FE2000: An Object-oriented Framework for Parallel Nonlinear Dynamic Finite Element Analysis

Y. S. Yang and S. H. Hsieh
Department of Civil Engineering, National Taiwan University, Taipei 10617, Taiwan
yuansen@caece.net, shhsieh@caece.net

Abstract

In this work, an object-oriented framework FE2000 for parallel large-scale nonlinear dynamic finite element analysis is presented. The parallel substructure method is employed for parallel finite element analysis. An extension of the general sparse matrix package SPARSPAK, an iterative mesh partitioning approach, and a multi-level substructure method are employed and integrated. Several large-scale finite element models are used to demonstrate the performance of FE2000 on commonly available personal computers. In the numerical experiments, even for finite element models with up to tens or hundreds thousand degrees-of-freedom, each nonlinear iteration of analysis can be completed within several minutes. Namely, a complete nonlinear dynamic analysis with totally hundreds of nonlinear iterations may be accomplished within a reasonable elapsed time, say, an overnight time.

Introduction

In the past, two object-oriented finite element frameworks: FE++ (Lu, 1994) and PFE++ (Mukunda et al., 1998) have been developed for sequential and parallel finite element analysis, respectively. However, our experiments show that both of them do not perform well enough in terms of efficiency. For further improving the efficiency of large-scale finite element analysis, an object-oriented framework FE2000 is developed in this work for parallel finite element analysis using the substructure method. The major issues addressed for FE2000 are:

(1) Performance: FE2000 is designed for large-scale structural analysis, which requires considerable computations. FE2000 is supposed to complete a finite element analysis as soon as possible. In FE2000, the general sparse matrix technique is employed to reduce the elapsed time on solving the equilibrium equation, which is usually the most time consuming part in finite element analysis.

(2) Maintainability: since FE2000 is designed for researches on finite element analysis, good maintainability is an important requirement for further tailoring design and implementation decisions. Based on the object-oriented design, the objects in FE2000 are easy to use, manipulate, and modify.

(3) Portability: FE2000 is supposed to be performed on various types of computers, especially for message-passing parallel computers. The ANSI C++ programming language, which is one of the widely used programming languages, is selected for implementing FE2000. In addition, the MPI standard (Message Passing Interface...
Forum, 1994), which is widely supported for message-passing parallel computers. Therefore, FE2000 can be easily ported to the parallel computer systems which support the MPI standard.

**Key Classes in FE2000**

Three key classes – feNode, feElement and feAssemblage are implemented in FE2000 to perform the basic finite element analysis, which store and manipulate the basic information of the nodes, elements and the entire finite element model, respectively. These classes are described in more details.

The feNode class provides accessory interfaces to the node properties in finite element analysis, including nodal coordinates (corresponding to X, Y and Z axes), six degrees-of-freedom, six nodal forces, six geometric boundary conditions (corresponding to three displacements and three rotations). It also includes nodal properties for dynamic and nonlinear analysis, such as velocities and accelerations, reference axes in C1 and C2 configurations (using Update Laguange method, Yang and Kuo (1994)), and increment displacements. For the sake of efficiency, the feNode is designed as a concrete class instead of an abstract class (see Stroustrup, 1997), and currently has no derived class.

The feElement class provides accessory interfaces to the element properties in finite element analysis, including adjacent nodes, element property matrices (including stiffness, mass, and damping matrices), internal forces, etc. Because different types of elements have different ways of forming the property matrices and calculating internal forces or stresses, and have different additional element properties, it is difficult to manipulate various types of element by using a single class. The feElement class is designed as an abstract class, from which other classes of elements are derived. At present, feB8Element (8-noded brick solid element), feB20Element (20-noded brick solid element), feTrussElement (3D truss element) and feBCEElement (3D beam-column element) are implemented in FE2000. The feB20Element and feBCEElement support the nonlinear and dynamic analysis. The hierarchy of these element classes is shown in Fig. 1. The arrow lines between two classes denote ‘derived from.’

![Figure 1 Hierarchy of the element classes in FE2000](image-url)
The feAssemblage class provides accessory interfaces to the global system of finite element analysis, including pointers to the feNode and feElement objects. It also performs the assembling of the global stiffness matrix, force vector, and other details (e.g., numbering the degrees-of-freedom.) A sketch of the relationships among the feAssemblage, feNode, and feElement objects of a finite element analysis example is shown in Fig. 2. The arrow lines between two objects denote ‘has a pointer or reference to.’

Figure 2 Object relationship in a finite element analysis example using FE2000

**Matrix Libraries**

Matrix computation is one of the most important computing in finite element analysis, especially on solving the equilibrium equations. In FE2000, the general sparse matrix package SPARSPAK (George and Liu, 1981) is employed and extended so that it not only solves a linear system but also performs matrix condensation (for substructure analysis). Nevertheless, the extended SPARSPAK is not embedded in FE2000. FE2000 provides an interface for connecting with the extended SPARSPAK. It is easy to implement other matrix interface for connecting with other matrix libraries. At present, the interfaces for connecting with general sparse matrix packages SPOOLES (Ashcraft et al., 1999) PETSc (Balay et al., 1997) are built, so that the equilibrium equations can be solved in parallel using either direct or iterative methods. At present, the SPOOLES and PETSc are only used for linear static analysis, and do not support the substructure analysis.
Mesh Partitioning and Substructure Hierarchy

Mesh partitioning is the first step for the substructure analysis. In this work, the mesh partitioning environment MPE++ (Hsieh et al., 1998) is used to produce the initial mesh partitioning result (Hsieh et al., 1995). MPE++ is an object-oriented environment for embedding existing partitioning approaches, and designing and implementing new mesh partitioning approaches. At present, MPE++ includes the GR method (Farhat, 1988), RST method, METIS method (Karypis and Kumar, 1998), and HYT (Hsieh-Yang-Tsai) approach (Hsieh et al., 1999). After the MPE++ produces the initial mesh partitioning result, the IHYT (Improved Hsieh-Yang-Tsai) iterative mesh partitioning approach (Yang, 2000) is then iteratively tuning the mesh partitioning result for better load balance among substructures. At present, the IHYT approach is implemented in FE2000, however, with little modification, it is easy to be separated as an independent mesh partitioning package.

For the substructure analysis, a feSuperElement class is designed and derived from the feElement class. The feSuperElement class can be seen as an assemblage of a number of feNode and feElement objects, therefore, is very similar to the feAssemblage. However, a feSuperElement is in fact a type of element, which provides its own stiffness (or damping and mass) matrix and load vector for the feAssemblage object by assembling the element matrices and vectors, and then condensing the internal degrees-of-freedom to interface ones. Multi-level substructuring is allowed in FE2000. That is, a feSuperElement may contain several feSuperElement objects.

For the parallel substructure analysis, pfePAssemblage and pfePSuperElement classes are designed and derived from the feAssemblage and feSuperElement objects, respectively. The only difference between pfePAssemblage/pfePSuperElement and feAssemblage/feSuperElement is the former ones allows their elements contained by other processors. That is, the stiffness (or damping and mass) matrix and load vector of their elements may be computed by other processors, and then transferred back via MPI protocols.

Yang and Hsieh
Numerical Experiments

In this work, four finite element models are used to show the parallel efficiency of nonlinear dynamic finite element analysis using FE2000. Four computing nodes of a PC cluster CAPS (Cost-effective scAlable Parallel System) (Hsieh et al., 2000) are used for the hardware environment, each of which contains a Pentium II 350 processor and at least 256 MB main memory. A 100-Mbps network system is used as the communication facilities among the computing nodes. The time step is 0.01 seconds and 0.2 seconds of time are performed. Table 1 shows the element types, number of degrees-of-freedom, total number of nonlinear iterations (column (A)), and the total elapsed time for sequential nonlinear dynamic finite element analysis (column (B)). It can be seen that, for a large-scale structural model, only 0.2-second nonlinear dynamic analysis of may spend hours of time. For shortening the elapsed time of nonlinear dynamic analysis, parallel substructure method using four processors ($N_p=4$) are performed. The IHYT approach (Yang, 2000) is used for mesh partitioning and the multi-level substructure method is employed. Table 1 shows the total number of nonlinear iterations (column (C)), total elapsed time for parallel nonlinear dynamic finite element analysis (column (D)), and the parallel speedups. It can be seen that the total elapsed time of nonlinear dynamic analysis is reduced by using the parallel substructure method implemented in FE2000.

<table>
<thead>
<tr>
<th>Model names</th>
<th>Element type</th>
<th>Number of degrees-of-freedom</th>
<th>Sequential analysis</th>
<th>Parallel finite element analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Total number of iterations</td>
<td>Total elapsed time</td>
<td>Total number of iterations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(A)</td>
<td>(sec.)</td>
<td>(C)</td>
</tr>
<tr>
<td>H5-12</td>
<td>BC</td>
<td>25,344</td>
<td>60</td>
<td>2165.0</td>
</tr>
<tr>
<td>O1-12</td>
<td>BC</td>
<td>181,152</td>
<td>60</td>
<td>4056.8</td>
</tr>
<tr>
<td>M12BD-2</td>
<td>B20</td>
<td>64,809</td>
<td>75</td>
<td>19679.1</td>
</tr>
<tr>
<td>B20P6416</td>
<td>B20</td>
<td>34,767</td>
<td>60</td>
<td>9113.1</td>
</tr>
</tbody>
</table>

Our experiments have shown that similar parallel speedups can be achieved using other PC cluster and an IBM/SP2 parallel computer. The detailed statistics are not listed here because limited space.

Conclusions

In this work, an object-oriented nonlinear dynamic finite element analysis framework FE2000 is presented. The parallel substructure method is employed for parallel finite element analysis. An extension of the general sparse matrix package SPARSPAK, an iterative mesh partitioning approach, and a multi-level substructure method are employed and integrated. In the numerical experiments, up to 3.6 of speedup can be achieved using four processors (90% of parallel efficiency). Because of good portability, FE2000 also runs on other PC cluster and an IBM/SP2 parallel computer, and achieves similar parallel speedups to those using CAPS.
Acknowledgements

The authors acknowledge National Science Council

References


