Development of Eigen Value Solver

Renu P S
SCMS School of Engineering and Technology
Karukutty, Angamali

Airin M G
SCMS School of Engineering and Technology
Karukutty, Angamali

Rajeev G Nair
Scientist/ Engineer
VSSC, ISRO
Thiruvananthapuram

***Abstract*—Preconditioned Conjugate Gradient Lanczos algorithm method is a powerful iterative method used to solve eigen-value problem of symmetric matrices encountered in Finite element method. A preconditioned iterative method for computing a few eigen pairs of large sparse symmetric matrices is presented in the present thesis. The proposed method of Preconditioned Conjugate Gradient Lanczos algorithm is suitable for determination of extreme eigenvalues. The convergence of the conjugate gradient method can be further improved by suitable preconditioning. The thesis presents solution of an actual problem by implementing the algorithm into the FEAST-SMT code. The results are verified with the existent Lanczos solver. It is noted that the implementation of this solver in FEAST drastically reduces the solution time compared to the existing solvers in FEAST-SMT. PCG Lanczos method is considered because of its improved efficiency in terms of time and accuracy compared to other methods.**

I. INTRODUCTION

The computational demand in aerospace industry is very high, due to the requirement of accurate prediction of the results on account of the less margin of safety available for the design of aircraft, spacecraft, launch vehicle space shuttle etc. Aerospace vehicles are very huge and complex in nature which requires a large number of degrees of freedom to model the structure, hence the time required to solve such model is also very high. The computational core that consumes more Central Processing Unit (CPU) time on almost all the numerical simulation packages in engineering is the linear solver. For this reason, the linear solver implementation demands efficient algorithms. A solver is a generic term indicating a piece of mathematical software, possibly in the form of a stand-alone computer program that 'solves' a mathematical problem. They are mainly, the direct solver and the iterative solver. Depending on the problems on hand, the latter of the two solvers can drastically reduce the solution (or elapsed) time and the computing resource (CPU time and disk space) requirements by orders of magnitude. Reduction in solution phase for General purpose Finite Element software will help analysts and design engineers to improve their productivity and product quality by arriving at better solutions faster. Typically, this solver is based on a classical solution method that could be a direct or iterative method. Besides, the matrix associated with the linear system of equations is often large and sparse. Iterative methods have the disadvantage that they are not general and require additional user settings; moreover they also need preconditioners to accelerate their convergence. However, these methods are popular because their main operation is the matrix - vector product, and this operation is highly parallelizable. Direct methods such as Cholesky, LDL^T or LU are perfect black boxes as they require only the matrix (A) and the right hand side vector (b) of the linear system $Ax= b$ as inputs. Eigen solver determines natural frequencies and mode shapes of structures, typically some of the most computationally demanding tasks in structural analysis. Eigen solver combines the speed and memory savings of PCG iterative solver with the robustness of the Lanczos algorithm. This powerful combination allows users to solve for the natural frequencies and mode shapes of their model using fewer computational resources, often in shorter total elapsed times than other available Eigen solvers. This project aims to develop an Eigen value solver utilizing PCG Lanczos algorithm. As problem size increased the PCG Lanczos Eigen solver was found to be faster than the Block Lanczoseigen solver for a higher number of requested modes. Like all iterative solvers, the PCG Lanczos Eigen solver is most efficient when the solution converges quickly. If a model does not converge quickly in a static analysis, the PCG Lanczoseigen solver would not be expected to converge quickly either and would therefore be less efficient. Other factors such as the size of the problem and the hardware being used can affect the decision on which Eigen solver to use. For example, on machines with slow I/O performance or limited hard drive space, the PCG Lanczos Eigen solver may be the better choice.

II. DYNAMIC ANALYSIS

Civil engineering structures are always designed to carry their own dead weight, superimposed loads and environmental loads such as wind or waves. These loads are usually treated as maximum loads not varying with time and hence as static loads. In some cases, the applied load involves not only static components but also contains a component varying with time which is a dynamic load. In the past, the effects of dynamic loading have often been evaluated by use of an equivalent static load, or by an impact factor, or by a modification of the factor of safety. Many developments have been carried out in order to try to quantify the effects produced by dynamic loading. Examples of structures where it is particularly important to consider dynamic loading effects are the construction of tall buildings, long bridges under wind-loading conditions and buildings in earthquake zones, etc.

A. Vibration

Vibrations are time dependent displacements of a particle or a system of particles with respect to an equilibrium position. If these displacements are repetitive and their repetitions are executed at equal interval of time with respect to an equilibrium position, the resulting motion is said to be periodic. One of the most important parameters associated with engineering vibration is the natural frequency. Each structure has its own natural frequency for a series of different modes which control its dynamic behaviour

III. FINITE ELEMENT METHOD

Finite Element Method (FEM) is a versatile and popular numerical method due to its general procedure followed in assembly and solution techniques. This method differs only in the development of element matrices which needs the field expertise to obtain such matrices. This method is independent of boundary conditions and loading, because loads and boundary conditions can be applied anywhere on the structure. This method can be easily extended to non-linear and multi-physics field.

A. Advantages of Finite Element Method

1) Can readily handle very complex geometry.
2) Can handle a wide variety of engineering problems like solid mechanics, dynamics, heat transfer problems, fluids, electrostatic problems etc.
3) Can accommodate complex restraints (Indeterminate structures can be solved).
4) Can handle complex loadings like nodal load (point loads), element load (pressure, thermal, inertial forces), time or frequency dependent loading, etc.

IV.EIGEN VALUES AND EIGEN VECTORS

Aircrafts, Launch vehicles, Space-crafts, Space-stations, Space-shuttles, Re-usable Launch vehicles etc. are inevitably subjected to dynamic loadings during its service life. Analysis of such structures using finite element method requires large number of elements and nodes for accurate prediction of the analysis parameters such as mode shapes, frequencies etc. The requirement of large number of finite elements and nodes in the analysis will exceed the memory (RAM) of the computer. In-order to solve such huge structures, effective solvers was introduced in software. The solver which introduced in this thesis work is eigen solver, it is an iterative solver based on PCG Lanczos method. Eigen solvers determine natural frequencies and mode shapes of structures, typically some of the most computationally demanding tasks in structural analysis.

The Eigen value problem is a problem of considerable theoretical interest and wide-ranging application. For example, this problem is crucial in solving systems of differential equations, analyzing population growth models, and calculating powers of matrices (in order to define the exponential matrix). Other areas such as physics, sociology, biology, economics and statistics have focused considerable attention on “Eigen values” and “Eigen vectors”-their applications and their computations

A. Properties of Eigen values and Eigen vectors

1) The absolute value of a determinant ($|detA|$) is the product of the absolute values of the Eigen values of matrix A
2) c = 0 is an Eigen value of A if A is a singular (non invertible) matrix
3) If A is an nxn triangular matrix (upper triangular, lower triangular) or diagonal matrix, the Eigen values of A are the diagonal entries of A.
4) A and its transpose matrix have same Eigen values. 5) Eigen values of a symmetric matrix are all real.
6) Eigen vectors of a symmetric matrix are orthogonal, but only for distinct Eigen values.
7) The dominant or principal Eigen vector of a matrix is an Eigen vector corresponding to the Eigen value of largest magnitude (for real numbers, largest absolute value) of that matrix.
8) For a transition matrix, the dominant Eigen value is always 1.
9) The smallest Eigen value of matrix A is the same as the inverse (reciprocal) of the largest Eigen value of A^{-1}; i.e. of the inverse of A.

V. LANCZOS AND PCG LANCZOS METHOD

The Lanczos process is an effective method for computing a few Eigen values and corresponding Eigen vectors of a large sparse symmetric matrix. Krylov subspace methods are popular for both Eigen value problems and linear equations problems. Krylov subspaces are used by the Lanczos algorithm for Eigen values and by the conjugate gradient method for linear equations. Modern iterative methods for finding one (or a few) Eigen values of large sparse matrices or solving large systems of linear equations avoid matrix-matrix operations, but rather multiply vectors by the matrix and work with the resulting vectors. Starting with a vector, b, one computes , and then one multiplies that vector by A to find and so on. All algorithms that work this way are referred to as Krylov subspace methods; they are among the most successful methods currently available in numerical linear algebra. The best known Krylov subspace methods are the Arnoldi, Lanczos, Conjugate gradient, GMRES (generalized minimum residual), BiCGSTAB (biconjugate gradient stabilized), QMR (quasi minimal residual), TFQMR (transpose-free QMR), and MINRES (minimal residual) methods.

The Iterative Solver is a PCG (Preconditioned Conjugate Gradient) solver. The PCG solver pre-conditions the global matrix instead of decomposing the global matrix, and then iterates the solution based on the pre-conditioned global matrix until the solution converges to certain accuracy. The preconditioning process takes very little time compared to the PCG iteration process and is based on the underlying physics of the discretization of a continuous problem, geometry of the elements, and characteristics of the different types of elements.

Direct solvers decompose the global matrix and goes through forward and backward substitution to obtain a solution. Once the global matrix is decomposed, multiple solutions can be obtained very fast from forward and backward substitution. Compared to the basic frontal solver, the sparse matrix solver stores the global matrix in a very compact form, and the operations are performed only on nonzero values. Therefore, the sparse matrix solver requires far less disk space and solution time than the basic frontal solver, especially for largesize problems.

The Lanczos algorithm is an iterative algorithm invented by Cornelius Lanczos that is an adaptation of power methods to find Eigen values and Eigen vectors of a square matrix or the singular value decomposition of a rectangular matrix. It is particularly useful for finding decompositions of very large sparse matrices. The power method for finding the largest Eigen value of a matrix A can be summarized by noting that if x_0 is a random vector and $x_{n+1} = Ax^n$, then in the largest n limit, $\frac{x_n}{\|x_n\|}$ approaches the normed Eigen vector corresponding to the largest Eigen value.

VI. CHOLESKY DECOMPOSITION

Cholesky method is a direct method for solving a linear system that makes use of the fact that any square matrix [A] can be expressed as the product of an upper and a lower triangular matrix. This method can express any square matrix as a product of two triangular matrixes. The subsequent solution procedure is:

$$[A]x = b$$

The matrix [A] can be written as $[A] = [a_{ij}] = [L][U]$ Where; $[L] = [l_{ij}]$ is a lower triangular matrix and $[U] = [u_{ij}]$ is a unit upper triangular matrix.

For example

$$\begin{bmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & l_{21} \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} I_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix}$$

So we have,

$$l_{11} = \sqrt{a_{11}},\ l_{21} = \frac{a_{21}}{l_{21}},\ l_{22} = \sqrt{a_{22} - l_{21}^2}$$

Cholesky Factorisation Algorithm

```
for k = 1 to n
$a_{kk} = \sqrt{} a_{kk}$
for i = k + 1 to n
$a_{ik} = a_{ik} / a_{kk}$
 end
for j = k + 1 to n
 for i = j to n
 $a_{ij} = a_{ij} - a_{ik}a_{jk}$
end
end
end
```

VII. SPARSE MATRIX

In the subfield of numerical analysis, a sparse matrix is a matrix populated primarily with zero. The term itself was coined by Harry M. Markowitz. Conceptually, sparsity corresponds to systems which are loosely coupled. Operations using standard dense matrix structures and algorithms are slow and consume large amounts of memory when applied to large sparse matrices. Sparse data is by nature easily compressed, and this compression almost always results in significantly less computer data storage usage. Indeed, some very large sparse matrices are infeasible to manipulate with the standard dense algorithms.

There are many methods for storing the data. They are compressed row and column storage, block compressed row storage, diagonal storage, jagged diagonal storage, and skyline storage.

A. Compressed row storage

The compressed row and column storage formats are the most general: they make absolutely no assumptions about the sparsity structure of the matrix, and they don’t store any unnecessary elements. On the other hand, they are not very efficient, needing an indirect addressing step for every single scalar operation in a matrix-vector product or preconditioner solve. The compressed row storage (CRS) format puts the subsequent non zero of the matrix rows in contiguous memory locations.

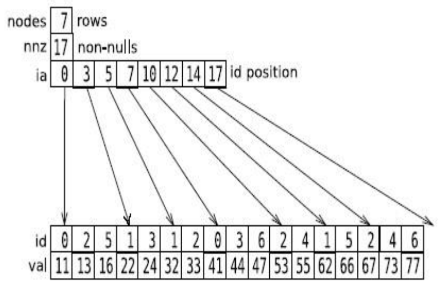
nodes 7 rows
nnz 17 non-nulls
ia 0 3 5 7 10 12 14 17 id position
id 0 2 5 1 3 1 2 0 3 6 2 4 1 5 2 4 6
val 11 13 16 22 24 32 33 41 44 47 53 55 62 66 67 73 77

Fig. 1. Compressed Row Storage

VIII. PCG LANCZOS ALGORITHM USING LANCZOS ALGORITHM

Eigen solvers determine natural frequencies and mode shapes of structures, typically some of the most computationally demanding tasks in structural analysis. In this thesis an algorithm is developed for preconditioned conjugate gradient method with the robustness of Lanczos algorithm. By using this algorithm an Eigen value solver is developed. The algorithms are shown below

IX. INTRODUCTION OF THE PCG LANCZOS ALGORITHM TO THE SOFTWARE PREWIN 9.0.

FEAST (Finite Element Analysis of Structures) is a finite element software which consists of a pre-processor (PreWin) and a post-processor (SMT Solver) both developed by VSSC.

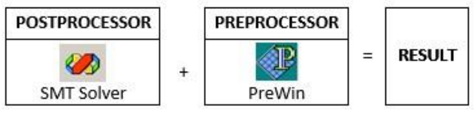

Fig. 2. General Representation of the software

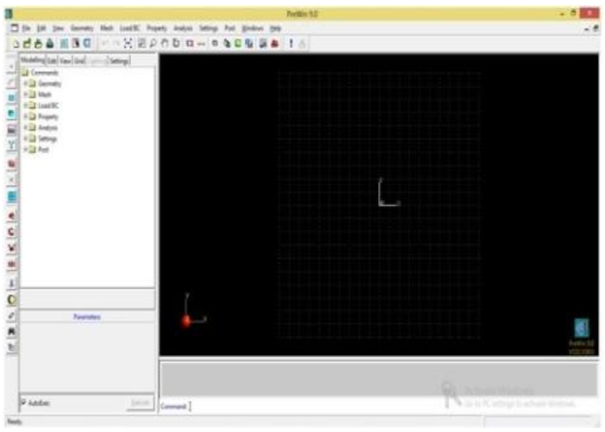

Fig. 3. PreWin 9.0 Workbench

SMT (Substructured and Multithreaded) software is developed as part of FEAST family of structural analysis programs, with a modified architecture using object-oriented programming concepts. 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 position of the node in the substructure.

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.7.3). 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.

The parent software FEAST (Finite Element Analysis of Structures) consists of several hundred lines of procedural code written in FORTRAN77. Data such as nodal coordinates and element connectivity are stored in multi-dimensional arrays, which are accessed throughout the program. The global access decreases the flexibility of the system. It is difficult to modify the existing codes and to extend the codes to adapt them for new uses, models and solution procedures. The inflexibility is experienced in several ways (1) a high degree of knowledge of the entire program is required to work in even a minor portion of the code, (2) reuse of code is difficult, (3) a small change in the data structures can have a ripple effect throughout the system, (4) the numerous interdependencies between the components of the design are difficult to determine and (5) the integrity of the data structures is not assured.

A redesign of the software is needed to remove this inflexibility. 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. Thus the software is redesigned to C++.

X. NUMERICAL STUDY

Civil engineering structures are always designed to carry their own dead weight, superimposed loads and environmental loads such as wind or waves. These loads are usually treated as maximum loads not varying with time and hence as static loads. In some cases, the applied load involves not only static components but also contains a component varying with time which is a dynamic load. In the past, the effects of dynamic loading have often been evaluated by use of an equivalent static load, or by an impact factor, or by a modification of the factor of safety. Many developments have been carried out in order to try to quantify the effects produced by dynamic loading. Examples of structures where it is particularly important to consider dynamic loading effects are the construction of tall buildings, long bridges under wind-loading conditions and buildings in earthquake zones, etc. Typical situations where it is necessary to consider more precisely the response produced by dynamic loading are vibrations due to equipment or machinery, impact load produced by traffic, snatch loading of cranes, impulsive load produced by blasts, earthquakes or explosions. So it is very important to study the dynamic nature of structures.

Vibrations: Vibration are time dependent displacements of a particle or a system of particles with respect to an equilibrium position. If these displacements are repetitive and their repetitions are executed at equal interval of time with respect to equilibrium position the resulting motion is said to be periodic. One of the most important parameters associated with engineering vibration is the natural frequency. Each structure has its own natural frequency for a series of different modes which control its dynamic behavior. Whenever the natural frequency of a mode of vibration of a structure coincides with the frequency of the external dynamic loading, this leads to excessive deflections and potential catastrophic failures. This is the phenomenon of resonance. An example of a structural failure under dynamic loading was the well known Tacoma Narrows Bridge during wind induced vibration. In practical application the vibration analysis assumes great importance. For example, vehicle-induced vibration of bridges and other structures that can be simulated as beams and the effect of various parameters, such as suspension design, vehicle weight and velocity, damping, matching between bridge and vehicle natural frequencies, deck roughness etc., on the dynamic behavior of such structures have been extensively investigated by a great number of researchers. Every structure which is having some mass and elasticity is said to vibrate. When the amplitude of these vibrations exceeds the permissible limit, failure of the structure occurs. To avoid such a condition one must be aware of the operating frequencies of the materials under various conditions like simply supported, fixed or when in cantilever conditions.

A. Application of Finite Element Software Feast

FEAST is a Finite element analysis software package developed in VSSC, Thiruvananthapuram, useful for structural and thermal analysis of structures. Initiated in 1977, the development and augmentation of the package has been continuing and over the years FEAST has evolved into a family of software modules, each module catering to a specific set of capabilities. The FEAST family has modules designated as FEAST-B, FEAST-C4.2, FEAST-HT 3.0, FEAST-NL 4.0, FEAST-R 2.0, PREWIN 9.0 and SMT 2.0, the subscript indicating the type of analyses that the module can cater to. This thesis develops a code for Eigen solver combined with Preconditioned Conjugate Gradient method with the robustness of Lanczos algorithm in FEAST-SMT. To validate the performance of the developed code in FEAST-SMT certain test problems are carried out and the results are compared with other general purpose software.

TABLE 1.MATERIAL PROPERTIES OF THE CYLINDER

Material properties	Values
Modulus of elasticity	2.05e+11 N/m2
Poissons ratio	0.3
Mass density	7800 kg/m3
Boundary conditions Ux, Uy, Uz , Rx , Ry , Rz	0,0,0,0,0,0
Analysis type	Free Vibration
Number of Eigen values	6

TABLE 2. COMPARISON OF FREQUENCIES AND TIME DURATION USING PCG LANCZOS SOLVER, LANCZOS SOLVER AND ANSYS

Sl no	PCG Lanczos solver Time taken (2.215sec)	Lanczos solver Time taken (3.089 sec)
	Frequencies (Hz)	Frequencies (Hz)
1	653.904	653.904
2	653.904	653.904

B. Dynamic Analysis Clamped Free Cylinder Using FEAST

1) Cylinder 32 × 16 *Fixed at the Bottom:* The model of a clamped-free cylinder in PreWin 9.0 (FEAST). The cylinder is made up of steel. 4 noded shell elements is used for modelling. The geometric details are shown in figure 4 and material properties are shown in table I.

2)

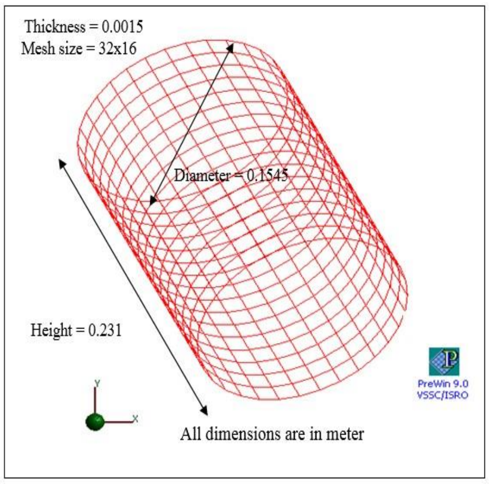

Fig. 4. Model of 32×16 Meshed Cylinder

Cylinder 32 × 16 is modeled and analyzed using FEAST and ANSYS. The same model is analyzed under free vibration; frequencies and time taken are then compared. The results obtained were tabulated. It shows that the frequencies are almost matched. But in case of time taken the result obtained from FEAST shows slight variation from those obtained using ANSYS. When compared with ANSYS, PCG Lanczos solver shows better performance than Lanczos solver.

3) Sounding Rocket Model: A typical sounding rocket model discretized with four noded quadrilateral plate/shell elements is shown in Fig.5 .The rocket model is subjected to free vibration with no boundary condition (free-free condition). The comparison of frequency and time of the model with
NASTRAN results is shown in Fig. 6 and 7. The table III

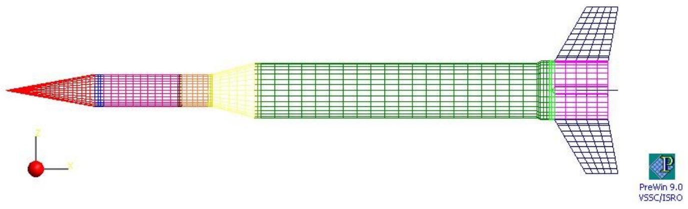

Fig. 5. Typical Sounding Rocket Model

TABLE 3. COMPARISON OF FREQUENCIES USING PCG LANCZOS SOLVER, LANCZOS SOLVER AND NASTRAN

Sl no	PCG Lanczos solver Frequencies (Hz)	Lanczos solver Frequencies (Hz	NASTRAN Frequencies (Hz)
7	53.0393	53.0393	59.601
8	61.4299	61.4299	61.429
9	61.6813	61.6813	61.681
10	63.0348	63.0348	63.034
11	64.8052	64.8052	64.805
12	65.5802	65.5802	65.580

Table 3 shows the comparison of bending frequencies using PCG Lanczos solver, Lanczos solver and NASTRAN.

TABLE 4.COMPARISON OF TIME TAKEN BY SMT SOLVER

Solver	FEAST-SMT	
	PCG Lanczos solver	Lanczos solver
Time Taken (sec)	15.486	26.988

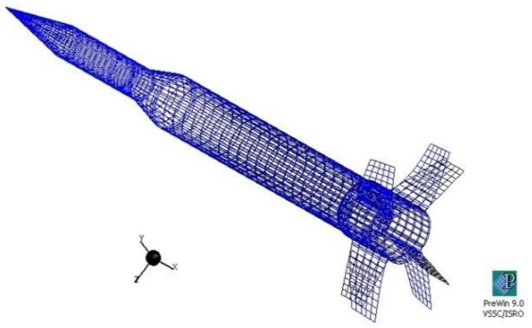

Fig. 6. Seventh mode shape from PreWin

XI. CONCLUSION

An eigen solver is developed using PCG Lanczos algorithm and it is implemented in FEAST-SMT. The code was tested using some numerical problems and results are compared with other solvers in FEAST-SMT. From tables 5.1 and 5.2 it is seen that the natural frequencies are almost matched and the time taken by each solver are varying. But the variation in time is found to be least in PCG Lanczos solver when compared to other solvers. The ultimate aim of the project is the same. The code developed for eigen solver using C++ programming language has been implemented in FEAST software. The

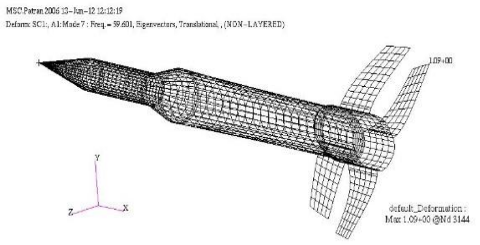
MSC.Patran 2006 13-Jun-12 12:12:19
Deform: SC1:, A1:Mode 7 : Freq. = 59.601, Eigenvectors, Translational, , (NON-LAYERED)
1.09+00
Y
Z
X
default_Deformation :
Max 1.09+00 @Nd 3144

Fig. 7. Seventh mode shape from Nastran

developed code is validated using numerical examples which include cylinder and an actual launch vehicle model.

Cylinder and an actual launch vehicle with different end conditions subjected to free vibration analysis are adopted as test cases to validate whether the developed code works well for any element. The results obtained from FEAST software are compared with the values obtained from other general purpose software. The comparison shows that the FEAST results are in good agreement with other software results.

In order to validate whether the developed code is capable of carrying out the free vibration analysis of not only in case of simple element like cylinder, a typical sounding rocket model is tested. Results obtained from FEAST software are compared with NASTRAN, finite element software frequently being used in Vikram Sarabhai Space Centre for dynamic analysis of launch vehicles. Comparison shows a good match between both the results. Hence it is concluded that the code developed for eigen solver extraction works well both in simple as well as complex domains.

REFERENCES

[1] 1. Ronald B. Morgan and David S. Scott (1993), Preconditioning the Lanczos algorithm for sparse symmetric Eigen value problems. Society for Industrial and Applied Mathematics, 14, 585-593.

[2] 2. Young Cho, Yook-Kong Yong (1999) A multi-mesh, preconditioned conjugate gradient solver for Eigen value problems in finite element models. Department of Civil and Environmental Engineering.

[3] 3. Ichikawa Tetsuji, Torigaki Toshikazu (1999) Eigen value Analysis of Large-Scale Voxel Models. Nissan Mot. Co., Ltd.

[4] 4. Hua Dai, Peter Lancaster (2000) Preconditioning Block Lanczos Algorithm for solving symmetric Eigen value problem

[5] 5. Anshul Gupta, Vipin Kumar; A Scalable Parallel Algorithm for Sparse Cholesky Factorization, In Super Computing 94.

[6] 6. Chapman, Stephen J; FORTRAN 90/95 for scientists and engineers Ed.2, Mc Graw Hill Boston 2004.

[7] 7. Hall, W J/Newmark, N M/Gallagher, Richard H; Finite element analysis: fundamentals, New Jersey/Prentice-Hall/1975.

[8] 8. J.A. Castellanos, G. Larrazabal; A Cholesky out-of-core factorization, Multidisciplinary Center for Scientific Visualization and Computing (CEMVICC), Faculty of Sciences and Technology (FACYT), University of Carabobo, Venezuela.

[9] 9. Joshua Dennis Booth, Anirban Chatterjee, Padma Raghavan, Michael Frasca; A Multilevel Cholesky Conjugate Gradients Hybrid Solver for Linear Systems with Multiple Right-hand Sides, Department of Computer Science and Engineering, The Pennsylvania State University

[10] 10. Juan C. Pichel, Francisco F. Rivera, Marcos Fernndez , Aurelio Rodrguez; Optimization of sparse matrixvector multiplication using reordering techniques on GPUs, Electronics and Computer Science DptUniversidade de Santiago de Compostela, Spain Galicia Supercomputing Center (CESGA), Santiago de Compostela, Spain.

[11] 11. Metcalf, Michael/Reid, John; FORTRAN 90/95 explained Ed.-.2, Oxford/Oxford University Press/1999.

[12] 12. MARTIN, Harold C / Carey, Graham F; Introduction to finite element analysis: Theory and application, New York/McGraw Hill/1973.

[13] 13. MujahedEleyat, and Lasse Natvig; Implementation of a linear programming solver on the Cell BE processor, Miriam AS, Violgata 8, N-1776 Halden, Norway Department of Computer and Information Science, Norwegian University of Science and Technology, SemSlandsvei 7-9 Glshaugen, 7034 Trondheim, Norway.

[14] 14. Norrie, D H/De Vries, G; Introduction to finite element analysis, New York/Academic Press/1978.

[15] 15. Ortega, James M; Introduction to FORTRAN 90 for scientific computing, New York/Saunders College Publishing/1994.

[16] 16. Prof. Michael T. Heath; Parallel Numerical Algorithms, Department of Computer Science University of Illinois at Urbana-Champaign.

[17] 17. R. R. Khazal and M. M. Chawla; A Parallel Cholesky Algorithm For The Solution of Symmetric Linear systems.