Java™ 언어를 이용한 객체 지향 유한 요소 해석 프로그램의 개발

Development of an Object-Oriented Finite Element Analysis Program Using Java™

이 정 재* · 이 호 재**
Lee, Jeong Jae · Yi, Ho Jae

Summary

The finite element analysis program should be prepared to deal with many of newly arising engineering problems. The sequential structured programming technique does not make a finite element method so flexible. So far, the object oriented programming technique was studied as an alternative programming paradigm. However, most of the research were in the state of the evaluation of the possibility and the applicability of the object oriented method for a finite element program.

In this study, a practical object oriented finite element analysis program, OOFE_JAVA was developed and the result of the analysis on a rectangular clamped plate was shown. The objects which compose the OOFE_JAVA were applied to several engineering problem without any modification and it was concluded that the object oriented technique was appropriate for the development of a complex and large engineering system. And a virtual machine which Java language is using can be loaded on any kinds of computer which has java interpreter regardless of the platform on which the OOFE_JAVA was developed.

I. 서 론

유한 요소 해석은 구조 역학 문제를 해결하기 위한 방법으로 개발되었으나, 열역학, 유체역학 등 미분방정식을 지배방정식으로 하는 대부분의 문제를 해석하는 대표적인 수치해석 기법으로 발전되어 왔다. 근래에 들어 유한 요소 해석은 물속에서의 물질의 이송 및 확산, 수분의 토양내 이동, 곡물 또는 목재의 건조시에 발생하는 열역학적 문제 등 농업에 관련된 문제를 해결하는데에도 이용되고 있다.[10] 그러나 유한 요소 해석을 일반 문제에 적용하기 위해서 기존에 개발된 유한 요소 해석 프로그램을 그대로 이용하는데 어려움이 많아 사용자가 프로그램을 다시 설계, 제작하여야 하는 경우가 많고, 컴퓨터의 사양이나 운영 체제의 종류 또는 버전이 달라지면 새로운 프로그램을 작성하여야 하였다. 또한 지금까지 많은 범용 유한 요소 해석 프로그램이 개발되어

*서울대학교 농업생명과학대학
**서울대학교 대학원

키워드 : 유한 요소 해석, 객체 지향 프로그래밍

있는 구조 해석 분야에서도 새로운 구조 재료를 이용하거나, 발전된 해석 기법 등을 반영하기 위해서는 프로그램의 핵심 부분을 수정하여야 하는 어려움이 있다. 이를 해결하기 위하여 기존 프로그램을 손쉽게 확장하여 프로그램을 개발하는 노력을 줄이며 개발된 프로그램의 수명을 늘일 수 있는 객체 지향 기법을 유한 요소 해석 프로그램에 이용하기 위한 노력이 계속되어왔다.[3,6]

객체 지향 기법을 적용하여 개발된 언어로는 APL, Ada, Clu, Loops, Small Talk 등이 있으나 대부분 일반화되지 못하였다. 1980년대 중반에 발표된 C++는 C와의 호환성을 염두에 둔 객체지향언어와 절차적 언어의 중간적인 성격을 띠고 있어 C를 이용하여 개발된 프로그램을 객체 지향적으로 변환 할 수 있는 이점이 있으나, FORTRAN과 같은 언어를 사용하던 사람은 C와 객체 지향 개념을 모두 습득하여야 하는 어려움이 있다.

1980년대 중반 이후 공학 분야에서는 대규모 프로그램의 개발과 유지 보수에 있어 객체 지향 기법이 가지는 뛰어난 재사용성과 확장성을 적용하려는 연구가 진행되어 Lee[6] 는 객체 지향 기법이 공학용 소프트웨어의 개발에 이용될 수 있음을 보였고, Forde[3] 등은 FORTRAN으로 개발된 유한 요소 해석 프로그램 NAP을 기본으로 C++을 이용하여 객체 지향적 유한 요소 해석 프로그램 Object NAP을 개발하는 등 객체 지향 프로그래밍 언어의 적용성과 개발 과정의 효율에 관하여 연구하였으며, 신[16]은 직접 강도법에 이 개념을 적용하였다. Miller[9]는 객체지향기법을 소개하며 통합 구조 해석 시스템에 대한 적용성을 검토하였으며, Zimmermann[15]은 SmallTalk을, Mackie[7], Scholz[13], Kong[5] 등은 C++과 같은 객체 지향 언어를 이용한 유한 요소 해석 프로그램을 개발하여 그 유용성을 살펴보았다. Mackie[8]는 그래픽을 이용한 유한 요소 구조 해석 프로그램에 객체 지향 기법을 적용하여 잘 설계된 객체가 그래픽 표현과 유한 요소 해석에 공통으로 이용될 수 있음을 보였다. 그러나 지금까지의 연구는 주로 기존의 절차적 언어로 개발된 프로그램을 객체 지향 언어로 변환하여 보거나, 기본적인 유한 요소 해석의 구현을 위하여 기초적인 객체를 규정하여 객체 지향 유한 요소 해석 프로그램의 실현과 발전의 가능성을 검토하는데 그치는 경향이 있다.

객체 지향 언어인 Java는 1995년 Sun Microsystems에서 객체 지향 기법을 근간으로 개발되었으며, 가상 기계(Virtual machine) 개념이 적용되었기때문에, 한번 개발된 프로그램은 Java 실행 인터프리터가 설치된 어떠한 컴퓨터에서도 사용할 수 있으므로 Java를 이용한 객체 지향 프로그램은 개발 환경과 다른 시스템에 맞도록 변환하는데 필요한 비용과 노력을 줄일 수 있다. 또한 네트워킹과 병렬 처리에 필요한 클래스들을 제공하므로 분산 처리를 위하여 PVM(Parallel Virtual Machine)이나 CPS(Cooperative Processes Software) 등 분산 처리를 위한 별도의 프로그램을 이용하지 않고도 분산 처리를 구현할 수 있으며, 그래픽, 멀티미디어 자료 처리에 필요한 클래스를 포함하고 있어 사용자 인터페이스 등의 개발을 보다 쉽게할 수 있다.[1,11]

본 연구에서는 판의 휨 해석을 예로 하여 Java 언어를 이용한 분산 처리, 사용자 인터페이스 등이 포함된 객체 지향 유한 요소 해석 시스템(An Object-Oriented Finite Element System by Java : OOFEJ_AVA)의 원형을 제시하고, 개발된 프로그램을 워크스테이션과 IBM 호환 컴퓨터에서 비교 운용하여 가상 기계 개념의 적용성을 평가하며, 기존의 프로그램과 비교함으로써 Java 언어가 유한 요소 해석 프로그램의 구현에 적합한 개발 환경이 될 수 있음을 보이고자 하였다.

Ⅱ. 유한 요소 해석 객체의 설계

1. 객체의 설계

유한 요소 해석은 그 자체가 모듈화되어 있어 객체로 접근하기 용이하지만 유한 요소 해석만을 구현하는 단순한 프로그램은 절차적 언어로도 충분히 구현할 수 있다. 따라서 유한 요소 해석의 객체를 설계할 때는 다음과 같은 상황의 복잡한 프로그램도 쉽게 작성할 수 있도록 하여야 그 장점을 살릴 수 있다.

- 상호 관련이 있는 다수의 프로그램 작성
- 수치 해석, 그래픽, 데이터베이스 등의 서로 다른 부분이 함께 포함되는 시스템의 개발
- 규모가 큰 프로그램의 작성[8]
- 기존 프로그램의 수정이나 새로운 기능의 추가

특히 Java는 그래픽, 데이터베이스 등에 관련된 함수와 인터넷등 컴퓨터 사이의 통신에 필요한 네트워크, 병렬 처리 등에 관한 함수를 직접 제공하고 있어, 이러한 기능을 이용하기 위해서 별도의 프로그램 라이브러리를 사용해야 하는 C++보다 쉽게 다양한 기능을 가진 유한 요소 해석 시스템을 개발할 수 있다. 확장성이 뛰어난 객체 지향 프로그램의 개발을 위해서는 개발 과정에서 가장 기본적인 자료와 메쏘드만을 갖는 객체의 설계가 이루어져야 한다.

본 연구에서는 Miller[9]가 제시한 객체의 결정, 객체 기능의 정의, 그리고 프로그램 구현의 순서로 이루어진 객체 지향 프로그램 개발의 3단계에 따라 먼저 기본적인 유한 요소 해석 절차로부터 객체 지향 유한 요소 해석 프로그램의 개발에 필요한 가장 기본적인 클래스를 Table 1과 같이 분류하였다.

이들 중 Node, Material, Load, LoadCombination과 Element 클래스는 유한 요소 해석에 필요한 기하학적 정보, 하중, 강성도 행렬 구성을 위한 상수들 등의 자료를 운용한다. 특히 Node, Material, Load, LoadCombination, Matrix 그리고 Solver 클래스는 유한 요소 해석 이외에도 컴퓨터를 이용한 구조 해석 문제 등에도 공통적으로 사용할 수 있도록 클래스를 정의할 때, 유한 요소 해석 절차가 포함되지 않도록 하여야 다른 프로그램을 개발할 때 객체 지향 프로그래밍의 장점을 살려 별다른 수정 없이 바로 이용할 수 있다. I/O 클래스와 Conversion 클래스는 유한 요소 해석에 사용되는 자료의 입력과 출력을 담당하며 프로그램과 파일 또는 사용자를 연결하는 역할을 한다. Matrix 클래스와 Solver 클래스는 수치 해석에 이용되는 Matrix 연산과 연립방정식의 해석에 사용된다. 대부분의 수치 해석 기법은 미분 방정식을 연립 방정식으로 변환하므로 행렬을 사용하여 문제를 해결한다. 따라서 잘 설계된 Matrix 클래스와 Solver 클래스는 수정없이 다른 수치 해석 기법에 사용할 수 있다. FEM 클래스는 유한 요소 해석의 진행을 관리하는데 정적 해석을 기본으로 객체를 확장하여 시계열 해석이나 더욱 복잡한 시스템을 구성하는 클래스가 된다.

Table 1. Classification of classes

Class	Function
I/O	input of data and output of result
Matrix	calculation of matrix
Node	topological data and boundary condition of nodes
Material	material property
Load	applied load
LoadCombination	load combination applied to one node
Element	element in Finite Element Method
Solver	solver of simultaneous linear equations
Conversion	converting result to user's requirement
FEM	FEM procedure

2. 클래스의 정의

가. 입/출력 클래스

유한 요소 해석에 필요한 자료의 입력은 Node 클래스와 Element 클래스의 자료를 저장하는 NodeData, ElementData를 인스턴스로 하는

```
public class ReadData{
   ReadData(
short              typeOfProblem,    //type of the FE Problem
short              typeOfElement,    //type of the Element
Load[]             loads,            //loads applied
LoadCombination[]  loadCombinations, //load combinations used
Material[]         materials,        //materials used
Node[]             nodes,            //nodes of the problem domain
Element[]          elements)         //elements of the FE problem
}
```

Fig. 1. Data used in FEA procedure(read data class)

ReadData 객체를 통하여 이루어 진다. Fig. 1에 보인 ReadData 클래스는 Java에서 제공하는 입/출력 클래스를 이용하여 자료 파일을 열고 미리 정해진 형식에 따라 자료를 읽고 저장한다.

ReadData 클래스는 Fig. 1에서와 같이 생성시에 읽어들인 자료를 저장할 공간을 인자로 받아 관리하며 이 때 받은 인자는 ReadData에 속해 있지 않으므로 자료의 입력이 끝나고 ReadData 클래스는 파일과 자료 저장 공간 사이의 중개자 역할만을 하게 된다.

나. Matrix 클래스

행렬은 많은 수치 해석 기법의 기본이 되므로 유한 요소 해석 절차와는 독립적으로 설계하여야 다른 수치 해석법을 구현 할 때 수정없이 바로 사용할 수 있는 객체가 될 수 있다. 유한 요소 해석에서는 행렬과 벡터의 연산으로 계산이 진행된다. 본 연구에서는 벡터 클래스를 별도로 정의하는 일반적인 예와는 달리 Fig. 2 처럼 1행 또는 1열의 행렬로 생각하였다. 행렬 클래스는 그 크기를 나타내는 변수와 값을 저장하는 변수를 인스턴스로 가지며 생성자에서는 주어진 크기에 따라 기억 공간을 할당한다. 행렬 연산을 위해서 행렬의 덧셈, 뺄셈, 행렬 곱하기, 전치 행렬 등의 연산을 메쏘드로 갖는다. Matrix 클래스는 Fig. 2와 같이 구현된다.

다. Node 클래스

절점은 수치 해석 기법 뿐만이 아니라 그래픽을 이용한 수치 해석기의 전·후 처리기 등의 기

```
public class Matrix{
/* ------------------------------------------------------------
** Fields
**------------------------------------------------------------ */
    int        row, column ;  //number of row and column of
    double[]   value ;        //Data of the Matrix
/* ------------------------------------------------------------
** Constructors
**------------------------------------------------------------ */
   Matrix(){}
   Matrix(int row, int column){
       this.row=row;
       this.column=column;
       value=new double[row * column];
   }
/* ------------------------------------------------------------
** Methods
**------------------------------------------------------------ */
   public double GetValue(int i, int j){}      //get the value of(i,j)
   public void printMatrix(){}                 //print this Matrix
   public void Plus(Matrix a, Matrix b){}      //Matrix Plus
   public void Minus(Matrix a, Matrix b){}     //Matrix Minus
   public void DetProduct(Matrix a, Matrix b){} //Dot Product of Matrix
   public void Transpose(Matrix a){}           //Transpose of Matrix
```

Fig. 2. Matrix calculation(matrix class)

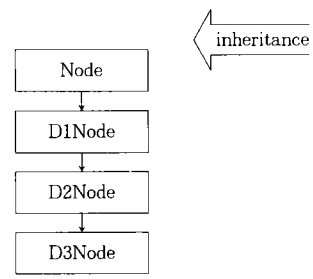

Fig. 3. Extending node classes

본이 되므로, 본 연구에서는 Node 클래스 역시 유한 요소 해석 절차로부터 독립적으로 사용될 수 있도록 설계하였다. Node 클래스는 기본적으로 경계 조건과 절점에 가해지는 힘의 종류를 나타내는 정수를 인스턴스로 갖는다. 절점의 위치를 저장하는 인스턴스는 기본적인 Node 클래스를 계승하여 1차원 절점, 2차원 절점, 3차원 절점 등으로 확장하였다. 객체 지향 기법 Fig. 3에서 계승 개념을 이용하여 관심이 되는 문제가 2차원인 경우에는 2차원 절점을 이용하고, 3차원인 경우에는 3차원 절점을 이용하면 다른 부분의 수정

없이도 쉽게 유한 요소 해석을 확장 할 수 있다.

라. Material, Load, LoadCombination 클래스

구조 해석을 할 때 필요한 재료 상수, 구조물에 재하된 하중들의 크기, 방향, 종류 그리고 한 절점에 가해지는 하중 조합을 각각 나타내는 클래스이며, 이들은 단순히 자료를 저장하기만 한다. 재료와 하중, 하중 조합은 유한 요소 해석이 아니더라도 구조 해석에는 항상 이용되는 항목들이므로, 이들 클래스 역시 객체의 설계에 유한 요소 해석 절차가 반영되지 않도록 하여 이후에 다른 수치 해석 기법을 구현할 때 수정 없이 이용할 수 있도록 하였다.

마. Element 클래스

유한 요소 해석에서 요소는 그 종류에 상관없이 동일한 계산 순서를 따른다.

- 형상 함수의 계산
- 재료 성질, 절점의 기하학적 정보에 따라 형상 함수로부터 강성도 행렬 계산
- 하중 벡터 계산

여기에서 문제의 종류에 따라 변하는 부분은 형상 함수뿐으로 형상 함수를 별도의 클래스로 구현하여 요소의 형상 함수만을 수정하는 것으로 새로운 요소를 구현할 수 있도록 Fig. 4와 같이 Element class를 구현하였다. 이렇게 계산한 요소의 강성도 행렬들을 집적하여 전체 문제의 강성도 행렬을 구하게 된다.

바. Solver 클래스

유한 요소 해석의 일반식 $KU=R$을 해석하기 위한 클래스로 이는 행렬의 연산으로 생각하여 행렬 클래스의 메쏘드로 구현할 수도 있다. 그러나 유한 요소 해석에서 해석기가 차지하는 비중이 크고 방법도 다양하며 대상이 되는 문제에 따라 일반식과 해법 등이 달라지므로 별도의 객체로 구현하였다.

사. Conversion 클래스

유한 요소 해석에 의하여 얻어진 결과는 곧바로 사용되기도 하지만 결과의 분석을 위해서는 사용자가 요구하는 다른 값이나 그래픽 자료 등으로 변환되어야 할 필요도 있다. 이를 위해서는 결과를 변환하기 위한 Conversion 클래스가 필요하며 이는 문제의 전체 자유도와 같은 크기의 벡터를 인스턴스로 갖고 결과 값을 변환하는 메쏘드를 포함한다.

아. FEM 클래스

요소와 마찬가지로 유한 요소 해석 역시 대상 문제와 상관없이 일정한 절차를 따른다. 유한 요소 해석 절차는 다음과 같이 단순화할 수 있다.

- 자료의 입력
- 전체 강성도 행렬의 집적
- 전체 하중 벡터의 집적
- 연립 방정식 풀이
- 결과의 변환

위의 기본 절차에 필요한 정보는 클래스 사이의 메시지 교환만으로 관리할 수 있도록 I/O 클래스를 통하여 별도의 클래스에 자료를 입력·저장한다. 요소의 구성과 요소 강성도 행렬, 요소 하중 벡터의 계산은 Element 클래스에서 담당하

```
public abstract class Element{
/** ------------------------------------------------------------
**  Fields
** ------------------------------------------------------------ */
int                 noOfElement;
int                 noOfMaterial;
int[]               connectivity;

ShapeFunction       shapeFunction;

Matrix              localK;
Matrix              localR;

/** ------------------------------------------------------------
**  Constructor
** ------------------------------------------------------------ */
    Element(){  shapeFunction=new ShapeFunction(noOfDOF);
                calcLocalK(localK, shapeFunction);
                calcLocalR(localR, shapeFunction);
    }
}
```

Fig. 4. Finite element in FEA procedure(element class)

며 최후에 구성된 연립 방정식은 Solver 클래스로 해석한다. Fig. 4의 FEM 클래스는 유한 요소 해석의 진행을 관리하고 각 요소의 강성도 행렬과 하중 벡터를 집적하는 메쏘드를 갖는다. 한편 클래스 변수로 문제의 종류를 나타내는 상수를 정의하여 새로운 문제가 추가 될 때는 FEM 클래스를 계승하여 새로운 문제를 상수로 정의하기만 하면 되도록 하였다.

Ⅲ. 적용성 검토

OOFE_JAVA의 해석 결과의 정확성과 효율성의 검토를 위하여 판 요소를 이용하였다. 또, Java 언어에 적용된 가상 기계의 호환성을 확인하기 위하여 개발한 프로그램을 워크스테이션과 개인용 컴퓨터에서 비교 운용하여 보았으며, OOFE_JAVA를 구성하고 있는 객체들의 이용성을 알아보기 위하여 유한 요소 해석 이외의 프로그램에 이들 객체들의 적용 방법에 대하여 검토하였다.

1. OOFE_JAVA의 적용성 검토

판의 휨 해석은 유한 요소 해석의 발전과 함께 많은 연구가 진행되어, 많은 종류의 판 요소가 개발되어 있다. 이는 판의 휨 해석이 현실의 문제를 해석하는 기초임에도 그 해석이 쉽지 않기 때문이다.[4] 본 연구에서는 지금까지 개발 되었던 여러 판 요소중 비교적 간단하면서도 얇은 판의 경우에 잘 맞는 것으로 알려진 MZC 판요소를 적용 예로 하여 개발된 OOFE_JAVA의 적용성을 검토하였다.

가. 판의 휨 요소에 필요한 클래스의 정의

판 휨의 가장 기본적인 형태인 사각형 요소를 구현하였다. Fig. 5에서 보인 요소는 Melosh, Zienkiewicz와 Cheung이 처음으로 개발하였기 때문에 MZC 사각형이라고 불린다.

이 요소는 각 변에 수직선-기울기 양립성(normal-slope compatibility)을 갖지 않는 비등각(nonconforming)요소이다.[2,4] 이러한 형태의 다른 요소와 마찬가지로 이 요소도 z방향의 변위만 갖는다. 따라서,

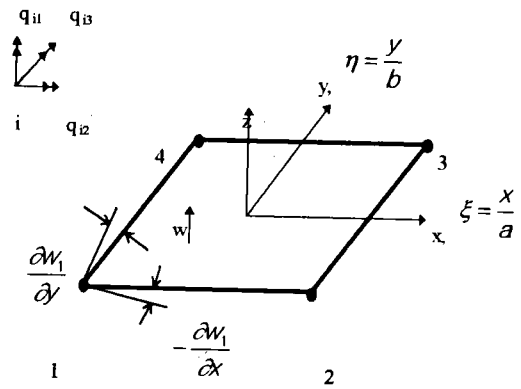

Fig. 5. MZC Plate-bending element

$$\mathbf{u}=\{w_i\} \quad (i=1,2,3,4) \cdots\cdots (1)$$

이고, 절점 변위는 다음과 같다.

$$q_i=\{q_{i1}q_{i2}q_{i3}\}=\left\{w_i \frac{\partial w_i}{\partial y} - \frac{\partial w_i}{\partial x}\right\}, (i=1,2,3,4) \cdots (2)$$

이 요소의 가상 변위 함수는

$$w=c_1+c_2\xi+c_2\eta+c_4\xi^2+c_5\xi\eta+c_6\eta^2+c_7\xi^3 + c_8\xi^2\eta+c_9\xi\eta^2+c_{10}\eta^3+c_{11}\xi^3\eta+c_{12}\xi\eta^2 \cdots (3)$$

이로부터 다음의 변위 형상 함수를 유도할 수 있다.

$$\mathbf{f}_i=\{f_{i1}f_{i2}f_{i3}\} \cdots\cdots (4)$$

여기서,

$$f_{i1}=\frac{1}{8}(1+\xi_0)(1+\eta_0)(2+\xi_0+\eta_0-\xi^2-\eta^2)$$

$$f_{i2}=-\frac{1}{8}b\eta_i(1+\xi_0)(1-\eta_0)(1+\eta_0)^2$$

$$f_{i3}=\frac{1}{8}a\xi_i(1-\xi_0)(1+\eta_0)(1+\eta_0)^2$$

또한

$$\xi_0=\xi_i\xi\ \eta_0=\eta_i\eta\ \ (i=1,2,3,4)$$

이며 곡률의 정의로 부터 선형 미분 연산자 $\bar{\mathbf{d}}$는

$$\bar{\mathbf{d}}=\left\{\frac{\partial^2}{\partial x^2},\ \frac{\partial^2}{\partial y^2},\ 2\frac{\partial^2}{\partial x\partial y}\right\} \quad \cdots\cdots\cdots\cdots\cdots\cdots (5)$$

이다. 따라서 일반 변형율-변위 행렬 $\bar{\mathbf{B}}$는 다음과 같이 쓸 수 있다.

$$\bar{\mathbf{B}}_i=\bar{\mathbf{d}}\mathbf{f}_i=\begin{bmatrix} \frac{\partial^2 f_{i1}}{\partial x^2} & \frac{\partial^2 f_{i2}}{\partial x^2} & \frac{\partial^2 f_{i3}}{\partial x^2} \\ \frac{\partial^2 f_{i1}}{\partial y^2} & \frac{\partial^2 f_{i2}}{\partial y^2} & \frac{\partial^2 f_{i3}}{\partial y^2} \\ 2\frac{\partial^2 f_{i1}}{\partial x\partial y} & 2\frac{\partial^2 f_{i2}}{\partial x\partial y} & 2\frac{\partial^2 f_{i3}}{\partial x\partial y} \end{bmatrix} \quad \cdots\cdots (6)$$

이를 인스턴스로 갖는 판의 휨 요소는 Fig. 6과 같이 Element 클래스를 계승하여 구현하였다.

나. 프로그램의 진행

프로그램이 수행되는 동안에 프로그램이 사용하는 클래스의 계층도는 Fig. 7과 같다.

다. 해석 결과 검토

본 연구를 통하여 개발한 OOFE_JAVA을 이용하여 네 면이 모두 고정된 판에 등분포 하중이 가해진 경우를 Reddy의 결과와 비교하면 Table 2와 같다.

이 결과는 Reddy[12]가 제시한 판의 폭과 높이

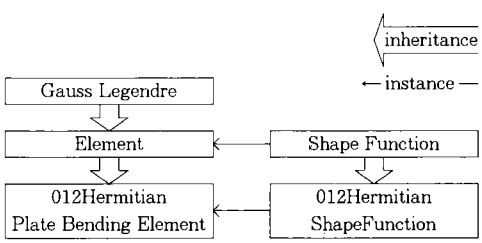

Fig. 6. Plate bending element class hierarchy

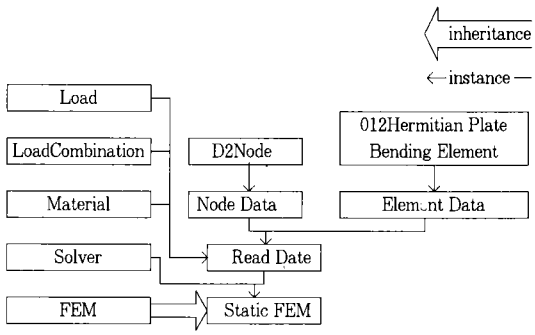

Fig. 7. Static FEM class hierarchy

Table 2. Comparison of the Analysis Result

a/h	2×2quadratic element by Reddy	Exact by Reddy	OOFEA
50	4.587	4.579	4.583
100	4.580	4.572	4.577

의 비, $\frac{a}{h}$가 100인 경우의 무차원 변위 w로 변환한 것이다. 표에서 알 수 있는 것처럼 OOFE_JAVA 정해에 근사한 해를 주는 것을 알 수 있었다.

2. OOFE_JAVA의 호환성 검토

Java 언어로 작성된 프로그램은 간단한 기계어 코드로 컴파일된다. 이 기계어 코드는 Java의 실행 해석기에 의하여 수행되는데, Java 실행 해석기는 이론상 모든 종류의 컴퓨터에 설치될 수 있다. 따라서 Java로 개발되어 한 번 컴파일된 프로그램은 Java 모든 컴퓨터에서 실행이 가능하다. 현재는 Solaris™ 운영체제를 사용하는 워크스테이션과 Windows95/NT를 사용하는 IBM 호환 컴퓨터, Macintosh™에서 사용 가능한 실행 해석기가 발표되어 있다.

본 연구에서는 가상 기계의 호환성을 검토하기 위하여 Solaris™에서 개발한 OOFE_JAVA를 워크스테이션과 Windows95가 설치된 개인용 컴퓨터에서 비교 운용하여 보았다. 프로그램의 실행에 사용된 컴퓨터의 사양은 Table 3과 같다.

워크스테이션에서 컴파일한 이진 코드를 그대

Table 3. Specification of computers

	Sun/Solaris	Axil/Solaris	Intel/Windows95
CPU	SuperSparc/ 60MHz	UltraSparc/ 167Mhz	Pentium/ 166MHz
CPU Memory	64 Mbyte	35 Mbyte	32 Mbyte

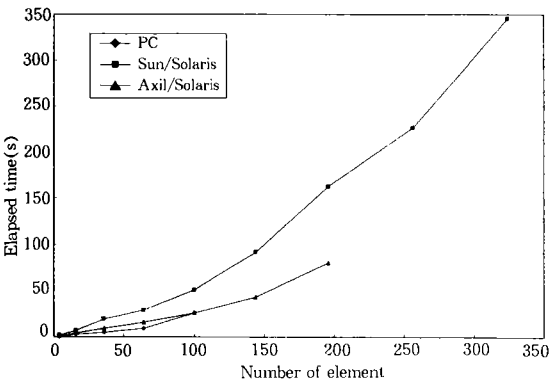

Fig. 8. The comparison of the performance of OOFE-JAVA

로 PC에서 수행하였을 때 양 쪽에서 같은 결과를 얻을 수 있었으며, 문제의 크기를 늘려가며 계산을 수행하였을 때 워크스테이션과 개인용 컴퓨터에서의 수행 시간은 Fig. 8과 같다.

Fig. 8에서 알 수 있는 바와 같이 수행 속도는 실험에 사용된 워크스테이션보다 PC에서 보다 빠른 것을 알 수 있다. 실험에 사용한 워크스테이션의 성능을 감안할 때 같은 빠르기의 연산장치를 사용하면 거의 같은 수행 속도를 나타낼 것임을 알 수 있다. 한편 PC에서는 요소의 수가 100개가 넘으면 메모리 공간의 부족으로 해석이 불가능하였으나 워크스테이션에서는 실제 메모리 공간 이외에 가상 메모리를 사용하여 훨씬 큰 문제도 해석할 수 있었다.

3. 개발된 객체의 이용성 검토

가. 유한차분법에 이용된 예

$$k\frac{\partial^2 T}{\partial x^2}=\frac{\partial T}{\partial t} \quad \cdots\cdots (7)$$

식 (7)으로 표현되는 1차원의 열전도 방정식에 대표적인 음함수법인 Crank-Nicolson 법을 적용하면 우변은

$$\frac{\partial T}{\partial t}\simeq\frac{T_i^{l+1}-T_i^l}{\Delta t} \quad \cdots\cdots (8)$$

이고, 좌변은

$$\frac{\partial^2 T}{\partial x^2}\simeq\frac{1}{2}\left[\frac{T_{i+1}^l-2T_i^l+T_{i-1}^l}{(\Delta x)^2}+\frac{T_{i+1}^{l+1}-2T_i^{l+1}+T_{i-1}^{l+1}}{(\Delta x)^2}\right] \quad \cdots\cdots (9)$$

이다. 여기서 i는 공간을 표시하는 격자점이고, l은 시간의 진행을 표시하는 격자점이다. 이 두 식을 식 (7)에 대입하여 정리하면

$$-\lambda T_{i-1}^{l+1}+2(1+\lambda)T_i^{l+1}-\lambda T_{i-1}^{l+1}$$
$$=\lambda T_{i-1}^l+2(1-\lambda)T_i^l+\lambda T_{i+1}^l \quad \cdots\cdots (10)$$

이 된다. 여기서, $\lambda=k\Delta t/(\Delta x)^2$이다. 위의 식을 공간의 격자점에 따라 정리하면 삼중대각 행렬로 표현되는 다음의 방정식이 된다.

$$[\mathbf{K}]\{\mathbf{T}\}=\{\mathbf{T}_0\} \quad \cdots\cdots (11)$$

이 식의 행렬과 벡터는 본 연구에서 개발한

```
class CrankNicolson{
/** ------------------------------------------------------------
** Fields
** ------------------------------------------------------------ */
Matrix K, T0;
 Solver T;
/** ------------------------------------------------------------
** Constructor
** ------------------------------------------------------------ */
    CrankNicolson(){
            matrix K=new Matrix(noOfNodes, noOfNodes);
            matrix T0=new Matrix(noOfNodes, 1);
            getInitialValues(T0);
            calcMatrix(K);
            T=new Solver(T, T0, noOfNodes);
    }
}
```

Fig. 9. Crank Nicolson class using matrix class

Matrix 클래스로 표현할 수 있으며, 방정식은 Solver Class를 이용하여 Fig. 9와 같이 Crank Nicolson 클래스를 구현하여 해결 할 수 있다.

나. 그래픽 표현에 이용된 경우

Baumgartner가 제안한 2차원 유한 요소 해석을 위한 Winged-Edge 자료 구조는 Edge, Vertex, Face의 세 자료 구조로 구성되어 있다.[14] 이들 가운데 Winged-Edge 자료 구조의 기본이 되는 Edge는 Fig. 10과 같이 구성된다.

여기서 가장 기본이 되는 Vertex는 Fig. 11과 같은 구성을 갖고 있다.

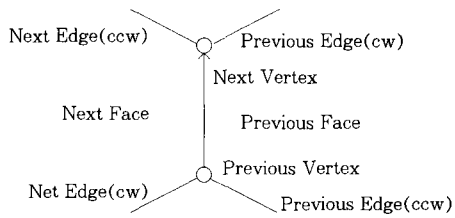

Fig. 10. The local topology for an edge in the edge-based data structure

Next Vertex	Previous Vertex
Previous Linked Vertex	
X coordinate	
Y coordinate	

Fig. 11. Data structure record organization of vertex

```
Vertex extend D2Node{
/** ------------------------------------------------------------
** Fields
** ------------------------------------------------------------ */
    //Geometric coordinate of x and y is inherited
    Vertex next, previous;
    Vertex previousLink;
/** ------------------------------------------------------------
** Constructor
** ------------------------------------------------------------ */
    Vertex(){
            assignPreviousVertex();
            assignNextVertex();
            assignPreviousEdge();
            assignCoordinate(x, y);
    }
}
```

Fig. 12. 2-Dimensional node used in winged-edge data structure extending D2Node

본 연구에서 개발된 2차원 절점을 표현하는 D2Node 클래스를 확장하여 Fig. 12와 같이 구현할 수 있다.

이상과 같이 본 연구를 통하여 개발된 OOFE_JAVA의 객체들은 수정 없이 다른 프로그램의 개발에 이용될 수 있어 객체 지향 기법의 장점을 살려 프로그램 개발 기간을 단축시킬 수 있음을 알 수 있었다.

V. 결 론

실제로 이용될 수 있는 유한 요소 해석 프로그램을 개발하기 위해서는 현재의 빠르게 발전하는 기술에 대응할 수 있도록 절차적 프로그래밍 언어 이외의 패러다임을 이용하여야하며, 이러한 대안으로 객체 지향 언어를 이용한 유한 요소 해석 프로그램의 개발에 관한 연구가 진행되어 왔다. 그러나 지금까지의 대부분의 연구는 객체 지향 프로그램의 개념적인 구현과 그 발전 가능성을 검토하는 데에만 그친 경향이 있었다.

본 연구에서는 지금까지의 객체 지향 프로그램 기법의 연구에 대한 보완과 확장을 통하여 객체 지향 언어인 Java로 객체지향 유한 요소 해석 프로그램인 OOFE_JAVA를 개발하고, 현실 문제에대한 적용성을 평가해 본 결과 다음과 같은 결론을 얻었다.

1. 적절히 설계된 객체 지향형 유한 요소 해석 프로그램의 객체들은 수정 없이 유사한 수치 해석 방법이나 공학적 용도에 적용할 수 있음을 확인하였다.

2. Java 언어로 제작된 OOFE_JAVA는 여러 종류의 컴퓨터에서 조작 없이 운영할 수 있었으며, 개인용 컴퓨터에서도 충분한 수행 능력을 보여주었으므로 개인용 컴퓨터와 워크스테이션이 혼합된 분산처리 환경의 지원이 가능함을 알 수 있었다.

3. OOFE_JAVA의 개발 과정 중에서 Java

언어는 유한 요소 해석 프로그램처럼 구조화된 문제의 해석에도 적합하고, 객체의 추가와 변경 및 확장이 용이하여 대규모 시스템 개발에 유리할 것으로 판단되었다.

참 고 문 헌

1. Arnold, Ken and James Gosling, The JavaTM Programming Language, Addison-Wesley Publishing Company, Inc., 1996.
2. Cook, Robert D, et al., Concepts and Applications of Finite Element Analysis, John Wiley & Sons, pp. 314-334, 1989.
3. Forde, Bruce W. R., et al., Object-Oriented Finite Element Analysis, Computers & Structures, Vol.34, No. 3, pp. 355-374, 1990.
4. Hrabok, M. M. and T. M. Hrudey, A Review and Catalogue of Plate Bending Finite Elements, Computers & Structures, Vol 19, No. 3, pp. 479-495, 1984.
5. Kong, X. A., and D. P. Chen, An Object-Oriented Design of FEM Programs, Computers & Structures, Vol.57, No. 1, pp. 157-166, 1995.
6. Lee, H. H. and J. S. Arora, Object-Oriented Programming for Engineering Applications, Engineering with Computers Vol. 7, No. 4, pp. 225-235, 1991.
7. Mackie, R. I., Object Oriented Programming of The Finite Element Method, International Journal for Numerical Mehotds in Engineering, Vol. 35, pp. 425-436, 1992.
8. Mackie, R. I., Using Objects to Handle Complexity in Finite Element Software, Engineering with Computers, Vol. 13 No. 2 pp. 99-111, 1997.
9. Miller, G. R., An Object-Oriented Approach to Structural Analysis and Design, Computers & Structures, Vol. 40, No. 1, pp. 75-82, 1991.
10. Mohtar, R. H. et al., Selected Topics in Numerical Analysis A Tutorial Set, Volume 1, ASAE, 1996.
11. Niemeyer, Patrick and Joshua Peck, Exploring JAVA, O'Reilly & Associates, Inc, 1996.
12. Reddy, J.N., An Introduction To The Finite Element Method, McGraw-Hill, Inc, 1993.
13. Scholz, S. P., Elements of An Object-Oriented FEM++ Program In C++, Computers & Structures, Vol.43, No.3, pp. 517-529, 1992.
14. Wawrzynek, Paul A. and Anthony R. Ingraffea, An Edge-Based Data Structure for Two-Dimensional Finite Element Analysis, Engineering with Computers, Vol. 3, No. 1, pp. 13-20, 1987.
15. Zimmermann, Thomas, et al., Object-oriented finite element programming : I.Governing principles, Computer Methods in Applied Mechanics and Engineering, Vol. 98, pp. 291-303, 1992.
16. 신영식 외, PC용 객체지향 구조해석 프로그램의 개발, 전산구조공학회지, 제5권, 제4호, pp. 125-132, 1992.