요소 독립성이 유지되는 반복해법에 의한 객체지향 유한요소모델 개발

Development of an Object-oriented Finite Element Model through Iterative Method Ensuring Independency of Elements

이한귀 · 김태곤 · 이정재

Lee, Hanki · Kim, Taegon · Lee, Jeong Jae

ABSTRACT

Application of the Object-oriented Programming (OOP) method to the Finite Element Model (FEM) program has various strengths including the features of encapsulation, polymorphism and inheritance. However, this technique should be based upon a premise that the independency of the object method and data to be used is guaranteed. By attempting to apply the OOP to the FEM, existing researches go against the independency of the OOP which is an essential feature of the method. The reason is this: existing researches apply the OOP to modules in accordance with analysis procedures, although the data to be used is classified as an element unit in the FEM. Therefore, the required independency cannot be maintained as whole stiffness matrices and boundary conditions are combined together. Also, solutions are sought from analysis module after data is regrouped at the pre-processor, and their results are analyzed during the post-processor. As this is similar to a batch processing, it cannot use data at analysis, and recalculation should be done from the beginning if any condition is changed after the analysis is complete, which are limitations of the existing researches. This research implemented the Object-orientation of elements so that the three features of the OOP (i.e. encapsulation, polymorphism and inheritance) can be guaranteed and their independency maintained as a result. For this purpose, a model called 'Object-oriented Finite element Model ensuring the Independency of Elements (OFMIE)', which enables the analysis of targets through mutual data exchanges within instance, was developed. In conclusion, the required independency was achieved in the instance of the objected elements and the analysis results of previous conditions could be used for the analysis after changes. The number of repetitive calculations was reduced by 75 per cent through this gradual analysis processes.

Keywords: OOP; FEM; independency

1. 서 론

유한요소법 (Finite Elements Method, FEM)은 많은 계산이 필요하므로 컴퓨터의 이용을 전제로 한다 (Logan, 2011). 컴퓨터 하드웨어와 소프트웨어가 발달함에 따라 FEM 프로그램의 구현 방법도 많은 변화를 겪어 왔다. CPU의 처리속도가 느리고 메모리 용량이 작았던 초기에는 해석 속도를 개선하기 위한 알고리즘이나 메모리의 한계를 극복하기 위한 Frontal 기법, Band Matrix, Skyline Algorithm 등이 고안되어 프로그램에 적용되었다 (Irons, 1970; Sloan, 1986; Kumar, 2010). 이러한 프로그램

들은 FORTRAN IV 언어를 이용한 배치처리 방식을 중심으로 구현되었지만 함수의 구현이 용이한 f77이 발달하면서 함수형 프로그램으로 발전하였고, C언어가 보편화 되면서 문제의 크기에 따라 메모리 조정이 가능한 동적 메모리 사용방식으로 개선되었다.

최근 컴퓨터의 성능이 개선되고, 컴퓨터가 보편화 되면서 프로그램의 개발자와 이용자가 분리되었고, 해석 속도나 메모리 사용의 최적화 보다는 유지관리와 일관성 및 확장성이 프로그래밍 기법의 중요한 목표가 되었다. 즉, 데이터의 처리에 알맞게 처리 모듈이 모든 메소드와 데이터를 공유하는 절차지향 프로그래밍 (Procedure-oriented Programming, POP)에서 메소드와 데이터를 중심으로 처리과정을 종속시키는 객체지향 프로그래밍 (Object-oriented Programming, OOP)으로 발전한 것이다 (Satzinger, 1996; Heng et al, 2009).

OOP는 캡슐화, 다형성, 상속성의 특징에 의해 다양한 장점을 갖는데, 이를 위해서는 객체의 단위가 되는 메소드 및 데이터의

* 서울대학교 생태조경·지역시스템공학부

** 서울대학교 조경·지역시스템공학부, 서울대학교 농업생명과학연구원

† Corresponding author Tel.: +81-2-880-4589

Fax: +81-2-873-2087

E-mail: lj@snua.ac.kr

2011년 11월 21일 투고

2012년 3월 8일 심사완료

2012년 3월 13일 게재확정

독립성이 전제되어야 한다 (Fenves, 1990; Rebecca, 1990). 객체는 프로그램에서 구현되어 변수처럼 사용 하는데 이를 인스턴스라고 한다. OOP에서는 이 인스턴스의 독립성이 보장 되어야 적절하게 목적을 달성 하였다고 할 수 있다.

FEM에서도 1990년 Forde를 시점으로 OOP 기법을 적용하기 시작하였다 (Forde et al, 1990). 유용성이 검증된 1990년대 중반에는 SmallTalk (Zimmermann et al, 1992)나 C++ (Kong and Chen, 1994)등 다양한 OOP 언어를 이용하여 FEM 프로그램이 개발되었고, 그래픽을 이용하여 사용자의 직관적 이해를 높이거나 객체의 독립성을 높이기 위해 Map객체를 이용하는 모델의 개발이 이어졌다 (Mackie, 1997; Archer et al., 1999). 이후 OpenSees, OOFEM과 같은 프로젝트 단위의 OOP기반 프레임워크를 비롯하여 POP와 OOP 언어를 혼합한 최적화된 FEM 프레임워크 개발에 대한 연구가 진행되었다 (Cho, 2009).

FEM에서 데이터는 요소단위로 가공되고 처리되므로, 요소단위로 인스턴스화 되어야 하며 독립성이 전제되어야 한다. 그러나 기존 연구들에서는 과거 POP 기반의 해석 알고리즘을 그대로 사용하기 위하여 해석 절차에 따라 나뉘는 모듈에만 OOP 기법을 적용 하고 있다. 이러한 이유로 해석단계에서 전체 강성행렬 및 경계조건의 데이터를 각각의 요소로부터 참조하여 조합하기 때문에 독립성을 유지할 수 없다. 즉, POP에서 사용 되었던 것처럼 외부에서 객체 내부의 메소드와 데이터를 직접 조작하기 때문에 독립성을 해치므로 OOP에서 지향하는 캡슐화, 다형성, 상속성을 달성할 수 없다 (Nie et al, 2010). 따라서 전처리기에서 데이터를 구축한 후, 해석모듈에서 해를 구하고, 후처리기에서 그 결과를 해석하므로 결국 배치처리와 동등한 상태가 되어 해석 도중에 데이터를 이용할 수 없으며, 해석이 완료된 후에 경계조건 등이 바뀌게 되면 해석 결과를 활용하지 못하고, 초기화하여 재계산을 해야 하는 한계가 있다.

본 연구에서는 OOP의 세 가지 특징인 캡슐화, 다형성, 상속성에 의한 장점을 FEM의 프로그램만이 아닌 해석 알고리즘의 측면까지 확장하여 적용하고자 데이터의 가공단위인 요소의 독립성이 유지되는 반복해석 기법을 새로이 개발 하고, 이를 모델화한 후 프로그램으로 구현하여 유용성을 검토해 보고자 한다.

II. 이론적 배경

본 연구의 핵심은, 기존의 FEM에서 전체좌표계를 기준으로 강성행렬을 구성하여 일괄 풀이할 때 필연적으로 OOP에서 보장하고자 하는 독립성을 해치게 되므로 전체 강성행렬을 조합하지 않고 해를 구하는 데 있다. 본 연구에서는 FEM을 구현하되, 모멘트분배법과 같이 전체행렬을 구성하지 않음으로써 요소 단위의 독립성을 유지하며 해를 구하기 위한 반복해의 기법 (Iterative

Method ensuring the Independency of Elements, IMIE)를 개발하였다.

일반 구조물과 같이 자연에 존재하는 물체의 경우 결합된 전체 강성행렬은 정지정부호행렬 (Positive Definite Matrix)이 되므로 반복해를 얻을 수 있고, 이때의 수렴반경도 알려져 있다 (Bath, 1996). 그러나 본 연구와 같이 전체 강성행렬을 구성하지 않고 요소간의 데이터를 이용하는 경우에는 해의 존재성에 대하여 밝히고 수렴성을 검토할 필요가 있다.

1. IMIE의 개발

잘 알려진 Jacobi Method, Gauss-Seidel Method, SOR Method와 같은 반복법에서는 다음단계의 해를 추정하기 위해서 이전단계에서 추정된 해를 모두 알아야 한다. 이러한 이유로 FEM에서 반복법을 이용하여 해석할 경우, 한 요소에서의 해를 추정하기 위해서는 이전단계에서 추정된 다른 요소들의 해를 모두 알고 있어야 한다. 요소의 독립성을 보장하기 위해서는 하나의 요소에 대한 해를 추정할 때 인접 요소만의 해석 값으로 다음 단계의 해를 추정할 수 있어야 한다.

임의의 벡터 x_0 에 대하여 x^{k-1} 을 SOR Method로 구한 벡터라고 하면 w 의 완화인자를 포함하는 SOR Method는 다음과 같이 표현된다.

$$x_j^{(k)} = (1-w)x_j^{(k-1)} + w\overline{x_j^{(k)}} \quad (1)$$

행렬 $A = D - E - F$ 라고 할 때, $\overline{x_j^{(k)}}$ 에 Gauss-Seidel Method 대신 Jacobi Method를 대입하면 식 (2)와 같이 전개할 수 있다.

$$Dx^{(k)} = (1-w)Dx^{(k-1)} + w(Ex^{(k-1)} + Fx^{(k-1)} + b) \quad (2)$$

여기서 D 는 A 의 대각행렬이며 E 와 F 는 각각 A 의 대각원소를 제외한 상삼각행렬과 하삼각행렬이다. 식 (2)에서 완화인자 w 를 묶어내면 식 (3)과 같다.

$$Dx^{(k)} = Dx^{(k-1)} - w[(D-E-F)x^{(k-1)} - b] \quad (3)$$

이때 $A = D - E - F$ 이므로 A 로 치환하여 식 (4)와 같이 수열로 정리할 수 있다.

$$x_i^{(k)} = x_i^{(k-1)} - \frac{w}{a_{ii}} \left[\sum_{j=1}^{i-1} a_{ij}x_j^{(k-1)} - b_i \right] \quad (4)$$

Fig. 1 Schematic diagram to illustrate generalized simple beam structure

식 (4)를 FEM 문제에 일반화하여 적용하기 위해 단순보의 구조를 도식적으로 나타내면 Fig. 1과 같다. 임의의 절점 n 에서 연결된 단순보 요소 n 과 $n+1$ 에 대하여 절점 n 에 대한 전단력과 처짐은 각각 $F_{n1}, d_{n1}, F_{n2}, d_{n2}$ 이다.

두 요소에 대한 전체 강성행렬을 이용하여 절점 n 에 대한 요소 n 의 처짐을 식 (4)에 대입하여 전개하면 식 (5)와 같이 나타낼 수 있다.

$$d_n^{(k)} = d_n^{(k-1)} - \frac{w}{\left(\frac{12}{L_n^3}k_n + \frac{12}{L_{n+1}^3}k_{n+1}\right)} \quad (5)$$

$$\times \left[\frac{-12}{L_n^3}k_n d_{n-1}^{(k-1)} + \frac{-6}{L_n^2}k_n \Phi_{n-1}^{(k-1)} + \left(\frac{12}{L_n^3}k_n + \frac{12}{L_{n+1}^3}k_{n+1}\right) d_n^{(k-1)} + \frac{-12}{L_{n+1}^3}k_{n+1} d_{n+1}^{(k-1)} + \frac{6}{L_{n+1}^2}k_{n+1} \Phi_{n+1}^{(k-1)} - F_n \right]$$

절점 n 에서 $F_n^{(k-1)} = F_{n1}^{(k-1)} + F_{n2}^{(k-1)}$ 이며 정리하면 각각 식 (6)과 식 (7)이 된다.

$$F_{n1}^{(k-1)} = \frac{-12}{L_n^3}k_n d_{n-1}^{(k-1)} + \frac{-6}{L_n^2}k_n \Phi_{n-1}^{(k-1)} + \frac{12}{L_n^3}k_n d_n^{(k-1)} \quad (6)$$

$$F_{n2}^{(k-1)} = \frac{12}{L_{n+1}^3}k_{n+1} d_n^{(k-1)} + \frac{-12}{L_{n+1}^3}k_{n+1} d_{n+1}^{(k-1)} + \frac{6}{L_{n+1}^2}k_{n+1} \Phi_{n+1}^{(k-1)} \quad (7)$$

절점 n 에 접한 각 요소의 처짐을 d_{n1}, d_{n2} 이라고 할 때 $d_n^{(k)} = d_{n1}^{(k)} = d_{n2}^{(k)}$ 이므로 식 (6)과 식 (7)을 식 (5)에 대입하면 요소 n 에서의 처짐은 식 (8)과 같다.

$$d_{n1}^{(k)} = d_{n1}^{(k-1)} - \frac{w}{\left(\frac{12}{L_n^3}k_n + \frac{12}{L_{n+1}^3}k_{n+1}\right)} [F_{n1}^{(k-1)} + F_{n2}^{(k-1)} - F_n] \quad (8)$$

이때 $F_{n1}^{(k-1)}, F_{n2}^{(k-1)}$ 은 각각 절점 n 에 접한 요소의 전단력 이므로 식 (8)을 통하여 임의의 요소 n 의 처짐은 절점 n 에서 접하고 있는 요소만의 추정된 전단력을 이용하여 다음 단계의 처짐을 추정할 수 있다.

2. IMIE의 이론적 검증

반복법으로 해석할 때 해를 보장하기 위해서 위해서는 수렴성에 대한 증명이 필요하다. 앞선 식 (3)에서 양변에 D^{-1} 을 곱하여 정리하면 식 (9)와 같다.

$$x^{(k)} = (I - wI + wD^{-1}E + wD^{-1}F)x^{(k-1)} + wD^{-1}b \quad (9)$$

여기서 F 는 항등행렬이며 $L_w = (I - wI + wD^{-1}E + wD^{-1}F)$ 라고 할 때, L_w 행렬이 수렴할 경우 반복해가 수렴하게 된다. 행렬 $A = (a_{ij})_{n \times n}$ 가 강한 대각지배일 때 $A = D - E - F$ 이므로 $L_w = (I - wD^{-1}A)$ 이다. 따라서 L_w 는 식 (10)으로 나타낼 수 있으며 이때 행렬 L_w 의 무한노름은 식 (11)과 같이 표현할 수 있다.

$$L_w = (l_{ij})_{n \times n} = \begin{cases} 1 - w, & i = j \\ -w \frac{a_{ij}}{a_{ii}}, & i \neq j \end{cases} \quad (10)$$

$$\|L_w\|_{\infty} = \max_{1 \leq i \leq n} (|1 - w| + \sum_{\substack{j=1 \\ j \neq i}}^n | -w \frac{a_{ij}}{a_{ii}} |) \quad (11)$$

행렬 L_w 가 수렴할 필요충분조건은 $\rho(L_w) < 1$ 일 경우 이므로 Gerschgorin의 원 정리에 의하여 $\|L_w\|_{\infty} < 1$ 일 경우 수렴한다 (Jung, 1994). 식 (11)에서 $\sum_{\substack{j=1 \\ j \neq i}}^n | -w \frac{a_{ij}}{a_{ii}} | = k$ 로 치환하면 k 의 범위

는 $0 \leq k < 1$ 이 되며 다음과 같이 수렴조건을 정리할 수 있다.

$$\max_{(1 \leq i \leq n)} (|1-w| + |w| \times k) < 1 \quad (12)$$

완화인자 w 의 범위가 $w \leq 0$ 인 경우 식 (12)는 식 (13)과 같이 정리되는데, 완화인자 w 가 음수이므로 식을 만족하지 않는다.

$$\max_{(1 \leq i \leq n)} ((1+k) \times w) > 0 \quad (13)$$

완화인자 w 의 범위가 $0 < w \leq 1$ 인 경우 식 (12)는 다음과 같이 정리된다.

$$\begin{aligned} \max_{(1 \leq i \leq n)} ((k-1) \times w) &< 0 \\ \Rightarrow \max_{(1 \leq i \leq n)} ((k-1)) &< 0 \\ \Rightarrow \max_{(1 \leq i \leq n)} (k) &< 1 \end{aligned} \quad (14)$$

부등식 식 (14)는 항상 만족하므로 $0 < w \leq 1$ 일 경우 항상 수렴하게 된다. 완화인자 w 의 범위가 $1 < w$ 인 경우 식 (12)는 식 (15)~(16)과 같이 정리할 수 있다.

$$w < \frac{2}{1 + \max_{(1 \leq i \leq n)} (k)} \quad (15)$$

$$\frac{2}{1 + \max_{(1 \leq i \leq n)} (k)} < 2 \quad (16)$$

식 (15)에서 우변항은 식 (16)과 같이 항상 2보다 작다. 따라서, w 의 범위 $0 < w < \frac{2}{1 + \max(k)} < 2$ 에서 수렴된다.

III. 모델의 설계

1. 객체화된 요소 기반의 반복 해석

객체화 된 요소들이 프로그램에서 인스턴스로 구현 되면 캡슐화에 의해 메소드와 데이터를 외부에서 참조할 수 없다. 각각의 독립된 인스턴스가 가지고 있는 메소드를 이용하여 인스턴스 단위의 해석만이 가능하므로 해석 데이터를 상호 교환하는 반복해석 기법을 통하여 전체 구조를 해석 하여야 한다. 이 경우 인접한 두 인스턴스의 해석을 위한 과정 및 상태에 대하여 Fig. 2에 도시하였다. 최초의 해석이 필요하지 않거나 해석이 완료된 경우는 Fig. 2에서 균형상태 (balanced condition)이다. 경계조건이 주어지고 초기 해석이 시작되거나 두 인스턴스 사이에 경계조건이 변경되면 이전 해석 값과 새로운 경계조건 사이의 오차가

Fig. 2 State diagram of model

발생하게 되므로 새로운 경계 조건 (new boundary condition)에 의해 불균형 상태 (imbalanced condition)가 된다. 불균형 상태를 해소하기 위해 인접 인스턴스와 해석 값을 주고받으며 새로운 경계조건과 비교한 뒤 발생한 오차를 분배 통해 줄여 나가며, 이는 Fig. 2에서 해석 값 교환과정 (data exchange)과 오차 분배 과정 (error distribution)에 해당한다. 반복적인 인스턴스간 오차 분배 및 데이터 교환을 통해 오차가 충분히 줄어들면 오차율 체크 (error rate check)를 하며 지정된 오차 범위내로 해석이 완료되면 다시 균형 상태 (balanced condition)가 되어 해석이 완료된 상태가 된다.

2. 모델화를 위한 해석 방식의 적용

하나의 요소에서 응력은 전단력과 굽힘모멘트, 변위는 처짐 및 회전을 같이 쌍으로 나타난다. 따라서 단일 변수에 대한 반복해법인 IMIE를 바로 적용할 수 없으며 한 요소에서 응력인 전단력과 굽힘모멘트, 변위인 처짐과 회전을 쌍으로 해석할 수 있도록 확장하여야 한다. 이를 위해서 OFMIE(Object-oriented Finite element Model ensuring the Independency of Elements, OFMIE)를 개발하였다. 단순보로 이루어진 구조의 임의의 지점에 하중과 모멘트 조건이 주어지는 형태를 Fig. 3과 같이 일반화 하였다. 절점으로 구분된 각각의 요소를 인스턴스로 보고, 전단력과 굽힘모멘트 그리고 처짐과 회전을 쌍으로 하여 반복해석을 하기 위한 인스턴스 단위의 해석과 데이터 교환의 알고리즘을 개발하였다.

가. 인스턴스간 데이터 교환

Fig. 3 단순보 요소 n 의 인스턴스에 대하여 절점의 하중과 변위와의 관계를 강성행렬을 이용하여 행렬 형태로 나타내 보면 다음의 식 (17)과 같다.

$$\begin{Bmatrix} F_n \\ M_n \\ F_{n+1} \\ M_{n+1} \end{Bmatrix} = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix} \begin{Bmatrix} d_n \\ \Phi_n \\ d_{n+1} \\ \Phi_{n+1} \end{Bmatrix} \quad (17)$$

Fig. 3 Data exchange in OFMIE

여기서 F_n 과 F_{n+1} , M_n 과 M_{n+1} 은 각각 절점 n 과 $n+1$ 에서의 전단력과 굽힘 모멘트이며 E 는 탄성계수, I 는 주관성모멘트이고 L 은 길이를 나타낸다. d_n 과 d_{n+1} , Φ_n 과 Φ_{n+1} 은 절점 n 과 $n+1$ 에서의 절점변위와 절점회전이다. 인스턴스 n 을 해석하기 위하여 먼저 인스턴스 $n-1$ 의 절점 n 에서 추정된 d_n , Φ_n 과 인스턴스 $n+1$ 의 절점 $n+1$ 에서 추정된 F_{n+1} , M_{n+1} 을 데이터 교환을 통해 전달받는다. d_{n+1} , Φ_{n+1} 을 추정하기 위해 식 (17)을 d_n , Φ_n , F_{n+1} , M_{n+1} , d_{n+1} , Φ_{n+1} 에 관한 행렬식으로 정리하여 보면 식 (18)과 같다.

$$\begin{Bmatrix} F_{n+1} \\ M_{n+1} \end{Bmatrix} = \frac{EI}{L^3} \begin{bmatrix} -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix} \begin{Bmatrix} d_n \\ \Phi_n \\ d_{n+1} \\ \Phi_{n+1} \end{Bmatrix} \quad (18)$$

전달받은 d_n , Φ_n , F_{n+1} , M_{n+1} 의 값을 이용하여 가우스 소거법을 적용하면 d_{n+1} , Φ_{n+1} 의 값을 추정할 수 있다. 추정된 d_{n+1} , Φ_{n+1} 의 값은 다음 인스턴스 $n+1$ 에 전달되며, 인스턴스 $n+2$ 로부터 전달 받은 d_{n+2} , Φ_{n+2} 의 값을 이용하여 인스턴스 $n+1$ 의 F_{n+1} , M_{n+1} 을 추정하기 위해서 식 (19)와 같은 행렬식을 구성할 수 있다.

$$\begin{Bmatrix} F_{n+1} \\ M_{n+1} \end{Bmatrix} = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \end{bmatrix} \begin{Bmatrix} d_{n+1} \\ \Phi_{n+1} \\ d_{n+2} \\ \Phi_{n+2} \end{Bmatrix} \quad (19)$$

식 (19)의 연산을 통해 인스턴스 $n+1$ 의 절점 $n+1$ 에서의 F_{n+1} , M_{n+1} 을 추정할 수 있으며, 이 값은 다시 인스턴스 n 에 전달된다. 인스턴스 $n-1$ 의 절점 n 에서 전달받은 d_n , Φ_n 을 식 (18)에 다시 적용한 뒤 가우스 소거법을 이용하면 인스턴스 n

에서 d_{n+1} , Φ_{n+1} 의 값을 재 추정 할 수 있다. 재 추정된 d_{n+1} , Φ_{n+1} 값과 d_n , Φ_n 을 이용하여 인스턴스 n 의 절점 n 에 F_n , M_n 을 추정하기 위한 행렬식을 구성하면 다음 식 (20)과 같다.

$$\begin{Bmatrix} F_n \\ M_n \end{Bmatrix} = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \end{bmatrix} \begin{Bmatrix} d_n \\ \Phi_n \\ d_{n+1} \\ \Phi_{n+1} \end{Bmatrix} \quad (20)$$

식 (20)을 통해 추정된 인스턴스 n 의 절점 n 에서의 F_n , M_n 은 다시 인스턴스 $n-1$ 로 전달하게 되며 인스턴스 n 에 대한 한번의 추정과정에 대한 알고리즘이 완성된다. 이 과정을 모든 인스턴스에 대해 적용하여 불균형 상태가 해소될 때까지 데이터 교환을 반복한다.

나. 오차의 분배

인스턴스간의 점점에는 작용력 또는 작용 모멘트에 대한 경계조건이 존재한다. 매 해석과정에서 해석 값과 경계조건과의 비교과정이 필요하며 이때 오차가 존재할 경우 이를 양단에 분배하여 오차를 줄여나간다. Fig. 4에서 F_{n-1} , F_n , M_{n-1} , M_n 은 절점 n 에서 인접한 두 인스턴스의 전단력과 굽힘 모멘트에 대한 해석 값이며 F , M 은 인스턴스 n 에서의 경계조건에 해당한다.

절점에서 경계조건인 작용 하중 F 는 절점 양단의 전단모멘트

Fig. 4 Error distribution in OFMIE

F_{n-1} , F_n 의 합과 같고 작용 모멘트 M 은 절점 양단의 굽힘 모멘트 M_{n-1} , M_n 의 합과 같다. 따라서 절점 n 에서 F_{n-1} , F_n , M_{n-1} , M_n 은 경계조건 F , M 에 대하여 다음 식 (21), (22)를 만족한다.

$$F_{n-1} + F_n = F \quad (21)$$

$$M_{n-1} + M_n = M \quad (22)$$

절점의 양단에서 해석 값이 식 (21), (22)를 만족하지 못할 경우 오차에 의한 불균형 상태 이므로 오차분배가 필요하다. 이때 다음 식 (23), (24)과 같이 오차분배에 대한 알고리즘이 만들어진다. 여기서 분배율인 0.05는 수렴하기 위한 완화인자 w 의 범위를 고려하여 선택하였다.

$$F_n = F_n + (F - (F_{n-1} + F_n)) \times 0.05 \quad (23)$$

$$M_n = M_n + (M - (M_{n-1} + M_n)) \times 0.05 \quad (24)$$

오차의 분배는 전단력과 굽힘 모멘트에 해당되므로 데이터 교환과정 중 전단력과 굽힘 모멘트의 교환과정에서 이루어진다.

3. 클래스의 구현

설계된 모델을 실제 프로그램으로 구현하기 위하여 필요한 클래스와 기능을 분류하면 Table 1과 같다. 구현과정에서 OOP에 적합한 JAVA™ 언어를 이용하였다 (Lee et al., 1998). MainProc 클래스는 메인 프로그램의 운용을 위한 기능을 한다. GraphicPannel 클래스는 터치스크린에 기반하여 인터랙티브한 그래픽 입출력 환경을 구현한다. ProcessThread 클래스는 인스턴스화된 요소들의 해석과 조작을 동시에 지원하기 위한 스레드 모듈이며 ElementListManager 클래스는 사용자로부터 인터랙티브하게 입력받은 데이터와 인스턴스간의 관계를 관리하는 프레임이다. Element는 요소를 객체화하기 위한 클래스이며 MatrixUtil 클래스는 행렬 연산을 위한 라이브러리이다.

Table 1 Classification of classes

Class	Function
MainProc	Main Procedure, Implement of Element's instance
GraphicPannel	Graphical I/O
ProcessThread	Process control
ElementListManager	Management of element structure
Element	Exchanging analysis data, Error distribution
MatrixUtil	Matrix solver

가. MainProc 클래스

프로그램의 운용을 위한 메인 프로시저로써 프로그램 전반의 동작과 입력받은 데이터로 요소의 인스턴스를 구현하는 클래스이다. OOP기법이 적용된 프로그램의 특성상 메인 프로시저는 많은 기능을 가지고 있지 않다. 입력받은 데이터를 이용하여 요소를 인스턴스화 하고 ElementListManager로 관리한다.

나. GraphicPannel 클래스

해석하고자 하는 구조의 디자인과 결과의 출력을 직관적으로 수행하기 위해 터치스크린에 기반한 입출력 모듈의 클래스이다. 화면에 요소를 나타내는 아이템을 드래그 하여 문제를 디자인할 수 있으며, 원하는 요소를 터치하여 조건을 수정할 수 있도록 그래픽 환경을 지원한다. 또한 해석과정에 따라 데이터 변화를 실시간으로 출력을 하여 연구자가 확인할 수 있도록 도와준다.

다. ProcessThread 클래스

해석이 진행되는 동안 요소의 추가, 수정 및 메시지 출력 등 다른 동작이 가능하도록 멀티 스레드 기술을 적용하여, 인스턴스의 해석은 스레드 기반의 클래스로 구현하였다. 해석 종료조건을 판별하기 위해 해석 상태를 메인 프로시저에 전달할 필요가 있으므로, 메시지를 주기적으로 발생 및 처리하는 역할을 수행한다.

라. ElementListManager 클래스

데이터를 입력받아 요소를 인스턴스로 구현할 때 인스턴스간의 관계를 정의할 필요가 있다. 이를 위해 프로그램 상에서 인스턴스간의 관계를 관리하기 용이하도록 구현된 프레임이며 요소가 추가, 삭제되거나 수정될 때 용이하게 동작할 수 있도록 도와주는 모듈이다.

마. Element 클래스

해석 Element는 인스턴스화 되는 요소를 구현하는 클래스이다. OFMIE를 기반으로 설계되어 인스턴스화 된 요소의 반복해석과 데이터 교환, 오차의 분배를 수행한다. 이를 코드화 한 결과는 Fig. 5와 같다.

바. MatrixUtil 클래스

해석과정에서 인스턴스 단위로 행렬의 곱셈과 역행렬의 계산이 필요하다. 이러한 연산과정은 계산 효율에 영향을 크게 미치므로 지속적인 업데이트가 필요하며 각각의 인스턴스에서 동일하게 사용되므로 라이브러리 형태로 구성할 필요가 있다. MatrixUtil 클래스는 각각의 연산 메소드를 멤버로 가지고 있으며 인스턴스에서 필드로 선언되어 사용된다.

```

1 public void nextCalc(double[] d,double[] f) {
2     //d[] is buffer adress to use guessed deflection and angle of previous element
3     //f[] is buffer adress to return guessed force and moment to previous element
4
5     guessDAVarofRightElement(boundaryFMVar[0],boundaryDAVar[0]);
6     //guess deflection and angle using partial inverse kmatrix
7
8     DA[0] = boundaryDAVar[0][2];
9     DA[1] = boundaryDAVar[0][3];
10    //save guessed deflection and angle to temporary buffer DA[]
11
12    next.nextCalc(DA,FM);
13    //caluate next element of force and moment using guessed deflection and angle at previous step
14    //save guessed force and moment of next element to temporary buffer FM[]
15
16    double ef = ( boundaryFMVar[1][2] - ( FM[0]+boundaryFMVar[0][2] ) );
17    double em = ( boundaryFMVar[1][3] - ( FM[1]+boundaryFMVar[0][3] ) );
18    //error distrubtion using guessed force and moment
19
20    boundaryFMVar[0][2] = boundaryFMVar[0][2] + (ef)*distRate;
21    boundaryFMVar[0][3] = boundaryFMVar[0][3] + (em)*distRate;
22    //reflect error rate in force and moment
23
24    mx.guessRightElementDAVal(boundaryFMVar[0],boundaryDAVar[0]);
25    //guess deflection and angle at right side of element using recalculated force and moment
26
27    mx.getMultiKMatrix(boundaryDAVar[0],boundaryFMVar[0]);
28    //calculate force and moment using guessed deflection and angle at previous step
29
30    getNext().nextCalc();
31 }

```

Fig. 5 Virtual code of OFMIE

IV. 결과 및 고찰

1. 프로그램의 구현

설계된 클래스를 이용하여 개발된 OFMIE모형을 프로그램으로 구현하였다. 프로그램은 사용자의 손쉬운 조작을 위해 터치스크린을 기반으로 하고 있으며 구현된 프로그램의 모습은 Fig. 6과 같다. 프로그램 화면 상단에 준비된 요소의 아이টে를 드래그 하여 화면에 끌어다 놓으면 새로운 요소가 인스턴스화 되어 추가되며 이러한 방법으로 해석하고자 하는 구조를 디자인할 수 있다. 관리하고자 하는 요소를 터치하면 해당 요소에 대한 특성 및 경계조건 등을 수정할 수 있다.

가. 해석과 전처리 과정의 통합

본 연구의 모델에서는 기존의 FEM프로그램에서 해석을 위해 필수적인 전처리 과정이 생략된다. 프로그램에서 요소는 자신의 해석 메소드와 데이터를 가지고 있는 상태로 인스턴스화 되며 이들의 상호 데이터 교환과 반복과정으로 해석이 진행되기 때문에 전처리 과정이 따로 필요하지 않다. 즉, Fig. 7과 같이 화면에서 입출력에 사용되는 요소의 인스턴스가 해석과정에도 바로 이용되므로 해석과정에 전처리 과정이 통합된다.

나. 해석과 후처리 과정의 통합

프로그램에서 객체화된 요소의 인스턴스별로 해석 데이터를 가지고 있기 때문에 후처리 없이 해석된 응력과 변위를 바로 이용할 수 있다. 프로그램에서는 Fig. 8과 같이 그래픽으로 출력되는 요소별로 해석결과가 실시간으로 업데이트 되며 해석 상태에 대한 정보도 손쉽게 표현된다. 전처리 과정의 통합과 마찬가지로 화면에 출력되는 요소의 인스턴스에서 해석도 이루어지므로 전체 행렬의 조합 없이 인스턴스별로 독립성이 유지되어 해석결과가 보존되므로 후처리 과정의 생략이 가능하다.

이와 같은 전처리, 후처리, 해석과정의 통합으로 프로그램의 구조가 단순해지기 때문에 GUI 등의 구현과정에서 개발자가 직관적으로 접근할 수 있으며 코드의 재사용이나 확장이 용이하다.

다. 해석 진행 중 데이터 사용의 연속성

디자인된 구조를 해석하는 도중에 Fig. 9와 같이 새로운 요소가 삽입되어 구조가 변경되더라도 변경시점까지 해석된 결과를 바탕으로 이어서 해석이 이루어진다. 객체화된 요소의 인스턴스가 각각 독립성을 유지하고 있으며 전체 행렬을 조합하지 않기 때문에 해석 도중에 새로운 요소가 추가되거나 조건이 변경되더라도 개별 인스턴스에 영향을 미치지 않으므로 연속적인 해석이 가능하다.

Consol box

2 ANDROID MODULE FOR OOP BASED FEM. V.1.0
 3 Program designed by genesiz
 4 FEM엔진 구성이 완료되었습니다.
 5 그래픽 환경이 준비되었습니다.
 6 예제 1번 입니다.

Rural Systems Engineering

Fig. 6 GUI of Program

Fig. 7 Analysis model without preprocessor

DL : .00	32.37	153.62	191.80	127.99
AL : .00	59.15	90.52	-10.62	-106.36
DR : 37.96	110.26	192.69	132.07	.00
AR : 69.32	89.49	-8.96	-99.02	-138.81

Fig. 8 Analysis model without postprocessor

Fig. 9 Continuity of data usage with analysis in progress

라. 해석 종료 후 데이터 사용의 연속성

디자인된 구조의 해석이 한번 완료된 후 Fig. 10의 붉은색 박스부분과 같이 조건을 일부 변경할 경우에 해석이 완료된 상태에서 로부터 데이터를 사용하여 연속적인 해석이 가능하다. 객체화 된 요소의 인스턴스의 독립성이 유지되므로 해석 결과에 대한 데이터 또한 독립적으로 보관된다. 또한 전체 행렬의 조합과정이 없으므로 요소의 추가나 조건의 변경 등에 따른 재조합 과정이 필

Fig. 10 Continuity of data usage when analysis is completed

요 없다. 따라서 재계산을 위한 선행 작업이 생략되므로 해석이 완료된 이후의 데이터를 연속적으로 사용할 수 있다.

2. 프로그램의 해석 검증

본 모델에 의한 FEM 프로그램 해석의 정확성을 검증하기 위하여 Fig. 11과 같은 구조를 해석하여 보았으며 결과를 실제 해와 비교하여 보았다. 그 결과 Table 2에서와 같이 유효숫자 범위에서 실제 해와 동일하게 해석됨을 알 수 있다.

Fig. 11 Structure for verification

3. 복잡도의 증가에 따른 OFMIE 효과

기존의 FEM에서는 대상의 해석이 종료된 이후 요소를 추가, 삭제되거나 조건이 일부 변경 되었을 때 다시 해석 하려면 전체 강성행렬을 새롭게 조합하여야 하므로 이전의 계산결과를 이용할 수가 없다. 본 연구에서는 해석 데이터가 객체화된 요소의 인스턴스별로 보관되며, 새로운 조건에 대하여 인스턴스 단위로 독립적인 해석이 이루어지기 때문에 변경되기 이전 단계에서의 해석 값을 변경된 후의 해석에 이용할 수 있다. 조건이 변경되지 않은 다른 요소의 인스턴스에서는 새로운 조건에 대하여 영향을 적게 받기 때문에 이전에 해석된 상태에서의 데이터를 이어서 사용하면 계산횟수를 단축시킬 수 있다. 계산 단축효과를 정량화하여 비교하기 위해 양단이 각각 고정단과 힌지로 지지되어 있으며 절점마다 하중 50 kN과 50 kN·m이 재하 된 1 m 길이의 단순보 요소로 이루어진 규칙적인 구조에 대하여 해석이 완료된 후 조건이 변경 되었을 때, 기존 FEM 프로그램에서와 같이 처음부터 새로 계산 하는 경우와 OFMIE의 단계적 계산을 하는 경우의 계산 횟수를 비교하여 보았다. 5개부터 20개까지 요소의 수를 늘

Fig. 12 Comparison of analysis FEM and OFMIE

Table 2 Verification of value

		Deflection		Angle		Shear force		Bending Moment	
		result	error rate	result	error rate	result	error rate	result	error rate
Case1	Point1	0.00	0.00	0.00	0.00	118.06	0.00	− 208.33	0.00
	Point2	84.49	0.00	149.31	0.00	118.06	0.00	− 90.28	0.00
						68.06	0.00	− 40.28	0.00
	Point3	242.59	0.00	155.56	0.00	68.06	0.00	27.28	0.00
						18.06	0.00	77.78	0.00
	Point4	356.25	0.00	68.75	0.00	18.06	0.00	95.83	0.00
						− 31.94	0.00	145.83	0.00
	Point5	357.41	0.00	− 61.11	0.00	− 31.94	0.00	113.89	0.00
						− 81.94	0.00	163.89	0.00
	Point6	228.01	0.00	− 184.03	0.00	− 81.94	0.00	81.94	0.00
− 131.94						0.00	131.94	0.00	
Point7	0.00	0.00	− 250.00	0.00	− 131.94	0.00	0.00	0.00	
Case2	Point1	0.00	0.00	− 25.67	0.00	− 2.07	0.00	0.00	0.00
	Point2	− 26.01	0.00	− 26.70	0.00	− 2.07	0.00	− 2.07	0.00
						7.93	0.00	2.93	0.00
	Point3	− 63.58	0.00	− 1.05	0.00	7.93	0.00	20.39	0.00
						− 4.07	0.00	28.39	0.00
	Point4	− 59.25	0.00	15.25	0.00	− 4.07	0.00	25.95	0.00
						− 4.07	0.00	13.95	0.00
	Point5	− 32.08	0.00	29.06	0.00	− 4.07	0.00	9.07	0.00
− 9.07						0.00	9.07	0.00	
Point6	0.00	0.00	33.59	0.00	− 9.07	0.00	0.00	0.00	

려가며 각 케이스 별로 마지막 절점에서 하중조건을 5 % 변경하는 방법으로 실험하였다.

기존 FEM과 OFMIE 결과의 계산횟수를 비교한 결과는 Fig. 12와 같다. 요소가 추가되어 계산 조건이 변경되었을 때 기존의 FEM은 처음부터 다시 계산하여 계산 횟수가 지속적으로 증가하지만 OFMIE에서 제시하는 모델의 경우 데이터의 연속적인 이용을 통해 선형에 근사하여 계산시간이 증가하는 장점을 갖는다. 요소가 20개인 경우 기존의 방법에 비해 계산 횟수를 약 75 % 개선하는 효과를 확인하였다. 계산요소가 복잡해질수록 데이터의 재사용으로 인한 계산 횟수의 단축효과가 커짐을 알 수 있으며 이는 독립된 요소의 인스턴스 별로 해석되므로 해석 데이터가 보존되어 변경된 조건에 대한 중복적인 해석이 줄어들기 때문으로 판단된다.

V. 결 론

FEM 연구가 발전 되면서 적용 대상이 다양해지고 FEM 프로그램 또한 복잡 거대화 되어가고 있다. 이러한 FEM 프로그램을 좀 더 효율적으로 연구, 개발하기 위해서 OOP 기법을 적용하는

것은 필연적으로 보인다. 그러나 지금까지의 연구들은 POP 기반으로 작성된 해석 알고리즘을 재활용하기 위해 프로그램 측면에서만 OOP 기법을 적용하였다. 때문에 프로그램이 지나치게 객체화 되어 오히려 복잡해지거나 OOP의 특징인 캡슐화, 다형성, 상속성을 해치는 결과를 초래하였다. 본 연구에서는 이러한 문제를 해결하고 OOP의 장점을 해석 알고리즘 측면까지 확장하여 적용하기 위해 데이터 가공 및 처리의 단위가 되는 요소를 객체화의 대상으로 하였다. 요소의 메소드와 데이터의 독립성이 유지되는 반복해법을 개발하였고, 이를 기반으로 객체화된 요소의 인스턴스의 상호 데이터 교환을 통해 해석하는 모델을 설계하여 프로그램으로 구현하였으며 다음과 같은 결론을 얻었다.

1. FEM에서 OOP기법을 해석 알고리즘 측면으로 확장하여 적용하기 위해 데이터 가공 및 처리의 단위가 되는 요소를 객체화의 대상으로 하여 독립성이 유지되는 반복해법 (IMIE)을 개발하였으며, 이론적 검증을 통해 전체 행렬을 조합할 필요 없이 독립된 개별 요소들의 해석을 전체 구조의 해석결과로 귀결시킬 수 있음을 확인하였다.

2. IMIE를 프로그램화 하기 위해 OFMIE로 모델화 하였으며 실제 프로그램으로 구현한 결과 전처리, 후처리, 해석과정의 통

합이 가능하였다. 단순해진 구조를 통해 개발자의 입장에서 프로그램의 재활용 및 확장이 용이할 것으로 판단된다.

3. 프로그램으로 구현된 OFMIE의 모델에서 해석 도중이나 해석이 완료된 후 조건이 변경될 경우 변경이전의 데이터를 이어서 사용할 수 있었다. 사용자의 입장에서 해석 도중에 구조의 조건을 변경하거나 요소를 추가할 수 있으므로 해석 데이터의 재사용성이 높아질 것으로 판단된다.

4. OFMIE를 프로그램으로 구현하여 적용한 예에서 계산결과 실제 해석 값과 오차범위 내에서 일치하였다. 이를 통해 실제 문제 해석에 대한 정확성이 입증되었다.

5. 요소가 추가되거나 조건이 변경되기 이전단계에서의 해석결과를 변경된 후의 해석에 이용할 수 있어 계산 횟수를 단축할 수 있었고 복잡한 문제일수록 그 효과가 커짐을 확인하였다. 따라서 복잡한 구조에서 일부 조건이 변경될 경우 해석시간을 단축시킬 수 있을 것으로 판단된다.

본 연구는 농림수산식품부 생명산업기술개발사업 기획과제인 “농업생산기반시설 통합 관리시스템 개발” 연구비지원 (과제번호 : 311009-3)에 의해 수행되었습니다.

REFERENCES

1. Archer, G. C., G. Fenves., C. Thewalt., 1999. A new object-oriented finite element analysis program architecture. *Computers and Structures* 70(1): 63-75.
2. Bath, K-J., 1996. *Finite Element Procedures*, New Jersey: Prentice-Hall, Inc.
3. Cho, J. J., 2009, Object-oriented finite element framework using hybrid programming. Ph.D. diss., Seoul, Ind.: Seoul National University.
4. Fenves, G. L., 1990. Object-Oriented Programming for Engineering Software Development. *Engineering with Computers* 6(1): 1-15.
5. Forde, B. W. R., R. O. Foschi., and S. F. Stierner., 1990. Object-oriented finite element analysis. *Computers & Structures* 34(3): 355-374.
6. Heng, B. C. P., R. I. Mackie., 2009. Using design patterns in object-oriented finite element programming. *Computers and Structures* 87(15-16): 952-961.
7. Irons, B. M., 1970. A frontal solution program for finite element analysis. *International Journal for Numerical Methods in Engineering* 2: 5-32.
8. Jung, S. G., 1994. *Finite Element Method*, Seoul: Minumsa.
9. Kong, X. A., and D. P. Chen., 1994. An object-oriented design of fem programs. *Computers & Structures* 57(1): 157-166.
10. Kumar, S., 2010. Object-oriented finite element programming for engineering analysis in c++. *Journal of Software* 5(7): 689-696.
11. Lee, J. J., H. J. Yi, 1998. Development of an object-oriented finite element analysis program using javaTM. *Journal of The Korean Society of Agricultural Engineers* 40(2): 130-139 (in Korean).
12. Logan, D., 2011. *A First Course in the Finite Element Method*, CL-Engineering.
13. Mackie, R. I., 1997. Using objects to handle complexity in finite element software. *Engineering with Computers* 13: 99-111.
14. Nie, J. H., D. A. Hopkins., Y. T. Chen., H. T. Hsieh., 2010. Development of an object-oriented finite element program with adaptive mesh refinement for multi-physics applications. *Advances in Engineering Software* 41: 569-579.
15. Rebecca, W-B., 1990. *Designing Object-Oriented Software*, New Jersey: Prentice-Hall, Inc.
16. Satzinger, J. W., 1996, *The object-oriented approach: concepts, modeling, and system development*, Danvers: Boyd & Fraser Pub. Co.
17. Sloan, S. W., 1986. An algorithm for profile and wavefront reduction of sparse metrics. *International Journal for Numerical Methods in Engineering* 23: 5-32.
18. Zimmermann, T., Y. Dubois-Pelerin., and P. Bomme., 1992. Object-oriented finite element programming: I. governing principles. *Computer Methods in Applied Mechanics and Engineering* 98: 291-303.