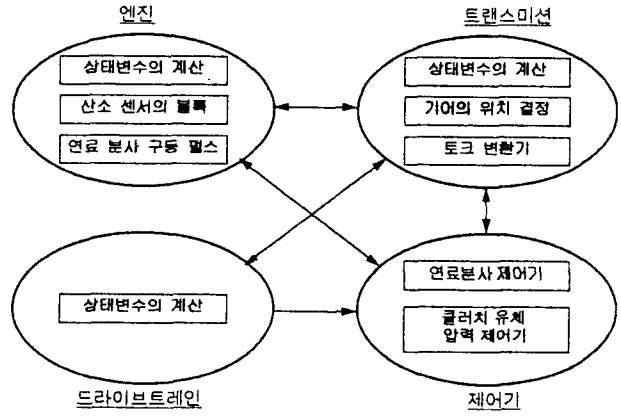



그림 6. 단위 시스템의 구성 블록.
Fig. 6. Component blocks for unit systems.

6. 입력 인자의 정의

엔진 서브시스템은 표 3과 같이 세가지의 입력 인자를 가진다. 엔진 시스템은 세 개의 상태변수를 가지므로 세 개의 상태변수에 대한 초기값이 기본적으로 필요하며 이외에 시간 지연을 구현하기 위한 정보를 필요로 한다. 엔진도시 토크를 계산하기 위해서는 두 가지의 시간지연에 대한 구현이 필요하다. 각각의 시간지연은 흡입으로부터 토크 발생까지의 시간 지연과 점화로부터 토크 발생까지의 시간 지연을

의미하며, 시간지연 큐로 구현된다. 실제적 구현을 위한 입력 인자는 큐의 최대크기와 시간지연의 최대값이다. 표 3에 도시된 것과 같이 트랜스미션과 드라이브트레인 시스템은 상태변수의 초기값이외에 다른 인자를 필요로 하지 않도록 구현하였다.

표 3. 단위 시스템의 입력인자.
Table 3. Input arguments of unit systems.

| | 엔 진 | 트랜스미션 | 드라이브트레인 |
|---|---|---|---|
| 입력 인자 | 상태변수의 초기값<br>지연시간 큐의 크기<br>시간지연의 최대값 | 상태변수의 초기값 | 상태변수의 초기값 |

표 4. 제어기의 입력 인자.
Table 4. Input arguments of controllers.

| | 연료분사 제어기 | 클러치 유압 제어기 |
|---|---|---|
| 입력 인자 | 제어기의 이득값 | 기어변속이 일어나는 속도<br>유압 시스템의 동적 특성<br>및 초기값 |

본 논문에서 구현한 제어기의 블록은 표 4와 같은 입력 인자를 받는다. 연료분사 제어기는 [3]의 가변구조 제어기로 두 개의 이득값을 정하여 주어야 한다. 클러치 제어기의 경우는 유압 시스템의 동적 특성 및 초기값을 필요로 한다.

### 7. 사건 동기 방식의 계산

Simulink는 사건 동기 방식의 ,계산을 지원하지 않으므로, 크랭크 축의 회전과 이에 대한 제어기의 동작을 구현하기 위해서 다음과 같은 방법을 사용하였다. 즉 시스템을 연속 시간으로 구현하고, 하나의 입력을 사건을 나타내는 신호로 설정하여 매번 계산을 수행할 때마다 그 신호를 검사하여 사건이 일어났다고 판단되는 때, 즉 입력 신호가 활성화되었을 때만 계산을 수행하도록 한다. 그 과정은 그림 7과 같으며, Simulink는 가변 시간간격의 계산을 수행하므로 최소 시간간격은 사건의 최소 발생시간 간격보다 작아야만 계산 결과를 신뢰할 수 있다.

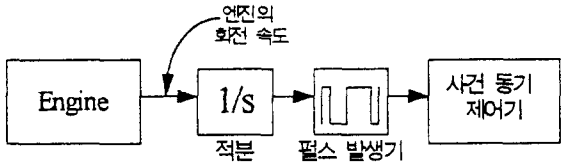



그림 7. 사건 동기 제어기를 구현하는 방법.
Fig. 7. Methods for implementation of the event-driven controller.

## Ⅳ. 시뮬레이션의 결과

그림 8과 같은 시간에 따른 트로틀 밸브의 주어진 시나리오에 대해 그림 9와 그림 10은 엔진의 회전속도, 주행속도를 나타낸 것이다. 그리고 이 모델의 시뮬레이션 결과가 실차실험 결과와 같음은 [4]에서 검증되었다.

여러 외부입력의 시나리오에 대하여 Pentium 60MHz에서 10초의 주행을 시뮬레이션하는데 걸리는 시간은 표 5와 같다. 기어의 변속이 일어나는 경우는 함수의 스위칭으로 인해 변화율이 크므로, 기어의 변속시 계산의 시간간격이 시간간격의 최소값 근처의 작은 값을 가지게 되고 이로 인하여 계산횟수가 많아 진다. 그림 11은 시간간격의 최대값이 0.01이고 최소값이 0.0001인 경우 초기상태와 기어 변속시에 시간간격이 매우 작아지는 것을 보여주고 있다. 하나의 예로 점화의 선행정도나 부하 토크의 변화를 주는 경우도 외란이 시스템에 입력되는 것이므로, 계산이 정상상태에 이르지 못하게 한다. 그러므로 시간간격이 커지지 못하여 계산시간이 많이 소요된다.

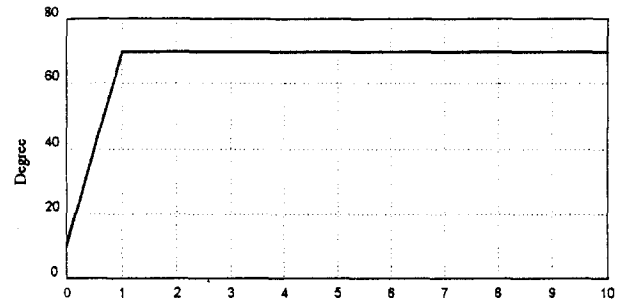



그림 8. 시간에 따른 트로틀 밸브의 시나리오.
Fig. 8. Throttle scenario with respect to time.

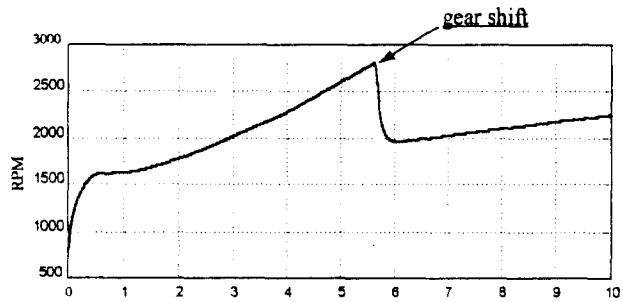



그림 9. 엔진의 회전속도.
Fig. 9. Engine speed.

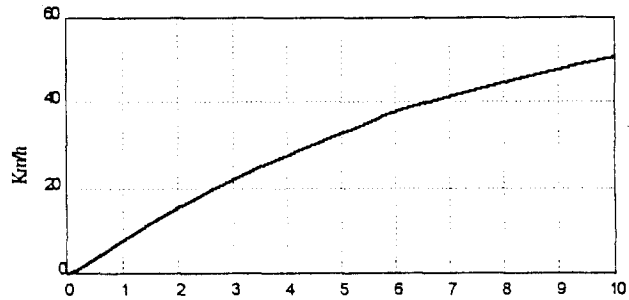



그림 10. 직전속도.
Fig. 10. Longitudinal speed.

표 5. 여러 가지의 시나리오에 따른 계산시간의 비교.
Table 5. Comparison of the computation time in various scenarios.

| 주어진 시나리오 | 계산시간(초) |
|---|---|
| | 가변구조 연료분사 제어기 |
| 고정된 기어 | 24.12 |
| 기어의 변속 | 36.58 |
| 부하 토크의 변화 | 29.71 |

제어기의 모듈이나 시스템의 일부분에 대하여 동력학적인 특성을 바꾼 경우에도 전체 시스템의 새로운 구현이 필요하지 않으며 바뀐 부분의 모듈만을 대체하는 것이 가능하

다. 그림 12는 이와 같은 모듈 단위의 대체를 가능하게 하기 위하여 각각의 서브시스템을 라이브러리화하여 등록한 것을 나타낸다. 그림 상에서는 엔진, 자동변속기(A/T), 드라이브트레인, ECU 및 TCU를 여러 개를 도시하였으나, 현재 각각 1개씩만 구현되었다. 앞으로 여러 개의 다른 시스템의 구현에 관한 연구를 수행할 계획이다.

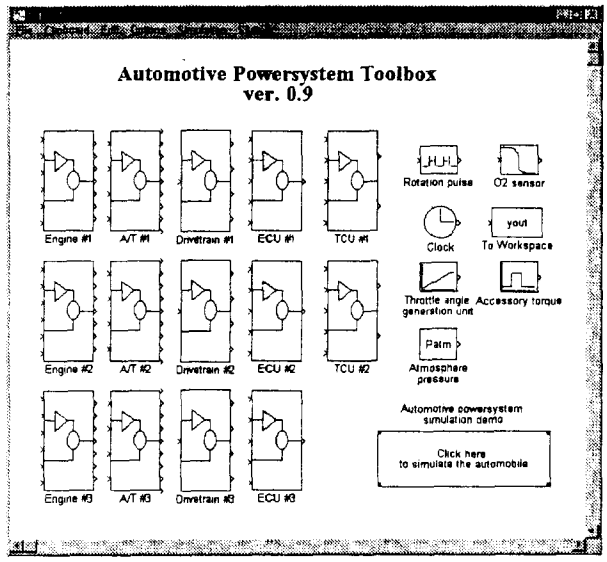



그림 11. 파워 시스템 툴박스.
Fig. 11. Automotive powersystem toolbox.

## V. 결론

본 논문에서는 Simulink의 환경 안에서 C MEX 파일을 작성하여 시스템의 구현시 비선형성이 강하고 계산의 오버헤드가 발생하는 부분에 대하여 계산속도의 증가와 구현의 편리성을 보였다. 구현되어진 시스템의 재사용성을 높이기 위하여 시스템 간의 연결과 수치적분 커널은 Simulink를 사용하였다. 차량 파워 시스템의 시뮬레이션시 여러 가지 제어기의 성능을 검사하기 위하여 전체적인 프로그램을 수정해야 하던 것을 Simulink의 환경안에서 따로 구현되어진 제어기의 라이브러리를 이용하여 간단히 연결함으로써 성능을 비교해볼 수 있었다. 또 플랜트의 동력학적인 특성이 변하는 경우에도 변화된 시스템만을 재구현함으로써 기존의 구현된 부분을 거의 수정없이 사용할 수 있었다. 계산속도 역시 많이 향상되어 기본 블록 라이브러리와 MATLAB 함수를 사용하여 계산할 경우 수십분이 소요되던 시간을 MEX 파일을 사용하여 오브젝트 코드를 수행함으로써 수십초대로 줄였다.

우리는 이러한 환경을 이용하여 여러 제어기의 설계 및 성능 분석, 시스템의 설계, 그리고 제어기의 교체나 시스템의 변화를 쉽게 수용할 수 있으며, 차량 파워 시스템뿐만 아니라 다른 비선형의 시스템에 대해서도 같은 방법을 적용할 수 있다. 그리고 본 논문에서 제시한 방법론은 시각적인 Simulink 환경안에서의 객체지향적인 구현이며 기존의 구현된 시스템에 대해서는 재사용성의 향상을 의미한다. 즉 입력이나 출력의 연결을 재정의하는 수준에서의 완벽한 호환성을 갖추었다고 할 수 있다. 따라서 본 연구에서 개발하고 있는 차량 파워시스템의 객체지향식 시뮬레이션 툴박스가 완성되면 제어기의 설계는 물론 차량의 파워시스템 설계에 유용한 도구가 될 수 있을 것이라고 사료된다.

## 참고문헌

[1] A. G. J. Macfarlane, G. Gruebel and J. Ackermann, "Future design environment for control engineering", *Automatica*, vol. 25, pp. 165 176, 1989.

[2] C. P. Jobling, P. W. Grant, H. A. Barker and P. Townsend, "Object oriented programming in control system design: A survey", *Automatica*, vol. 30, pp. 1221 1261, 1994.

[3] D. Cho and H. Oh, "Variable structure control method for fuel-injected systems", *ASME J. of Dynamic Systems, Measurement and Control*, vol. 115, 1993.

[4] D. Cho and J. K. Hedrick, "Automotive powertrain modeling for control", *ASME J. of Dynamic Systems, Measurement and Control*, vol. 111, pp. 568-576, 1989.

[5] H. A. Barker, M. Chen, P. W. Grant, C. P. Jobling and P. Townsend, "A man-machine interface for computer-aided design and simulation of control systems", *Automatica*, vol. 25, no. 2, pp. 311-316, 1989.

[6] J. J. E. Slotine and W. Li, *Applied Nonlinear Control, Prentice Hall*, 1991.

[7] J. J. Moskwa and J. K. Hedrick, "Automotive engine modeling for real time control application", *Proc. 1987 American Control Conference*, pp. 341 346, June 1987.

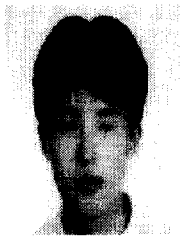
[8] M. J. Denham, "Design issues for CACSD systems", *Proc. IEEE*, vol. 72, no. 12, Dec. 1984.

[9] R. A. Walker, S. C. Shah and N. K. Gupta, "Computer Aided engineering (CAE) for system analysis", *Proc. IEEE*, vol. 72, no. 12, Dec. 1984.

[10] R. W. Weeks and J. J. Moskwa, "Automotive engine modeling for real-time control using MATLAB/SIMULINK", SAE paper 950417, 1995.

[11] The MathWorks, Inc., SIMULINK User's Guide, April 1994.

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes*, Cambridge University press, 1986.



**정 병 용**

1994년 서울대학교 제어계측공학과 학사. 1996년 서울대학교 제어계측공학과 석사. 1996년 ~ 현재 삼성전자 정보통신 연구소 연구원.

**조 동 일**

1980년 Carnegic Mellon 대학 기계공학과 학사. 1984년 M. I. T. 기계공학과 석사. 1987년 M. I. T. 기계공학과 박사. 1987년 9월 ~ 1993년 8월 Princeton 대학 기계·항공·우주공학과 및 Materials Institute 조교수. 1993년 9월 ~ 1995년 12월 서울대학교 제어계측공학과 조교수. 1995년 12월 ~ 현재 전기공학부 조교수. 1990년 ~ 현재 IOP J. of Micromechanics and Microengineering Associate Editor. 1991년 ~ 현재 IEEE/ASME J. of Microelectromechanical Systems Associate Editor.