ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

IARJSET

International Advanced Research Journal in Science, Engineering and Technology

National Conference on Emerging Trends in Engineering and Technology (NCETET'16)

Lourdes Matha College of Science & Technology, Thiruvananthapuram

Vol. 3, Special Issue 3, August 2016

Object Oriented Paradigm for Programming Finite Element Analysis

T.J. Raj Thilak[1], P.V. Anil Kumar[2]

Sci/Eng-SE, Structural Engineering Entity, VSSC, ISRO, Thiruvananthapuram, Kerala, India[1]

Sci/Eng-SG, Structural Engineering Entity, VSSC, ISRO, Thiruvananthapuram, Kerala, India[2]

Abstract: Finite element structural analysis program typically requires a huge amount of data to be managed and number of lines of the code runs to millions. If the architecture of the program is not properly designed, even a small change in the program will lead to a humungous task. Earlier procedural programming methods were used to code the software which accesses global data in the entire program. This paper highlights the need of an object oriented approach and its advantages for developing a finite element software.

Keywords: Finite Element Analysis; Object oriented programming; FEAST$^{SMT©}$.

I. INTRODUCTION

Finite element method is an inevitable tool used extensively for the analysis of structures to reduce the cost of testing. A typical finite element programming involves millions of lines to be coded. Most of the finite element programs are written in FORTRAN, a procedural programming language. The code contains a lot of data structures that are declared in common block and are usually accessed throughout the program. This global access reduces the flexibility of the program and increases the complexity in modifying the program. As the number of lines in the code increases, new users will find difficulty in introducing new modules and a small change in the code needs a complete understanding of the entire code [1]. Since a finite element program needs undergoes continuous modifications to enhance its capabilities, it is better to adopt object oriented methodologies that have the following advantages [2]

- Huge amount of data can be handled in a secured manner
- Data can be accessed and manipulated without affecting the integrity
- Processes can be changed easily without affecting other processes, even if change of data structure is required
- New modules can be added with minimal effort

This paper deals with a new object oriented approach of finite element software FEAST$^{SMT©}$ an in-house finite element analysis software being developed by VSSC/ISRO. The software is developed in C++ with object oriented paradigm. To explain the architecture of the program, element class has been taken as example and elaborated. Similarly other classes can also be extended to suit the requirement.

To reduce the solution time, the solver uses sub-structuring technique in the finite element domain coupled with multi threading technique to attain system level parallelisation.

II. PROGRAM ARCHITECTURE

The application of object-oriented design has proven to be very beneficial to the development of flexible programs. The basis of object-oriented design is abstraction. The object-oriented philosophy abstracts out the essential immutable qualities of the components of the finite element method into classes of objects. Objects store both their data, and the operators on the data that may be used by other objects. This abstraction forms a stable class definition in which the relationship between the objects is explicitly defined. The implicit reliance on another component's data does not occur. Thus, the design can be extended with minimal effort. The abstraction of data into classes of objects limits the knowledge of the system required to work on the code, to only the class of interest. Encapsulating the data and the operations together isolates the classes and promotes reuse of code. Changes to a class affect only the class under consideration. There is no ripple effect. Interdependencies between the classes are explicitly laid out in the class interfaces and are easily determined. The object-oriented languages inherently ensure data integrity through restricted access mechanisms. This modular architecture provides a flexible and extensible set of objects that facilitate a faster development without sacrificing the performance of the program.

FEAST$^{SMT©}$ uses sub structuring techniques for its analysis capabilities. Sub-structuring is a well-established method in finite element technology. In this method the complete structure is subdivided into number of substructures called super elements. For each substructure the nodes are distinguished as internal or external nodes according to the

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
National Conference on Emerging Trends in Engineering and Technology (NCETET'16)
Lourdes Matha College of Science & Technology, Thiruvananthapuram
Vol. 3, Special Issue 3, August 2016

position of the node in the substructure. External nodes are those common nodes at the interface between multiple substructures as shown in Fig. 1. All other nodes in a substructure are designated as internal nodes.

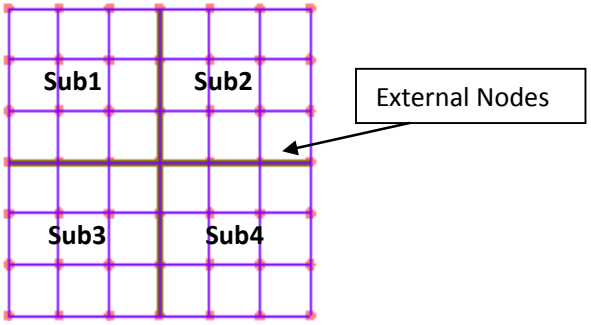

Fig. 1 Substructures and external nodes

After assembling the stiffness matrices for all the substructures, the internal degrees of freedom are condensed out and the stiffness matrix corresponding to the external degree of freedom are assembled and solved for external displacements. The external displacements are used to calculate the internal displacements of each substructure. Most of the computations associated with the substructure, like, node renumbering, assembly, internal variable evaluation etc. can be performed independent of other substructures (Fig 2). The sub-structuring approach is advantageous due to the following reasons:

1. It splits the work involved in the calculation process into several independent and discrete packages enabling parallel processing
2. Much of the work carried out for any given substructure can be used again in later calculations.
3. It reduces the amount of memory required to solve the model.

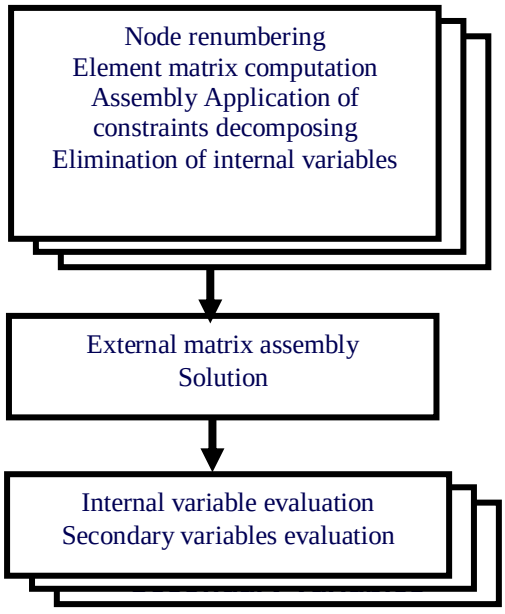

Fig. 2 Substructure based computational scheme

Multi threading is a kind of parallel processing that can be used on single processor machines such as a typical desktop, especially with a processor having multi-core computing capabilities. The computations associated with each substructure, as shown in fig.2 are assigned to separate threads and the entire tasks can be carried out without giving idle time for the processor, and it can be used to increase the interactivity of the program. Since the operating system assigns time slices to different activities like computations of substructures and user operations, the user can continuously interact with the program and the substructure calculations are carried out in the background of the program in different threads. Thus, multi threading offers the advantage of carrying out the task very quickly with greater interactivity.

III. DERIVATION OF ELEMENT STIFFNESS MATRIX

A solid continuum is discretised into simple geometrical domains known as elements, where the governing differential equations are solved. The solution is sought at specified discrete locations within the element extent, known as nodes, in most cases they are at the element boundaries. The discrete solutions from various nodes of the element are used for interpolating the dependent variable at any arbitrary location within the element. In this section, a general description of the displacement method is given [3]. Principle of minimum potential energy states that "Of all possible displacement states a body can assume that satisfy compatibility and specified kinematic conditions, the state that satisfies the equilibrium equations makes the potential energy assume a minimum value".
For a linearly elastic body with conservative loads, the expression for potential energy is

$$\prod = U + \Omega \qquad (1)$$

Where, U is the strain-energy and Ω is potential of all external loads. Since the strain energy depends on the strain and stress state of the system in turn it depends on the degree of freedom of the system. Thus the potential energy is a function of the degrees of freedom q_i. Using principle of stationary potential energy, where equilibrium prevails when $d\prod = 0$, is expressed as

$$\frac{\partial \prod}{\partial q_i} = 0 \qquad \text{for } i = 1,2,....n \qquad (2)$$

Utilizing equation (2), n equations solved for obtaining values of degrees of freedom. The general form of equation (1) is

$$\prod = \int_V \left(\frac{1}{2} \{\varepsilon\}^T [D] \{\varepsilon\} - \{\varepsilon\}^T [D] \{\varepsilon_0\} + \{\varepsilon\}^T [\sigma_0] \right) dv - \int_V \{u\}^T \{\Gamma\} dv - \int_S \{u\}^T \{\Phi\} ds - \{q\}^T P \qquad (3)$$

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
National Conference on Emerging Trends in Engineering and Technology (NCETET'16)
Lourdes Matha College of Science & Technology, Thiruvananthapuram
Vol. 3, Special Issue 3, August 2016

The terms in (3) corresponds to the strain energy stored in the system and work done by the body forces Γ, surface tractions Φ and point load P respectively on the system. ε is the strain , σ is the stress, D is the elasticity matrix, ε_0 is the initial strain and σ_0 is the initial stress. By using the constitutive equation and strain displacement relation and applying (2) we will get the standard form of finite element equation as below

$$[K]\{q\}=\{F\} \qquad (4)$$

where,

$$[K]=\int_V [B]^T [D][B]dv$$

$$\{F\}=\int_V [N]^T \{\Gamma\}dv+\int_S [N]^T \{\Phi\}ds+\int_V [B]^T [D]\{\varepsilon_0\}dv$$

$$-\int_V [B]^T \{\sigma_0\}dv+P$$

where, B is the strain displacement relation and N is the shape function used to interpolate the field variables anywhere within the element in terms of nodal field variables.

IV. PROGRAMMING ELEMENT STIFFNESS MATRIX

In the previous section a general derivation of the element stiffness matrix is derived independent of element type. In a procedural programming, the element stiffness matrix has to be coded for each and every element type, increasing the number of lines and thus increasing the complexity of the code. In an object oriented approach, features like encapsulation, inheritance, polymorphism etc. are used to concisely write the code. In the present software, a base class CElement is written which contains the general framework of the element. The main purpose of this class is to serve the basic common services, which are common to all finite elements (like storing the references to element's material model and physical attributes, nodes, etc.). The derived classes (direct children of Element class) are assumed to provide general services required for a specific analysis purpose - like evaluation of stiffness, geometric stiffness and mass matrices for structural analysis, or evaluation of capacity and conductivity matrices for heat transfer analysis. The common operations are handled by classes at the higher levels in the hierarchy.

Those operations dependant on the dimension of elements, like numerical integration and shape function evaluation etc., are handled in CElement1d, CElement2d or CElement3d. The class diagram for the elements module of FEAST$^{SMT©}$ program is as shown in Fig.3. Fig.4 shows a function for calculating the stiffness matrix. The functions to be overridden are shown in bold.

For defining a new element type, derive it from a class already in the hierarchy and add the missing functionalities or override the existing functionality respecting the interface. Some important code segments in the case of defining an axisymmetric solid of revolution element is described below. The class CAxisymmetricElement has to be derived from CElement2d in order to utilise the functions defined in that class. Axisymmetric element reuses the services provided by CElement for computation of stiffness matrix.

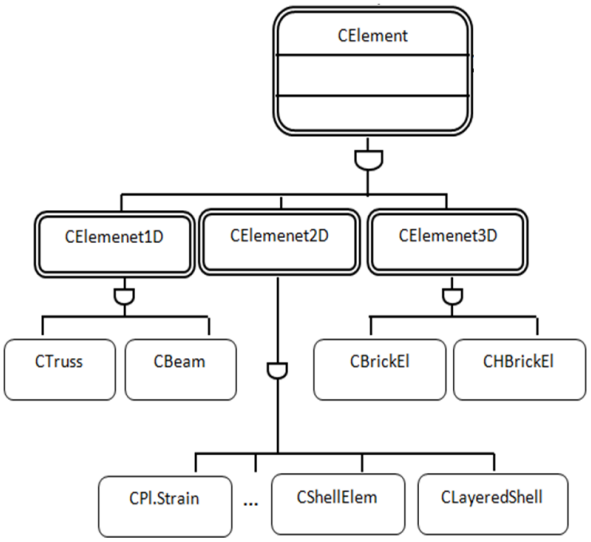

Fig. 3 Class diagram for introducing a new element type

An element derived from CElement is assumed to provide the following services if the implementation already provided in its class hierarchy is not acceptable.

```
bool CElement::GetStiffness(CSymmetricMatrix &S)
{
        S.Init();

        GP gp[30];
        int ng = GetGaussPoints(gp);

        CSymmetricMatrix D;
        for(int i=0; i<ng; i++) {
            CMatrix B;
            double detJ =ComputeBMatrix(gp[i], B);

            m_pm->GetMaterialMatrixAt(this, gp[i], D);

            CMatrix DB;
            DB.SetProductOf(D, B);

            double dVol = gp[i].w *  detJ*GetVolumeFactor(gp);

            S .AddTProductOf(B, DB, dVol);
        }
        return true;
}
```

Fig. 4 Function to calculate Stiffness Matrix

• **GetGaussPoints** – Returns a list of Gauss Integration points (containing Guass absissae and weights) depending

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

National Conference on Emerging Trends in Engineering and Technology (NCETET'16)

Lourdes Matha College of Science & Technology, Thiruvananthapuram

Vol. 3, Special Issue 3, August 2016

on integration rule. This function is implemented in CELement2D class.

• **ComputeBMatrix** – Computes strain-displacement matrix. This function is implemented in the respective element type classes.

• **GetMaterialMatrixAt** – Obtain the material constitutive matrix as provided by the referenced material. This function is implemented in the CMaterial class. Depending on the material type and element type, CElement class will take the constitutive matrix.

In order to reuse the service for stiffness-matrix computation CAxisymmetricElement has only to provide ComputeBMatrix function that depends on its strain displacement relation.

V. RESULTS AND DISCUSSION

To show the efficiency of the substructuring and multithreaded architecture, a pressure vessel as shown in Fig.5 is modeled with 4 node shell element. The model is executed in a personal computer having 3GB RAM and the execution time is shown in Table 1. The number of degree of freedoms and elements in the model are respectively 173508 and 28800. Any new element to be added has to over ride certain functions that are already coded in CElement class.

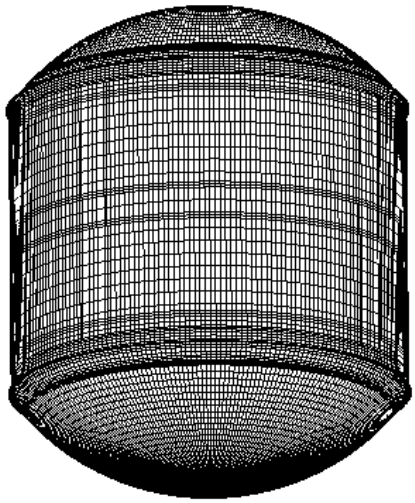

Fig. 5 Pressure vessel discretised with 4 node shell element

Table 1 Computational Timings for different number of substructures

No. of Substructures	Solution Time (Sec.)
4	158.94
6	63.83
8	70.89
12	58.1

ACKNOWLEDGMENT

The author would like to acknowledge **Dr. T. Sundararajan,** Head, SMSD, **Shri. P. Balachandran,** GD, SDMG, and **Dr. S. Unnikrishnan Nair**, DD, VSSC (STR) for providing necessary support in the development of the software and team FEAST for writing this manuscript.

REFERENCES

[1] G.C.Archer, G.Fenves, C. Thewalt, " A new object-oriented finite element analysis program architecture," Computers and Structures, vol.70, pp.63-75, 1999.

[2] P.V.Anil Kumar, T.J.Raj Thilak, Dr.B.Sivasubramonian, Shri. K.L.Handoo, " Development of an Object -Oriented Pre and Post Processing Software for Finite Element Analysis of Structures", ICCAE 2007, 2007.

[3] Robert D.Cook, David S.Malkus, Michael E. Plesha, Robert J. Witt, " Concepts and Applications of Finite Elemet Analysis - Fourth edition", Wiley India (P.) Ltd.