Proc. Pakistan Acad. Sci. 45(2):97-103.2008

PERFORMANCE STUDIES OF A FINITE ELEMENT SOFTWARE FOR LINEAR ELASTICITY ANALYSIS

Abul Mukid Mohammad Mukaddes and Mohammed Mahbubul Islam

Department of Industrial and Production Engineering, Shahjalal University of Science and Technology, Sylhet, Bangladesh

Received December 2007, accepted April 2008

Abstract: ADVENTURE on Windows is a collection of finite element modules including ADVENTURE_Solid ported to Microsoft Windows for use in a single CPU. It supports the iterative domain decomposition method with a preconditioner for the solution of linear equations. This paper analyzed 3 dimensional linear elastic stress problems and measured the performance of the finite element software ADVENTURE on Windows. A complete finite element analysis from CAD design to visualization was performed in this research successfully. Two simple models have been considered for the linear elastic problems. Computational results show the computation time, required memory, convergence criteria and some limitations of the software.

Keywords: Finite Element Method, Linear Stress Analysis, ADVENTURE on Windows, Domain Decomposition Method, Balancing Domain Decomposition.

Introduction

A finite element method is a computational technique for the numerical solution of engineering problems such as elasticity analysis [1], heat transfer [2] and fluid mechanics [3]. There are many commercial finite element softwares in the field of Engineering. ADVENTURE on Windows [4] is one of them, which can perform the linear elasticity analysis using a single CPU. This module is a collection of finite element modules including: ADVENTURE_CAD [4], a polygon-based solid modeling program; ADVENTURE_TriPatch module [4], a triangular surface patches generating program from a solid model; ADVENTURE_TetMesh module [4], a tetrahedral mesh generating program from the triangular surface patches; ADVENTURE_ BCtool [4], a module used to set up boundary conditions onto a finite element mesh; ADVENTURE_Metis [4] a domain decomposition module in a parallel environment; ADVENTURE_Solid [1,4] an elastic stress analysis solver; ADVENTURE_Visual [4] a module which can visualize deformation contours on surfaces and cross sections, temperature, flux etc. All of the above modules are Linux based. ADVENTURE on Windows ported them to Microsoft Windows for using in a single CPU. The finite element solver (ADVENTURE_Solid) employed in the software uses the Hierarchical Domain Decomposition Method (HDDM) [5] which decomposes the whole model (problem) into some subdomains. The degrees of freedom inside the subdomains are solved using the Gaussian elimination method while the degrees of freedom that are shared by more than two subdomains are solved by the Conjugate Gradient (CG) [6] method. In addition, a diagonal scaling and a Balancing Domain Decomposition (BDD) [7] preconditioner are employed in the CG method. This paper intended to perform the linear elasticity analysis on two selected models using the ADVENTURE on Windows and measures the performance as well as some limitations of the software. A complete finite element analysis was thoroughly studied in this research. The computational performance of the finite element solver was also measured. Numerical results show that BDD is the most effective solver for any kind of problem. Based on the performance

Email: mukaddes-ipe@sust.edu

analysis some recommendations for the new users are also highlighted in the paper.

Materials and Methods

Flow Analysis

This research performed the complete finite element analysis of a elasticity problem. At first we prepare the solid model using the commercial CAD software and finally we visualize the result using the ADVENTURE software. The complete flow chart is shown in Fig. 1.

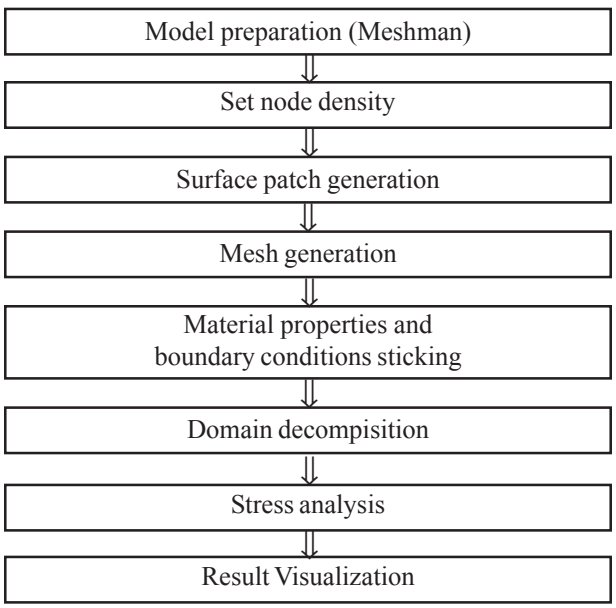

Fig. 1. Flow analysis.

For the model preparation, commercial CAD software, Meshman [8] was used. The surface patch and the mesh were generated using the software ADVENTURE_TriPatch and ADVENTURE_TetMesh. Then the material properties and boundary conditions were set up using the software ADVENTURE_BCtool. The ADVENTURE_Metis was used to decompose the whole domain while the stress analysis was performed using the ADVENTURE_ Solid. Finally, the results were visualized by the ADVENTURE _Visual.

Formulation of the Elasticity Analysis

To explain the theory of elasticity analysis employed in the ADVENTURE on Windows, let us consider a structural problem concerning a domain Ω, as shown in Fig. 2. Here, $\overline{F_i}$ is the traction force applied on the boundary Γ_F, $\overline{B_i}$ the body force applied in the domain Ω, and $\overline{u}$ the prescribed displacement on the boundary Γ_u.

Fundamental equations of this structural problem are summarized as follows:

$$\begin{aligned} &\tau_{ij,j} + \overline{B_i} = 0 && \text{in } \Omega \\ &\varepsilon_{ij} = \left(u_{i,j} + u_{j,i}\right)/2 && \text{in } \Omega \\ &\tau_{ij} = D_{ijmn}\varepsilon_{mn}^{(e)} && \text{in } \Omega \\ &\tau_{ij}n_j - \overline{F_i} = 0 && \text{on } \Gamma_F \\ &u_i = \overline{u_i} && \text{on } \Gamma_u \end{aligned} \tag{1}$$

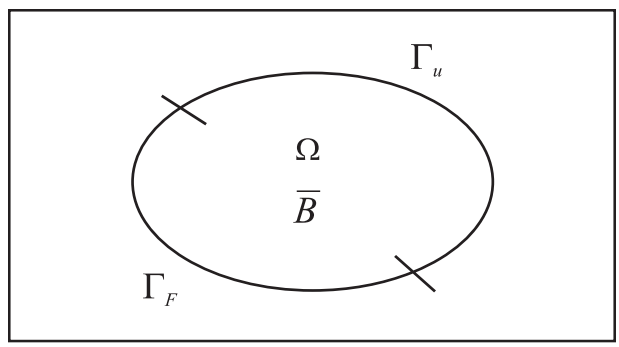

Fig. 2. Domain analysis.

where, *i, j, m, n* take the value 1, 2, 3, u_i is displacement, ε_{ij} a strain tensor, τ_{ij} stress tensor, D_{ijmn} a coefficient tensor of the Hookís law and n_j an outer normal vector on the boundary Γ, respectively.

The finite element (quadratic tetrahedral) discretization of Eq. 1 yields a linear system of the form

$$Ku = b. \tag{2}$$

where, K, u and b are the stiffness matrix, the displacement vector and the force vector, respectively. In this research, we analyzed the elasticity problem by using the software ADVENTURE_Solid where the BDD method based on the HDDM system was employed.

Domain Decomposition Method

The domain Ω is decomposed into N non-overlapping subdomains, $\{\Omega_i\}_{i=1,\ldots,N}$. As usual K can be generated by subassembly:

$$K = \sum_{i=1}^{N} R^{(i)} K^{(i)} R^{(i)T} \tag{3}$$

where $R^{(i)T}$ is the 0-1 matrix which translates the global indices of the nodes into local numbering. Let $u^{(i)}$ be the vector corresponding to the elements in $\Omega^{(i)}$ and it can be expressed as $u^{(i)} = R^{(i)T} u^{(i)}$. Each is split into degrees of freedom, which correspond to, called interface degrees of freedom and the remaining interior degrees of freedom. The subdomain matrix, vector and 0-1 matrices are then split accordingly:

$$K^{(i)} = \begin{pmatrix} K_{II}^{(i)} & K_{IB}^{(i)} \\ K_{IB}^{(i)T} & K_{BB}^{(i)} \end{pmatrix}, \tag{4}$$

$$u^{(i)} = \begin{pmatrix} u_I^{(i)} \\ u_B^{(i)} \end{pmatrix}, \tag{5}$$

and $R^{(i)} = \left(R_I^{(i)}, R_B^{(i)}\right)$ (6)

After eliminating the interior degrees of freedom, the problem of Eq. 2 reduces to a problem on interface,

$$Su_B = g \tag{7}$$

where $S = \sum_{i=1}^{N} R_B^{(i)} S^{(i)} R_B^{(i)T}$ is assumed to be positive definite, u_B is the vector of the unknown variables on the interface, g is a known vector and $S^{(i)}$ are the local Schur complements of subdomain $i = 1,\ldots, N$, assumed to be positive semi-definite. The problem of Eq. 7 is solved by a preconditioned CG method which solves a problem.

$$z = M^{-1} r \tag{8}$$

where r is the residual of Eq. 7 and M is a preconditioner. When the interface problem is solved iteratively, of course, an efficient solution of the large scale problems depends on how we choose an efficient and scalable preconditioner.

Balancing domain decomposition method (BDD)

The BDD preconditioning technique proposed by Jan Mandel [7] uses at each CG iteration solution of the local Neumann-Neumann problems on the subdomains coupled with a coarse problem in a coarse space. The BDD preconditioner is of the form:

$$M_{BDD}^{-1} = Q_c + \left(I - Q_c S\right) Q_l \left(I - S Q_c\right) \tag{9}$$

where Q_1 is the local level part and Q_c is the coarse level part of the preconditioner.

Local Level

The local level part of the preconditioner basically involves the solution of local problems. is Q_1 expressed by

$$Q_l = \sum_{i=1}^{N} R_B^{(i)} D^{(i)} S^{(i)+} D^{(i)T} R_B^{(i)T} \tag{10}$$

The dagger (+) indicates pseudo-inverse, since the $S^{(i)}$ is singular for floating subdomain. The BDD method uses a collection of matrices $D^{(i)}$ that determine partition of unity on interface (2,7),

$$\sum_{i=1}^{N} R_B^{(i)} D^{(i)} R_B^{(i)T} = I. \tag{11}$$

The simplest choice for $D^{(i)}$ is the diagonal matrix with diagonal elements equal to the reciprocal of the number of subdomains with which the degree of freedom is associated.

Coarse Level

The application of the coarse term $Q_c = R_0 \left(R_0^T S R_0\right)^{-1} R_0^T$ amounts to the solution of a coarse problem whose coefficient matrix is $S_w = R_0^T S R_0$. The operator R_0 translates the coarse degrees of freedom to the corresponding global degrees of freedom and is defined by

$$R_0 = \left[R_B^{(i)} D^{(i)} Z^{(i)}, \ldots, R_B^{(N)} D^{(N)} Z^{(N)}\right] \tag{12}$$

For the structural problem comes from the degrees of freedom of rigid body of motion (1).

Simplified diagonal scaling (DIAG)

We choose a diagonal matrix as a preconditioner whose diagonal elements are constructed from the corresponding ones of $K_{BB}^{(i)}$. We define the diagonal matrix

$$Q_{DIAG} = \sum_{i=1}^{N} R_B^{(i)} \left(diag\left(K_{BB}^{(i)}\right)\right)^{-1} R_B^{(i)T} \tag{13}$$

Results and Discussion

This research considered an L-shaped model (Fig. 3) which was approximately 50 mm on the longest edge and the sectional shape was 10 mm x 10 mm and a cubic model (Fig. 4) which was approximately 20 mm × 20 mm × 20 mm for the linear elasticity analysis. The sizes of the model are shown in the Table 1. Fig. 5 shows the relationship between the node density and degrees of freedom (for L-shaped model). For a fixed size problem, decreasing the node density resulted in increasing the number of nodes, hence increasing the number of degrees of freedom. Same material properties as Youngís Modulus 21000.0 MN/m^2, Poissonís Ratio 0.4, Mass density 760.0 kg/m^3 were set for both models. For the L-shaped model a tensile load on the face 1(1.0N per unit area in the positive X direction) and constraint on the face 2 was applied. For the cubic model, a compressive load on face 1 (1.0N per unit area in the negative Y direction) and constraint on face 2 was applied. Fig. 6 shows the visual result (modifying the value of magnification factor of deformation as 40.0) of the L-shaped model which shows the deformation along X axis. Fig. 7 depicts the visual result (modifying the value of magnification factor of deformation as 100) of the cubic model which shows the deforma-

Table 1.
Size of the model.

	Model 1 (L-shaped)	Model 2 (cubic)
Number of subdomains	42	38
Number of Nodes	3943	3231
Degrees of Freedom	11829	9693

Table 2.
Computational performances (L-shaped model).

	No of iteration		Memory (MB)		Computation time (sec)	
Solver type	L-shaped	Cubic	L-shaped	Cubic	L-shaped	Cubic
CG	453	1492	6.2	7.18	28.2	93.2
HDDM	244	745	12.3	13.82	15.3	45.5
BDD	28	26	20.3	22.5	2.61	2.28

Table 3.
Performance comparison among different CPU (RAM size).

RAM size (MB)	Maximum DOF	Computation time (min)
128	115,026	220.2
256	186,921	113
512	463,071	137

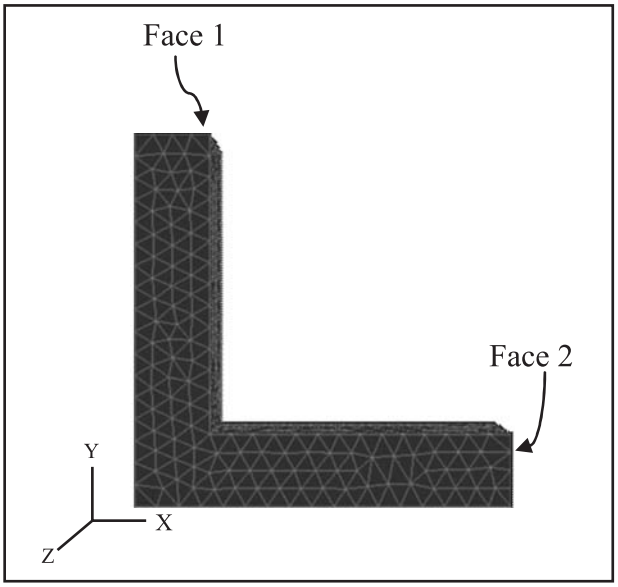

Fig. 3. Mesh of L-shaped model.

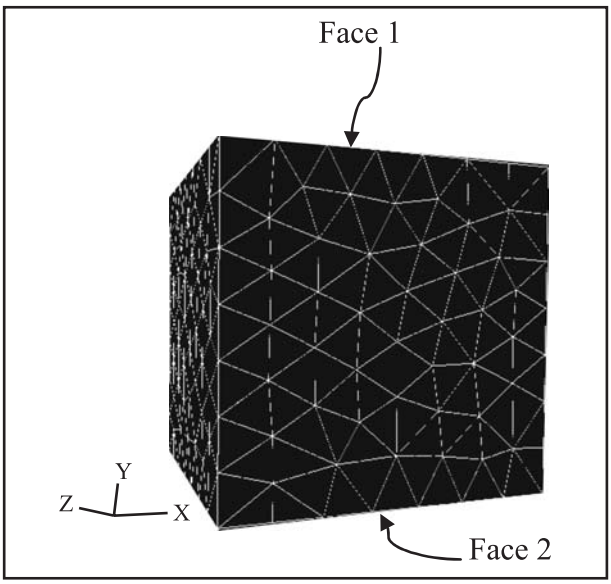

Fig. 4. Mesh of Cubic model.

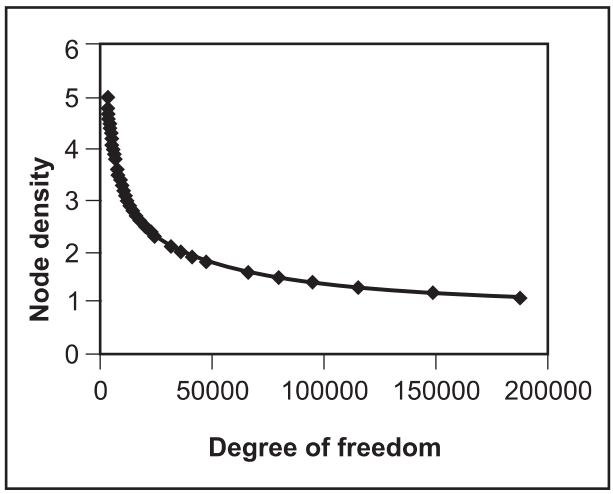

Fig. 5. Degrees of freedom vs. Node density.

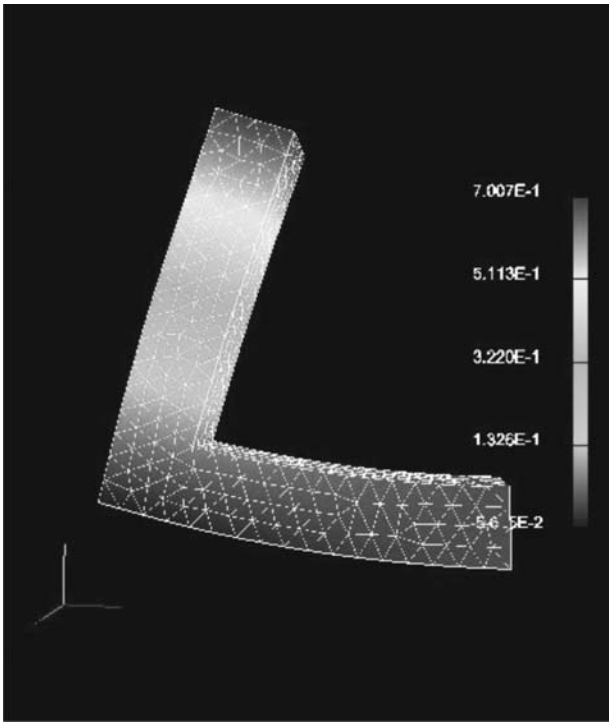

Fig. 6. Deformation (X 40) in the X-direction (L-shaped model).

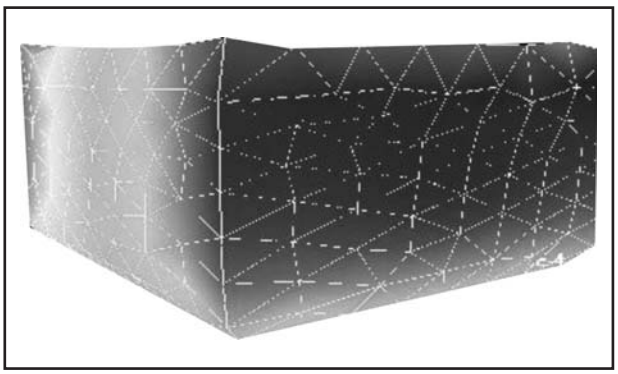

Fig. 7. Deformation (X 10000) in the Y-direction (Cubic model).

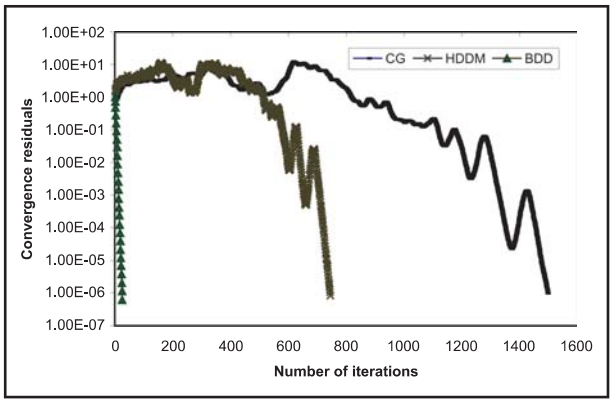

Fig. 8. CG convergence (L-Shaped model).

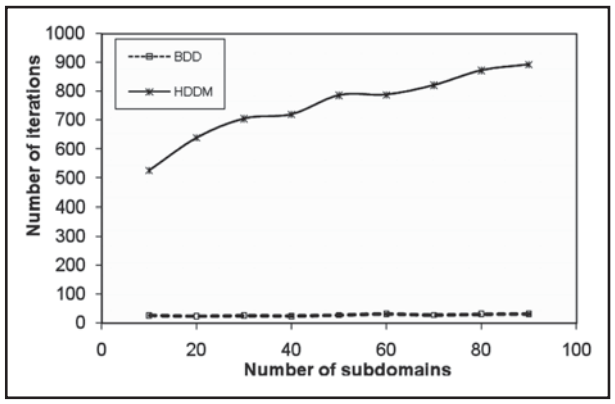

Fig. 9. Numerical scalability of BDD method.

tion along Y axis. The computational performance for both models is shown in Table 2. In Table 2, CG represents DDM [5] without any preconditioner, HDDM [1, 2] represents DDM with a simplified diagonal scaling (Eq. 13) and BDD represents the DDM with the BDD preconditioner (Eq. 12). Among the three solvers, BDD showed better performance compared to CG and HDDM solvers. The CG convergence of L-Shaped model in Fig. 8 shows that BDD converged rapidly compared to the other two solvers. Fig. 9 shows the relation between the number of subdomains and number of iterations for BDD and HDDM solver. It shows that, for a fixed size problem, with HDDM solver increasing the number of subdomains results in increasing the number of iterations, hence increasing the computation time. With BDD solver, the number of iterations is independent of the number of subdomains. So, BDD is the most effective solver for any kind of problem. The problem size (maximum degrees of freedom) that can be solved in the computer is specified in Table 3. It is concluded that by increasing the RAM size one can solve the large scale problem with the same computer.

This paper compared three finite element solvers DDM [5], HDDM [1, 2] and BDD [7]. With the BDD discussed in this paper the number of iterations is reduced to 1/16 of original DDM [5] and computational time is reduced to 1/40. But BDD requires more memory compared with the algorithm discussed in [1, 2]. Since it takes more memory, this research suggests the user to use BDD solver in the computer with sufficient memory. Again BDD is numerically scalable in elasticity problem as it is in heat transfer problem [9] and fluid mechanics problem [10].

Conclusion

A complete finite element process was thoroughly studied in this research. A simple elastic problem was analyzed using the finite element software ADVENTURE on Windows. The performance of the software was measured. BDD solver was found as the most efficient one. It converged rapidly and solved the problem with minimum computation time compared with the other solvers. The number of iterations was independent of the number of subdomains for the BDD solver hence it is said in this study to be numerically scalable. Based on the computation time, some information for the new user is highlighted. Future work is needed to further analyze the practical elastic problem using the above mentioned software.

References

1. **Shioya, R., Ogino, M., Kanayama, H. and Tagami, D.** 2003. Large Scale Structural Analysis Using a Balancing Domain Decomposition Method. *Key Eng. Materials*. 243,244:21-26.
2. **Shioya, R., Kanayama, H., Mukaddes, A.M.M. and Ogino, M**. 2003. Heat Conductive Analysis with Balancing Domain Decomposition. *Theor. Appl. Mech*. 52:43-53.
3. **Goldfeld, O.** 2003. *Balancing Neumannn-Neumann for (In) Compressible Linear Elasticity and Stokes-Parallel Implementation*. Proc. of 14th Intl. Conf. on Domain Decomposition Methods for Partial differential Equations.
4. http://adventure.q.t.u-tokyo.ac.jp/
5. **Yagawa, G. and Shioya, R.** 1993. Parallel Finite Elements on a Massively Parallel Computer with Domain Decomposition. *Comput. Systems Eng.* 4:495-503.
6. **Yousef, S.** 1996. *Iterative Methods for Sparse Lin-*

ear Systems. PWS Publishing Company, Boston, MA 02116.

7. **Mandel, J.** 1993. Balancing Domain Decomposition. *Comm. Numer. Methods Eng*. 9:223-241.
8. http://www.meshman.jp/
9. **Mukaddes, A.M.M., Ogino, M., Kanayama, H. and Shioya, R.** 2006. A Scalable Balancing Domain Decomposition Based Preconditioner for Large Scale Heat Transfer Problems. *JSME Int. J.* Ser. B. 49-2:533-540.
10. **Kanayama, H., Ogino, M., Takesue, N. and Mukaddes, A.M.M.** 2005. Finite Element Analysis for Stationary Incompressible Viscous Flows Using Balancing Domain Decomposition. *Theor. Appl. Mech*. 54:211-219.