Transactions of KSAE, Vol. 11, No. 5, pp.147-155 (2003)

Copyright © 2003 KSAE
1225-6382/2003/065-21

객체지향 동력전달계 동적 시뮬레이션 프로그램 개발 연구

한 형 석[*1] · 이 재 경[1] · 김 현 수[2] · 임 원 식[3]

한국기계연구원[*1] · 성균관대학교[2] · 서울산업대학교[3]

Development of the Object-oriented Powertrains Dynamic Simulation Program

Hyungsuk Han[*1] · Jaikyung Lee[1] · Hyunsoo Kim[2] · Wonsik Lim[3]

[1] KIMM, 171 Jang-dong Yousonggu, Daejeon 305-343, Korea
[2] School of Mechanical Engineering, Sungkyunkwan University, Suwon 440-746, Korea
[3] Department of Automotive Engineering, Seoul National University of Technology, Seoul 139-743, Korea
(Received 17 April 2003 / Accepted 31 July 2003)

Abstract : The application of object-oriented modeling to develop a powertrain performance simulation program, called O-DYN, is introduced. Powertrain components, such as the engine, transmission, shaft, clutch are modeled as classes which have data and method by using object-oriented modeling methodology. O-DYN, a performance simulation program, based on the object-oriented modeling is made in C++. One powertrain simulation using O-DYN is proposed, and it is expected that the simulation program or individual class constructed in this paper would be useful for automotive engineers to predict the performance of powertrains and to develop a simulation program.

Key words : Powertrain(동력전달계), Object-oriented modeling(객체지향 모델), Class(클래스), Dynamic simulation(동적 시뮬레이션), O-DYN

Nomenclature

x	: input variables
y	: output variables
z	: states
$\dot{z}$	: state derivatives
f, g, F	: functions
Ψ	: Jacobian

1. 서 론

자동차 동력전달계의 고부가가치화를 위해서는 품질향상과 개발기간 단축이 필요하다. 이를 달성하기 위한 방법의 하나로 개념설계 단계부터 전 수명에 걸쳐서 가상공학의 적용이 요구된다. 가상공학의 하나인 컴퓨터 시뮬레이션에 의한 동력전달계의 동력성능, 연비성능, 진동특성, 변속 특성 등을 미리 예측함으로써 실물 제작비용을 줄이고 개발기간도 단축할 수 있다. 그런데 동력 전달계의 구조 특성상 시뮬레이션에 의한 성능 예측이 타 분야에 비하여 아직 어려운 면이 있다. 동력 전달계는 구성하는 요소들 간의 연성이 많고, 동일한 요소에 대하여 동적 해석 모델이 다양하며, 제어계가 포함되어 있는 특성이 있다. 타 분야에서는 시뮬레이션 프로그램이 활발히 적용되는 것에 비하면 동력전달계 동적 성능 시뮬레이션 분야는 적용 속도가 느리다고 사료된다. 그러한 특성으로 인하여 동력 전달계 제조사마다 자체 성능 시뮬레이션 프로그램을 개발하

*To whom correspondence should be addressed.
hshan@kimm.re.kr

여 사용하는 것이 때로는 요구된다. 그러므로 동력 전달계의 동적 성능 시뮬레이션을 일반화 하기 위해서는 시뮬레이션 프로그램이 사용자 중심적이어야 한다. 사용자가 사용자의 목적에 맞춰 사용자화 할 수 있도록 확장성, 개방성, 수정성, 유지보수성, 하드웨어와의 인터페이스성, 재 사용성 등에서 편리해야 한다.

현재 상용 또는 자체 개발한 시뮬레이션 프로그램은 주로 절차적 소프트웨어 개발 방법을 사용하였기 때문에 상기에서 언급한 해석기의 요구 특성을 만족하는데 어려움이 있다. 절차적 모델링 기법으로 개발된 시뮬레이션 프로그램은 수정이 어렵고 해석 모델의 재 사용성도 떨어진다. 새로운 해석법 또는 새로운 요소의 추가가 어려워 사용자의 목적에 따라 유연하게 대처하기 어렵다고 할 수 있다. 그러한 한계를 극복하기 위한 기법이 객체지향 모델링 기법으로 건축, 조선, FEM 분야에서는 이미 활발히 적용되고 있다.[1-3]

국내에서 동력전달계의 성능 시뮬레이션을 객체지향 모델링 기법을 이용하여 수행된 적이 있다.[4-6] 그러나 범용이라기 보다는 특정한 대상에 중점을 두고 있다고 사료되며 MATLAB /SIMULINK와 같은 상용 도구에 기반을 두고 있는 것이 특징이다. 외국의 다물체 동역학 시스템 시뮬레이션 프로그램 개발에 객체지향 데이터 모델을 이용한 사례가 있다.[7-9]

본 논문에서는 국책연구과제 "웹기반 범용 동력전달계 성능해석 시스템 개발"에서 개발 중인 동력전달계 성능 시뮬레이션을 위한 해석기 개발 개념을 주로 소개하는데 목적을 두고 있다. 본 논문에서 소개하는 해석기는 범용이며 사용자 중심적이 되도록 하기 위하여 객체지향 모델링 기법을 적용하여 개발된다. 동력전달계를 구성하는 요소나 부 시스템을 클래스로 모델링하고 모듈화 한다. 시뮬레이션 프로그램은 객체지향 언어 C++를 통하여 구현하고 단순 동력 전달계에 적용한다. 연구를 통하여 본 논문에서 소개하는 시뮬레이션 프로그램은 개발 효율성, 향후 확장성, 재 사용성, 유연성에 있어서 장점이 있음을 확인할 수 있었다.

2. 객체지향 모델링 개요

소프트웨어 공학에서 발전한 객체지향 모델링 및 프로그래밍 기법은 소프트웨어의 개발 방법으로 이미 실용화되었다. 기존의 절차적 모델링 기법은 문제를 분석하고 프로그램을 구현하는 과정이 기능(함수) 지향적이다. 여기서 데이터는 부수적인 요소로 취급된다. 반면에 객체지향 기법에서는 실세계의 사물을 소프트웨어 영역으로 사상 시킨 객체(object)를 중심으로 구현된다. Fig. 1에서와 같이 전통적인 절차적 모델링 기법에서는 데이터와 절차가 구분된 후 절차의 필요에 따라 데이터가 사용된다. 반면에 객체 지향에서는 데이터와 기능을 캡슐화 시킨 객체를 중심으로 원하는 기능을 수행하게 된다. 객체는 클래스로 추상화되고 클래스는 객체의 특성과 기능을 캡슐화한다. 그러므로 객체지향 모델링은 존재하는 사물을 데이터와 기능을 통합한 객체로 정의하는 것이라 할 수 있다.

객체지향 모델의 기본 개념에 이용되는 것은 객체, 클래스, 멤버(데이터), 멤버함수(기능), 메시지이다. 객체는 객체 기술에 필요한 데이터 즉 맴버와 객체의 기능인 멤버함수로 구성되고 객체는 독립적이면서 타 객체와는 메시지를 통하여 인터페이스를 하게 된다. 많은 객체들은 상호 동일하거나 유사한 데이터 구조와 기능을 가지는데 이러한 객체들은 클래스를 통하여 생성된다. 결과적으로 소프트웨어 모듈의 부품화를 가능하게 하고 개발 효율 및 신뢰성을 높여준다. 또한 유지보수, 확장성, 재 사용성이

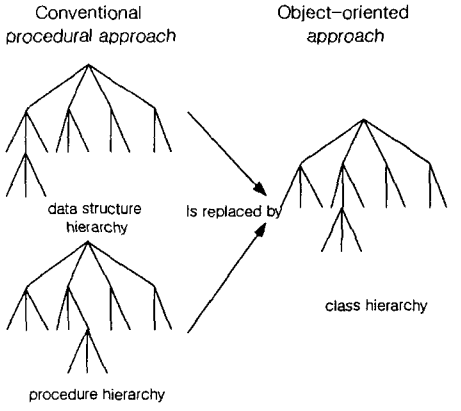

Fig. 1 An object-oriented approach with one unified hierarchy

탁월하다.

클래스는 상속성이 있어 상위 클래스를 이용하여 하위 클래스를 쉽게 파생시킬 수 있다. 또한 다형성이 있어서 한 함수가 상황에 따라 다른 기능을 수행한다. 결과적으로 현실 세계의 사물을 소프트웨어적으로 시뮬레이션 하는 프로그램의 개발에 있어서 객체지향 기법을 이용하는 것이 타당하다고 할 수 있다. 객체지향 프로그래밍 언어 C++는 객체지향의 개념을 구현하도록 개발된 것이며 C의 문법을 포함하고 있어 기존의 C로 만들어진 자원을 이용할 수 있는 장점을 가진다.

3. 동력전달계 모델링

3.1 요소의 클래스화

동력 전달계는 엔진, 클러치, 변속기, 축, 기어와 같은 요소로 구성된다. 이러한 요소들로 구성된 동력 전달계의 시스템적인 동적 시뮬레이션을 위한 프로그램을 개발하기 위하여 본 연구에서는 객체와 클래스를 도입한다. 클래스는 동종의 객체를 정의하기 위한 템플릿이라고 할 수 있다. 전술한 바와 같이 클래스는 속성, 즉 데이터와 기능을 동시에 갖고 있다.

동력 전달계를 구성하는 각 요소는 Fig. 2와 같이 각 요소의 속성과 기능을 캡슐화 한 클래스로 정의된다. 이러한 캡슐화는 각 객체가 독립적으로 존재하게 되어 객체의 생성, 확장이 시뮬레이션 프로그램의 전체적인 구조에 대한 지식이 없이도 가능하게 된다. 동력전달계를 구성하는 요소들을 클래스화 하기 위하여 본 논문에서는 동력 전달계 요소들을 3 종류로 구분하였다. Fig. 3에서와 같이 대수식 형태, 미분방정식 형태, 미분-구속방정식 형태이며 미분-구속방정식 형태는 미분방정식과 미분변수(상태변수)의 구속식이 존재하는 형태이다. 각 형태에 대한 상세한 특성 설명과 예제 클래스는 3.2절에서 소개한다.

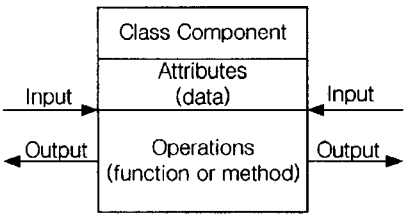

Fig. 2 Encapsulation of powertrain components

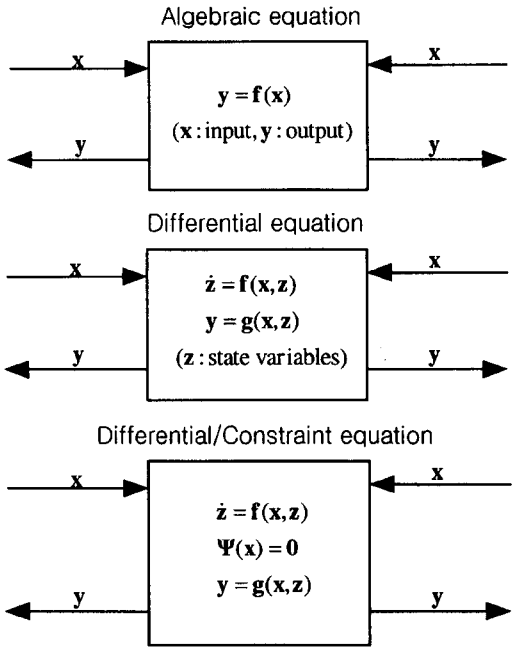

Fig. 3 Base class types for powertrain components

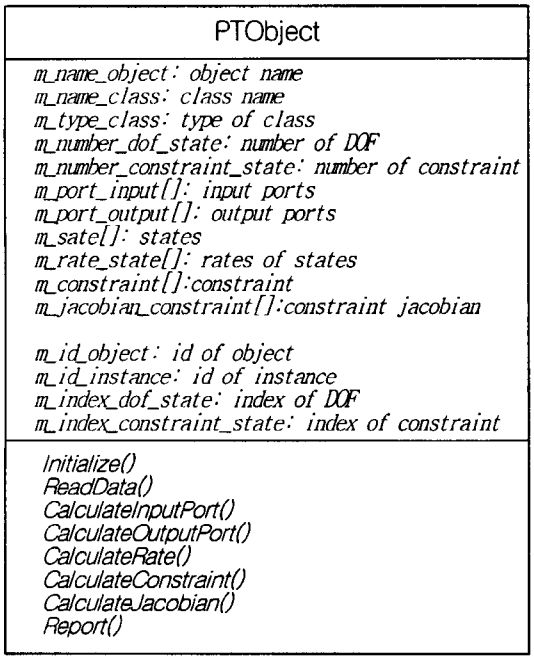

Fig. 4 Superclass for powertrain classes

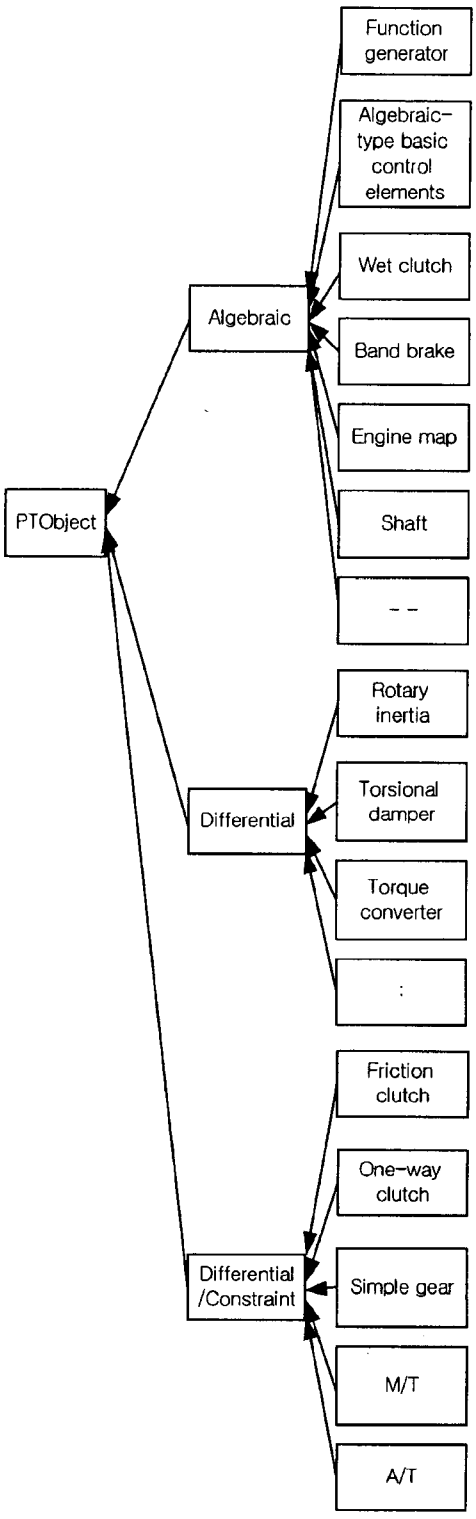

Fig. 5 Multilevel inheritance hierarchy

동력 전달계 요소를 클래스화하기 위하여 우선 최상위 클래스 PTObject가 Fig. 4에서와 같이 정의된다. PTObject는 모든 클래스의 공통 속성과 기능을 포함하게 된다. 속성으로는 객체명, 클래스 형식, 자유도 수, 구속식 수, 객체번호, 인스턴스 번호, 자유도 번호, 구속식 번호, 입력 포트 값, 출력 포트 값, 상태 값, 상태 속도 등으로 구성된다. 함수(기능)로는 입출력 포트 계산, 상태 속도 계산, 데이터 읽기, 초기화, 결과 출력 기능으로 구성된다. 이러한 속성과 기능은 하위 클래스에 공통적이기 때문에 새로 파생되는 클래스는 PTObject 클래스에서 정의한 속성과 기능의 재 정의 없이 단지 객체지향 언어의 오버라이딩(overriding) 기법을 이용하여 각 클래스에 적합시킨다.

Fig. 5는 Fig. 4의 최상위 클래스를 이용하여 파생시킨 클래스 계층도이다. Fig. 5에서와 같이 최상위 클래스 PTObject에서 Algebraic, Differential, Differential/Constraint의 부 클래스가 파생되고, 부 클래스에서 다시 동력 전달계를 구성하는 요소들의 클래스가 파생되는 것이다. 클래스의 파생에는 객체지향의 특징인 상속성, 다형성이 이용되어 쉽게 새로운 클래스를 정의할 수 있다.

3.2 클래스화 예

Fig. 6, 7은 대수식 형태의 클래스를 예로 들기 위한 엔진 토크 성능 정의 클래스를 보여주고 있다. 엔진의 출력 토크를 정의하는데 있어서 간단한 방법의 하나로 드로틀과 엔진의 회전속도에 대한 토크 성능 곡선을 이용하는 것이다. 실험에 의하여 얻어진 성능 곡선을 보간하여 이용하게 된다. 이러한 요소는 대수식 형태로 정의할 수 있는데, 이러한 형태를 정의하기 Fig. 7과 같은 구조의 클래스가 정의된다. Fig. 7에서와 같이 대수식 형태의 클래스는 입출력 값을CalculateInputPort(), CalculateOutputPort() 함수를 이용하여 정의된다.

Fig. 8과 Fig. 9는 미분방정식 형태의 클래스의 예로 회전 관성을 보여주고 있다. 회전 관성은 입력으로 토크를 받아들이고 출력은 회전 관성의 회전각, 회전 각속도이다. 이 요소에서 회전 관성의 회전각, 각속도는 상태변수로 정의되고 미분방정식의 적분

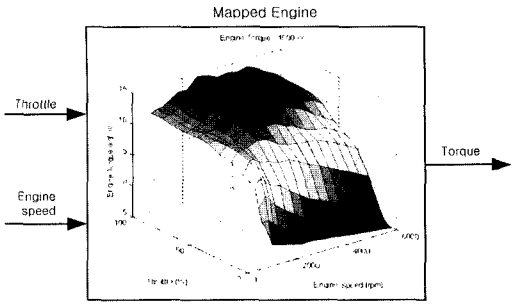

Fig. 6 Mapped engine model

Class MappedEngine
m_number_port_input = 2 *m_number_port_output = 1*
CalculateInputPort(){ *m_port_input[0] = Throttle;* *m_port_input[1] = Engine Speed;* *}* *CalculateOutputPort(){* *m_port_output[0] = F(m_port_input[0],* *m_port_input[1]);* *}*

Fig. 7 Class for the mapped engine

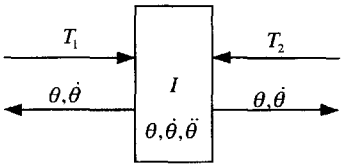

Fig. 8 Rotary Inertia

Class RotaryInertia
m_number_dof_state = 2 *m_number_port_input = 2* *m_number_port_output = 2* *m_I*
CalculateInputPort(){ *m_port_input[0] = T_1;* *m_port_input[1] = T_2;* *}* *CalculateOutputPort(){* *m_port_output[0] = θ;* *m_port_output[1] = $\dot{\theta}$;* *}* *CalculateRate(){* *m_rate_state[0] = $\dot{\theta}_1$;* *m_rate_state[1] = $\frac{T_1+T_2}{I}$;* *}*

Fig. 9 Class for the rotary inertia

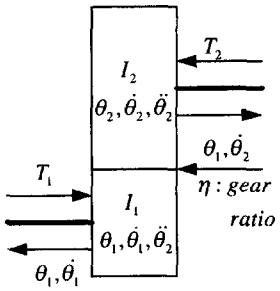

Fig. 10 Simple gear

Class SimpleGear
m_number_dof_state = 4 *m_number_constraint = 1* *m_I_1,m_I_2,m_η*
CalculateInputPort(){ *m_port_input[0] = T_1;* *m_port_input[1] = T_2;* *}* *CalculateOutputPort(){* *m_port_output[0] = θ_1;* *m_port_output[1] = $\dot{\theta}_1$;* *m_port_output[2] = θ_2;* *m_port_output[3] = $\dot{\theta}_2$;* *}* *CalculateRate(){* *m_rate_state[0] = m_state[1];* *m_rate_state[1] = T_1 / I_1;* *m_rate_state[2] = m_state[3];* *m_rate_state[3] = T_2 / I_2;* *}* *CalculateJacobian(){* *m_jacobian_constraint[0][0] = 0;* *m_jacobian_constraint[0][1] = 1;* *m_jacobain_constraint[0][2] = 0;* *m_jacobain_constraint[0][3] = η;* *}* *CalculateConstraint(){* *m_constraint[0] = m_state[1] + m_state[2] * m_η;*

Fig. 11 Class for the simple gear

에 의하여 결정되게 되며, CalculateRate() 함수를 이용하여 상태변수에 대한 1차 상미분 방정식을 정의하게 된다. 이 상미분 방정식은 수치적분에 의하여 해를 구하게 된다. Fig. 10과 Fig. 11은 미분/구속방정식 형태의 클래스 예로 단순 기어를 보여주고 있다. 기어는 미분방정식에 추가하여 식 (1)과 같은 구

속조건과 식 (2)의 자코비안으로 표현된다. 이러한 기어는 Fig. 11과 같이 정의된다.

$$\Psi(\theta_i, \dot{\theta}_i, \theta_j, \dot{\theta}_j) = \dot{\theta}_i + \eta\dot{\theta}_j = 0 \quad (1)$$

$$\Psi_\theta = [0\ 1\ 0\ \eta] \quad (2)$$

이러한 형태의 클래스는 구속식과 자코비안을 정의하기 위하여 CalculateJacobian() 함수와 CalculateConstraint()가 이용된다.

3.3 프로그램 개발

전 절에서 제안한 방법에 의하여 모델링 되는 동력 전달계의 지배방정식은 식 (3)과 같이 표현된다.

$$\begin{bmatrix} I & \Psi^T \\ \Psi & 0 \end{bmatrix} \begin{bmatrix} \dot{z} \\ \lambda \end{bmatrix} = \begin{bmatrix} F(z,u,t) \\ 0 \end{bmatrix} \quad (3)$$

식 (3)의 해는 상미분방정식(ODE) 해법과 선형해법을 이용하여 구할 수 있다. 본 연구에서의 식 (3)의 해를 구하기 위한 수치해석 구조는 Fig. 12와 같다. 수치해석을 수행하기 위해 별도의 클래스 Dynamic을 이용한다. Dynamic 클래스는 Fig. 12에서와 같이 수치적분기와 선형 해석기를 함수로 포함하고 있으며 시스템 지배방정식을 구성하고 해를 구하는 과정에서 필요한 정보를 각 객체와의 메시지 전달을 통하여 주고 받는다. 이러한 구조의 특징은 공통 메모리를 사용하지 않는다는 것이다. 즉 데이터를 각 객체가 포함하고 있으면서 필요에 따라 객체간의 메시지 교신에 의하여 데이터 전달이 이루어진다. 이에 대한 설명을 Fig. 13, Fig. 14에서 보여주고 있다. Fig. 13은 전통적인 기법으로 공통 메모리와 기능 중심으로 구성되어 있다. 그러므로 새로운 요소를 추가하려면 공통 메모리에 대한 지식이 요구된다. 반면에 Fig. 14는 본 논문에서 적용하는 객체지향형으로 객체는 데이터와 기능을 포함하고 있고 수치 해석기와 메시지 교환에 의하여 데이터를 송수신한다. 그러므로 사용자가 새로운 클래스의 개발을 원할 때 해석 프로그램의 전반적 구조에 대한 지식이 없이도 쉽게 정의할 수 있게 된다. 또한 새로운 수치해석 과정을 도입할 때도 객체의 내부에 대한 지식이 필요하지 않다. 단지 필요 데이터만 교신하면 되는 것이다. 결과적으로 확장성, 유연성, 재사용성 등에서 기존의 절차적 방법에 의하여 개발된 해석기보다 장점을 갖는다.

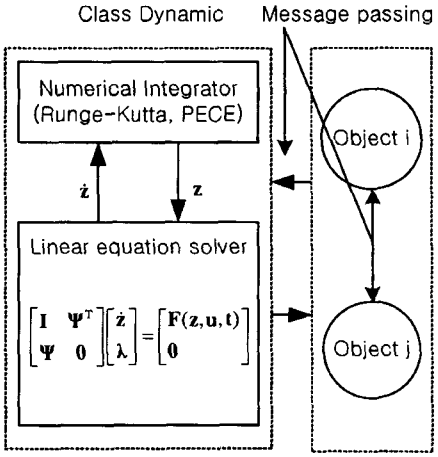

Fig. 12 Structure of O-DYN numerical analysis

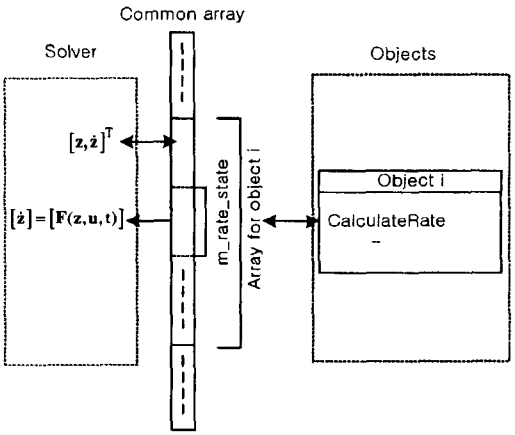

Fig. 13 Conventional data model for powertrain dynamic analysis

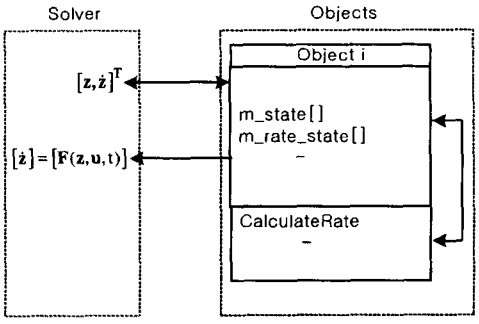

Fig. 14 Object-oriented data model for powertrain dynamic analysis

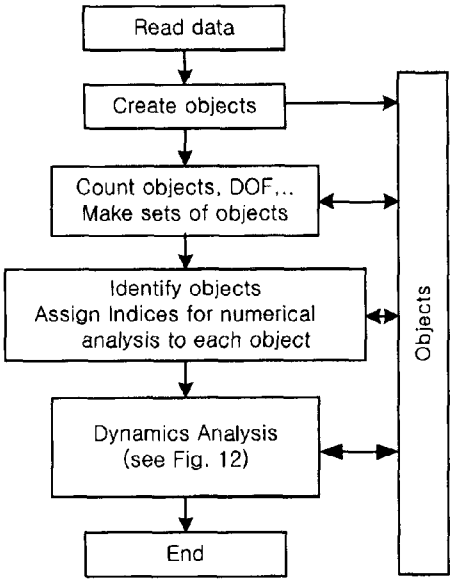

Fig. 15 O-DYN powertrain dynamic analysis flow

본 논문에서는 Fig. 12와 같은 수치해석 과정을 기반으로 하여 Fig. 15와 같은 흐름을 갖는 해석기 O-DYN을 객체지향 언어 C++를 이용하여 개발하였다. 전통적인 흐름과는 다르게 객체의 생성과 관리가 포함되어 있다. O-DYN의 전후 처리기는 웹기반으로 개발되었으며 이에 대하여는 다른 논문에서 상세히 소개할 것이다.

4. 적용

전 절에서 개발한 동력전달계 성능 시뮬레이션 프로그램 O-DYN을 이용하여 Fig. 16과 같은 단순 동력전달계에 적용하여 발진 성능 시뮬레이션을 수행하였다.

Fig. 16 차량에 대한 O-DYN 해석 모델을 Fig. 17에서 보여주고 있다. 본 해석 차량에 이용된 주요 데이터는 Table 1과 같다. Fig. 17은 요소간의 연결과 각 요소의 모델에 이용된 클래스를 보여주고 있는데, 동력은 엔진→토크 컨버터→기어→축→차량으로 전달된다. 엔진은 실험에 의하여 얻어진 성능 곡선을 이용하는 Mapped Engine 클래스를 이용한다. 이 클래스는 Fig. 6과 Fig. 7에서와 같이 정의된다. 본 해석에서는 발진 성능을 해석하기 위하여 드로틀을 개방한 것으로 가정하고 엔진의 회전 속도를 이용하여 엔진 출력 토크를 계산하게 된다. 토크 컨버터는 부드러운 발진, 토크 증대, 유체 커플링 기능을 수행하며 대표적 비선형 요소이다.

본 논문에서는 Torque Converter 클래스를 정의하였는데 펌프와 터빈의 속도비에 기초한 토크 컨버터의 성능에 대한 다항식을 이용하여 터빈 출력 토크를 계산한다. 변속기를 나타내는 기어는 1단 기어로 가정하고 변속기 입력축으로부터 차동 기어까지의 기어비를 적용한다. 변속기와 차체 관성 사이의 동력전달 요소는 단지 강성을 갖는 축으로 모델한다. O-DYN의 해석 결과를 검증하기 위해 각 요소를 O-DYN과 동일한 해석 모델을 이용하는 전용 해석 프로그램을 개발하여 시뮬레이션을 수행하였다. 그러나 전용 해석 프로그램은 일정 적분간격 4차 Runge-Kutta 적분기를 사용하기 때문에 수치적분의 안정성을 위하여 해석 모델에서의 축(Simple Shaft)은 강체로 가정하였다. O-DYN은 가변 적분간격 적분기 DE를 이용한다.[10] 그러므로 O-DYN과 전용 프로그램과의 결과에 있어서 강성 고려에 따른 차이가 있을 것으로 예상된다. 주요 입력 데이터는 Table 1과 같다.

Fig. 18은 해석 결과 중 토크 컨버터의 펌프와 터빈의 각속도를 보여주고 있다. 펌프는 초기속도 100rad/sec에서부터 계속 증가하며 터빈의 속도가 펌프의 속도에 수렴하는 것을 볼 수 있다. 검증용으로 개발한 해석 프로그램과의 결과 차이가 강성의 고려 유무로 인하여 초기에만 미미하게 차이가 나는 것을 알 수 있다. Fig. 19는 차량의 속도를 보여주고 있다. Fig. 19를 보면, 1단 기어 연결조건에서 차량은 약 5초 후에 60 km/h에 도달하는 발진 성능을 갖는 것을 알 수 있다. 차속에 있어서도 O-DYN과 전용 해석 프로그램의 결과가 유사하여 O-DYN의 전체적인 수치해석 흐름과 해의 정확도를 간접적으로 검증할 수 있다.

Table 1 Input parameters

Pump inertia	0.16kg-m^2
Turbine inertia	0.1kg-m^2
Vehicle inertia	102.3kg-m^2
Gear ratio	10.926
Stall torque ratio	2.36
Coupling point	0.86

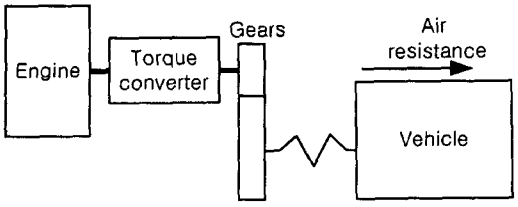

Fig. 16 Simple vehicle model

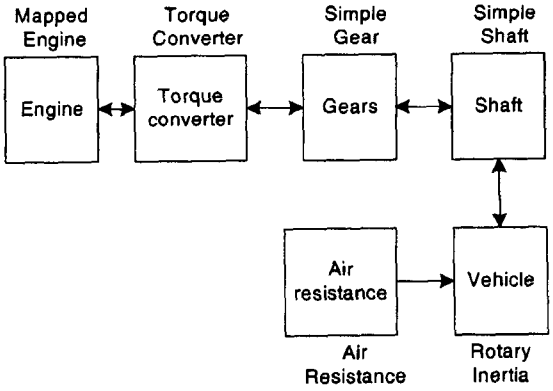

Fig. 17 O-DYN model with connected components

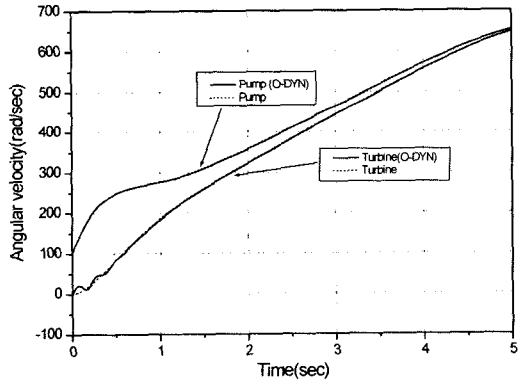

Fig. 18 Velocity of pump and turbine

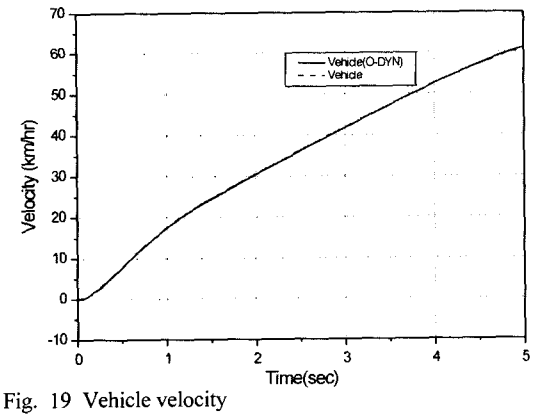

Fig. 19 Vehicle velocity

5. 결 론

본 논문에서는 객체지향 모델링 기법을 적용하여 동력 전달계 구성요소를 클래스로 모델하고 성능해석기(O-DYN)를 개발하였다. 설계된 해석기는 C++를 이용하여 구현하고 예제 동력 전달계에 적용하여 성능 시뮬레이션을 수행하였다. 이를 통하여 본 논문에서 제시하는 기법이 다음과 같은 장점이 있는 것을 확인하였다.

첫째로는 동력전달계 구성 요소를 클래스로 정의하여 독립시켰기 때문에 요소의 수정성, 교환성, 분산 개발성, 재 사용성을 증가시켰다. 실 예로 분산개발을 실시하고 개발된 클래스를 등록하는 작업이 전체 성능 해석기의 구조에 대한 이해가 없이도 쉽게 수행할 수 있음을 경험하였다.

둘째로는 새로운 해법의 적용이나 타 S/W, H/W와의 인터페이스가 용이할 것으로 기대된다. 이는 새로운 해법을 적용하기 위하여 주 해석기와 라이브러리를 구성하는 요소들의 수정이 필요 없이 필요에 따라 각 객체에게 메시지를 통하여 정보를 주고 받을 수 있기 때문이다.

셋째로는 새로운 요소의 정의를 상속성을 이용하여 쉽게 정의할 수 있다는 것이다. 결과적으로 물리적 시스템의 컴퓨터 시뮬레이션을 위한 소프트웨어 개발 시 객체지향 모델링 기법을 이용하는 것이 합리적이라 사료된다. 또한 동력 전달계의 성능 시뮬레이션에 있어서 요구되는 사용자 중심의 해석기를 개발할 수 있기 때문에 동력 전달계 개발자가 목적에 따라 쉽게 사용자화 하여 성능을 해석할 수 있을 것으로 기대된다. 본 논문에서 소개한 O-DYN은 향후 다양한 동력 전달계 구성 요소들의 클래스화를 통한 라이브러리를 구축할 계획이다.

후 기

본 연구는 과학기술부에서 지원하는 주력산업의 고부가가치화 사업 "웹기반 범용 동력전달계 성능해석 시스템 개발" 과제의 연구비로 수행되었음을 밝히며 본 연구의 지원에 대하여 감사드립니다.

References

1) J. H. Kim, C. H. Cho, S. J. Kim "Distributed Parallel Computing of Finite Element Analysis on a Network using Object Oriented Programming Paradigm," Journal of KSAS, Vol.23, No.1, pp.97-106, 1995.
2) Y. S. Shin, J. K. Suh, H. W. Choi, Y. S. Park, "Development of Object-Oriented Structural Analysis Program for PC," Computational Structural Engineering, Vol.5, No.4, pp.125-132, 1992.
3) H. K. Kim, J. Y. Lee, J. J. Kim, B. H. Lee, "Application of Object-Oriented Methodology for Structural Analysis and Design," Computational Structural Engineering, Vol.8, No.3, pp.123-133, 1995.
4) P. J. Yoon, M. H. Sunwoo, S. J. Lee, "Study on an Engine Control System using an Object Oriented Programming Method," Transactions of KSAE, Vol.8, No.3, pp.98-109, 1995.
5) K. J. Yang, K. S. Hong, K. I. Lee, "An Object -Oriented Model for Gasoline Engine and Automatic Transmission Systems," Journal of Control, Automation and Systems Engineering, Vol.4, No.4, pp.534-542, 1998.
6) B. Y. Jeong, D. I. Cho, "Object-Oriented Programming and Automotive Powersystem," Journal of Control, Automation and Systems Engineering, Vol.2, No.2, pp.127-133, 1996.
7) M. Otter, H. Elmqvist, F. E. Cbllier, "Modeling of Multibody Systems with the Object-Oriented Modeling Language Dymola," Nonlinear Dynamics 9, pp.91-112, 1996.
8) A. Kecskemethy, M. Hiller, "An Object-oriented Approach for an Effective Formulation of Multibody Dynamics," Comput. Methods Appl. Mech. Engrg.115, pp.287-314, 1994.
9) A. Kecskemethy, C. Lange, G. Grabner, "Object-oriented Modeling of Multibody Dynamics Including Impacts," ECCM-2001, pp.1-28, 2001.
10) L. F. Shampine, M. K. Gordon, Computer Solution of Ordinary Differential Equations, Freeman, San Francisco, 1975.