Engineering with Computers (1996) 12: 63–83
© 1996 Springer-Verlag London Limited

Engineering with Computers

Interactive–Adaptive Geometrically Nonlinear Analysis of Shell Structures

Kjell Magne Mathisen and Knut Morten Okstad
Department of Structural Engineering, The Norwegian Institute of Technology, Trondheim, Norway

Abstract. *This paper presents an investigation of interactive–adaptive techniques for nonlinear finite element structural analysis. In particular, effective methods leading to reliable automated, finite element solutions of nonlinear shell problems are of primary interest here. This includes automated adaptive nonlinear solution procedures based on error estimation and adaptive step length control, reliable finite elements that account for finite deformations and finite rotations, three-dimensional finite element modeling, and an easy-to-use, easy-to-learn graphical user interface with three-dimensional graphics. A computational environment, which interactively couples a comprehensive geometric modeler, an automatic three-dimensional mesh generator and an advanced nonlinear finite element analysis program with real-time computer graphics and animation tools, is presented. Three examples illustrate the merit and potential of the approaches adopted here and confirm the feasibility of developing fully automated computer aided engineering environments.*

Keywords. Adaptive mesh refinement; Adaptive solution procedures; Graphic visualization tool; Interactive-adaptive integrated system; Nonlinear finite element analysis; Rezoning; User-interface

1. Introduction

Since the development of the first 'general purpose' finite element (FE) computer programs in the 1960s, the finite element method (FEM) has become the main tool in computational structural mechanics (CSM). The success of the method is manifested by the number of program packages available (more than 500 [1]), the number of FE users (more than 20 000 [2]), and the amount of money spent on finite element analysis worldwide (more than $500 million annually [2]).

The computing environment has changed significantly since this tool's inception, when batch processing on stand-alone computers was the norm. Today, the users have access to a network of almost unlimited hardware resources; processors, memory, mass storage, graphics and other I/O devices. Some of these resources are collected into platforms specialized in certain functions: (1) Supercomputers or mini-supercomputers with relatively few (less than 100) complex high performance processors for computationally intensive operations, (2) mainframe-type computers with many processors that support a high level of parallelism (more than 1000 processors), for large problems that may be split into smaller problems, and (3) high-powered personal workstations for data generation, component analysis, graphic visualization and coarse grained distributed processing. Regardless of the program or the problem being solved, a finite element analysis (FEA) involves the following three main steps:

(1) FE modeling including geometry definition, initial mesh definition, prescription of loads and boundary conditions, definition of element and material properties, problem setup and output requests.
(2) FEA which involves assembly of element contributions and solving a resulting set of linear equations.
(3) Result generation which involves compilation and presentation of the results of the analysis in a form which is meaningful and provides insight to the analyst.

The steps (1) and (3) are usually referred to as the 'pre- ' and 'post-processing' steps respectively. These steps are today usually carried out in separate programs tailored with special capabilities for building FE models and reviewing analysis results [3, 4, 5].

In order to carry out nonlinear analysis of complex problems, the FEA module (step 2) has to include a complete suite of capability tools including; full treatment of nonlinear statics including bifurcation and post-buckling, linear and nonlinear transient dynamics, random vibration and natural frequency

Correspondence and offprint requests to: Prof. K. M. Mathisen, Department of Structural Engineering, The Norwegian Institute of Technology, N-7034 Trondheim, Norway.

response, a complete set of elements and material models accounting for finite rotations and finite deformations. In order to exploit the current hardware and obtain computational efficiency, the inner-loops in the FE calculations should be vectorized as well as parallelized [6, 7]. Recent advances in sparse matrix technology should also be incorporated in order to reduce the CPU and storage costs during the equation solving [8, 9].

This paper presents an investigation of *interactive–adaptive* methods for nonlinear finite element analysis (NFEA) of complex structural problems. 'Interactive–adaptive' methods are those in which the FE model and the algorithmic parameters are automatically selected or changed by the program or the user during the analysis itself. Large-scale NFEA of such problems, calls for supercomputer power in combination with supergraphical three-dimensional (3D) real-time graphics in order to be carried out interactively.

The use of interactive real-time graphics in NFEAs has heretofore been mainly restricted to the 'pre- ' and 'post-processing' steps, respectively. Although interactive graphics in these steps represents a significant improvement in productivity over earlier techniques, in which these steps were performed manually, it does not address the process of the NFEA (step 2) itself. The division of a NFEA into three distinct steps may be appropriate for the way a NFEA usually is carried out today, for which step 2 is carried out as batch jobs. However, for an adaptive NFEA these three steps have to be tightly integrated, in that typical functions of the 'pre- ' and 'post-processing' steps have to be performed during the analysis itself.

Traditional NFEA based on a batch approach, requires globally convergent solution algorithms that are self-adaptive and handle computational difficulties (bifurcation, limit and turning points, zero pivots, etc.) without diverging or spending an excessive amount of computer time. Even though the emerging automatic and self-adaptive solution algorithms [10, 11] improve the efficiency and the convergence of a NFEA, they do not overcome the difficulties completely. Moreover, conventional batch procedures, even when applied successfully, do not actively promote the understanding of the nonlinear structural behavior. In addition, all batch procedures, self-adaptive or not, cannot compete with interactive procedures in a CSM research environment where new algorithms, elements, material models, equation solvers and other computational strategies are being developed and tested.

In this work, some new concepts of interactive–adaptive analysis of nonlinear structural problems are studied, developed and implemented in an integrated system. Section 2 gives a description of a computational environment tailored for interactive–adaptive NFEA. This section is followed by a presentation of the basic steps involved in an adaptive nonlinear solution procedure. In particular this study focuses on an *h*-adaptive version of the FEM applied to nonlinear shell analysis. The paper also examines easy-to-use, easy-to-learn graphical techniques as means to monitor the results and to intervene and control the adaptive analysis in real computing time. In Section 3 a practical implementation of an integrated system that takes advantage of both a supercomputer and a supergraphic workstation in a heterogeneous distributed environment is presented. The possibilities for interactive–adaptive nonlinear analyses are also presented. These analyses are carried out through a seamless integration of an automatic 3D mesh generator, a highly modular general purpose nonlinear FE program, and a high-performance visualization tool. This Section is concluded by some examples that illustrate and evaluate these techniques in *h*-adaptive nonlinear analysis of three shell type problems.

To a certain extent the investigation of this paper is related to the available hardware configuration at the Department of Structural Engineering at the Norwegian Institute of Technology (NTH). The typical graphical workstation considered here must be able to visualize color 3D analysis results in real time. In order to give a realistic on-screen representation, an Iris Indigo XZ with a 19 inch (1280 × 1024) monitor, 64 MByte RAM and 24 Bits graphics with 24 Bits Z-buffer from Silicon Graphics Computer Systems has been considered in this study. For small to moderately sized nonlinear shell type problems, which typically consist of less than 10 000 degrees of freedom, the analysis program may be executed locally on the workstation. Larger problems are currently being analyzed as distributed tasks at other workstations in our network or on a CRAY Y-MP/464, that is located at NTH in Trondheim.

2. Conceptual Description of a Computational Environment for an Interactive–Adaptive Integrated System

Development of an interactive–adaptive integrated system for NFEA requires a combination of a variety of computer tools. These tools should provide the following basic functions:

(1) A visually menu-driven easy-to-use, easy-to-learn user interface that controls all functions of the integrated system.
(2) A 3D geometric modeler in which the complete

geometry, the material properties, the boundary conditions and the loads can be built up quickly and easily.

(3) An automatic 3D mesh generator that effectively and efficiently can create a FE discretization for all kinds of structures.
(4) A geometry database that is precisely and effectively linked to the geometric modeler and the mesh generator.
(5) Support for NFEA through form-driven analysis setup.
(6) Interactive communication with a highly modular NFEA program allowing the user to intervene and modify the algorithmic parameters as well as the FE model during the analysis itself.
(7) Distributed processing taking advantage of the most suited hardware at any time.
(8) A 3D color graphic visualization tool that is seamless wired to the user interface and communicates with the analysis program interactively.
(9) Plotting-functions for monitoring the development of time varying results, characteristic analysis parameters and convergence statistics for the solution process.

2.1. User Interface for the Integrated System

The user interface should provide a complete CSM problem-solving environment that integrates the typical 'pre-processing' functions, which include geometry definition, initial mesh definition, prescription of boundary conditions and loads, with problem setup, execution control and advanced real-time visualization. To use an analogy, we may imagine the structural engineer as being in the 'driver's seat' relative to his computing environment, in which 3D visualization of the physical problem, real-time animation, and color will create for him a virtual reality of the physical problem that is simulated numerically. The user should also be able to converse with the system in several engineering 'jargons' and at several levels of expertise.

To provide a user friendly system, the user interface must be intuitive and easy-to-use and common for the whole system, such that it is also easy-to-learn. The user interface should also provide a menu system that guides the user through all operations. A beginning or occasional user should be able to access the system through 'short' tailored menus to simplify the selection of options available in the system. At the same time an experienced user should have the possibility of accessing the full menus or even bypass the menus by typing commands directly. The user interface should also provide an on-line help system. This help system should preferably contain examples that illustrate how the different features of the system can be used.

Another important requirement is that the user interface should be built around a standard that is supported on most hardware platforms.

2.2. Geometric Modeling

In performing a FEA, today's computer aided engineering (CAE) analyst typically consumes a significant amount of time on model construction [12]. Reducing model generation time is therefore a key issue for increasing the productivity. In contrast the design engineer has benefited significantly from computer aided design (CAD) systems. By integrating the geometric modeler with the FE modeler similar benefits may be provided to the CAE analyst. It is important to note that the term CAD used in this context, refers to a full function solid modeler based, computer aided design system [13]. In such a system, it is imperative that wireframe, surface, and solid modeling tools coexist, to allow for *dimensional reduction*. 'Dimensional reduction' refers to the representation of a thin solid by a surface, or a long and narrow solid by a wireframe entity.

A NFEA model consists of a FE mesh, loads, boundary conditions, material properties, and solution algorithm control parameters. Traditionally the functional assignment (loads, boundary conditions and material properties) are applied to nodes and elements. However, in an automated adaptive process the functional assignments should be moved from the FE model to the geometry. As a result, when the user or the adaptive algorithm changes the FE model, the often complex array of functional assignments need not be recreated, but automatically passed to the underlying FE model.

Perhaps the greatest advantage of geometry based attributes relates to the natural and intuitive improvement in the way the analysts think about physical systems that they are attempting to simulate. By using geometry based features, the analysts can begin to focus on the essential nature of the physical problem at hand, rather than get lost in the details of how to transform the physical problems into a FE model.

2.3. Automatic Mesh Generation

Traditionally, an automatic mesh generation system is a system which given sufficient and unique geometry is capable of producing a valid FE model without user intervention [14]. This definition is necessary, but not sufficient to convey the intrinsic concepts of automatic adaptive mesh generation based on error estimation. However, the definition does point out the complete reliance on geometry of the automatic mesh generation system. The geometry must contain all the information necessary to produce the FE model. In addition the mesh generator must also provide capabilities for automatic modification of mesh density and element shape such that, e.g., an analyst can begin with a coarse mesh and allow adaptive meshing to refine the model only in areas where more elements are needed for accuracy.

There has been a significant amount of work done in the field of automatic mesh generation based on automatic modification of mesh density. A survey of different approaches for unstructured grid generation has been presented by Shephard [15]. One of these approaches is the method advocated by Peraire *et al.* [16], the advancing front method. Another successful approach is the finite octree technique, presented by Shephard and Georges [17].

2.4. Geometry Databases

A single geometry database that is common for the CAD system, used to construct the geometry and to create the functional assignments, and the automatic mesh generation system, is one of the major technological components of an integrated system. By letting the mesh generator access the geometry database directly, the need to translate, transfer or recreate the geometry is eliminated. Since prescribed loads, boundary conditions and material properties all work directly on the CAD geometry, they may be passed directly from the common database to the FE model without any transformations. An example of a standard for engineering data exchange is the STEP standard (Standard for the Exchange of Product model data) [18, 19]. Another standard is IGES (Initial Graphics Exchange Specification) which was developed in the USA and is used for the exchange of engineering drawings and geometry throughout the world.

2.5. Analysis Setup for NFEA

The user interface should integrate the NFEA program by providing a direct interface where the user could create all necessary input that is needed for a nonlinear analysis. After the user has selected the analysis type (linear or nonlinear, static or dynamic, time domain or frequency domain, random or deterministic, transient or steady state, etc.), the user should only be requested information that is necessary for the selected analysis job. The user interface should function as an intelligent user interface providing expert advice on standard practice, such that it can lead a user through the complete analysis setup and provide default values for each possible choice in the menus. The user interface should include checks on incomplete or inconsistent data input, such that errors could be detected interactively and the user will be notified before the analysis starts.

In addition this part of the user interface should also include capabilities for creating and verifying (through, e.g., interactive visualization) nonlinear loads, nonlinear boundary conditions, nonlinear material models and the mathematical discretization model (element type; e.g., Mindlin/Reissner or Kirchhoff type plate theory, linear or quadratic element, etc.).

2.6. Interactive Communication with the NFEA Solver

No matter how automated the process becomes, simulation using the FEM is still an approximation of a real world problem. Fundamental simplifications and assumptions are thus needed throughout the numerical simulation. Experience and expertise is required to select an idealized mathematical model for the physical problem at hand. Adaptive meshing through error estimation provides guidance and automation of mesh density. However, it must be kept in mind that engineering insight is irreplaceable in the process of numerical simulation of a complex structural problem. Consequently the CAE monitor should provide interactive communication with the NFEA program in real computing time throughout the whole analysis, such that the analyst at any stage of the process can:

(1) Monitor any results he wants.
(2) Control the flow of the analysis.
(3) Change any parameters that he wants (e.g., analysis type, step length, convergence tolerance, discretization error tolerance, etc.).
(4) Modify the problem definition (e.g., element type, loads, boundary conditions, material models, etc.).

2.7. Distributed Processing

A CAE environment of the 1990's should be able to take advantage of more than one computer system in order to produce a result. However, there must be some good reasons why one should use more than one computer system to tackle a problem. To a computer scientist, simply being able to do such a thing may be good enough reason. But, for a structural engineer, the following may be considered as reasons:

(1) Distributed processing can improve turn-around time.
(2) Some computer systems have specialized hardware that makes them ideal for some tasks, but undesirable for others.
(3) A computer system may simply not be capable of solving all aspects of our problem.
(4) There may be economical reasons to utilize more than one computer system.

While this list by no means is meant to be complete, it does cover the two main reasons of why distributed processing is desirable; time and money.

An interactive–adaptive nonlinear structural analysis of a complex problem may take a lot of computing time. If the problem takes too long a time (wall clock time) to be run on a local node, distributed processing may be the only solution. In general as a rule of thumb, a distributed task should be done at least five times faster on the distributed node as compared with the local node, before it is distributed. However, this general rule of thumb does not apply to this special case. In designing the 'total system', the following goals should be kept in mind:

(1) Take advantage of distributed processing (e.g., computational power of a supercomputer) for the task of running the analysis program only.
(2) Take advantage of the local node (e.g., a graphics workstation with hardware graphics) for running the user interface and the visualization software.
(3) Minimize the network traffic by utilizing high level communication between the distributed processes (e.g., between a workstation and a supercomputer) and by distributing the data in an optimal manner.

2.8. Graphic Visualization Tool

The graphic visualization tool should be seamless wired to the user interface and have direct access to the FE model and results in real computing time. The visualization tool should include capabilities for real-time visualization of 3D color graphics, which includes animation of models or results in combination with a variety of display options.

Since communication is the most fundamental component in distribution processes, the visualization tool should accept communication on different levels, e.g., TCP/IP socket level communication, direct piping and communication through files. This makes it possible to carry out real-time visualization as well as *a posteriori* animation of results stored on files.

In order to carry out visualization of 3D color graphics in real computing time, the visualization software should take full advantage of any available hardware graphics on high performance workstations. A powerful visualization tool should also allow for easy access to rotation and zooming to enable the user to get an overall picture as well as to identify critical information immediately.

2.9. Plotting Functions for Time Varying Results

In order for the analyst to interpret the analysis correctly, results developed during the simulation must be presented in an understandable form. In addition to 3D color graphics, the user interface should include capabilities for:

(1) Two-dimensional function plotting for time varying results together with user defined functions.
(2) Three-dimensional function plotting across model geometry.
(3) Data manipulation capabilities that combine and/or manipulate results via predefined mathematical functions before presentation.
(4) Selective sorting and tabular reporting of results.
(5) Numerical and visual reporting of selected solution control quantities (e.g., the number of iterations spent per load step, the displacement, force and/or energy norms, convergence norm during iterations, current iteration number, load step number and time, etc.).

3. Solution Procedures

The governing equilibrium equation of a nonlinear FE problem is often written compactly as

$$\mathbf{R}(\lambda, \mathbf{r}) = \mathbf{0} \tag{1}$$

where $\mathbf{R}(\lambda, \mathbf{r})$ denotes the residual force vector at a certain configuration represented by a load level parameter, λ, and a vector of nodal displacements, $\mathbf{r}$.

The goal of a nonlinear solution procedure is thus to solve (1) for the unknowns, $\mathbf{r}$, within a certain range of the load parameter, λ. This is usually done through incrementation and equilibrium iterations, and is based on a linearized version of (1), which reads

$$\mathbf{K}\,\Delta\mathbf{r} - \mathbf{P}\,\Delta\lambda = \mathbf{R}(\lambda, \mathbf{r}) \tag{2}$$

where $\mathbf{K} = \left.\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\right|_{\varepsilon=0} \mathbf{R}(\lambda, \mathbf{r} \oplus \varepsilon\Delta\mathbf{r})$ and $\mathbf{P} = \frac{\partial}{\partial\lambda}\mathbf{R}(\lambda, \mathbf{r})$ are respectively the tangential stiffness matrix and the tangential load vector.

For a fixed value of λ and $\mathbf{r}$, Eq. (2) is a linear system of equations consisting of n_{dof} equations, but $n_{\mathrm{dof}} + 1$ unknowns. It is therefore augmented by an additional

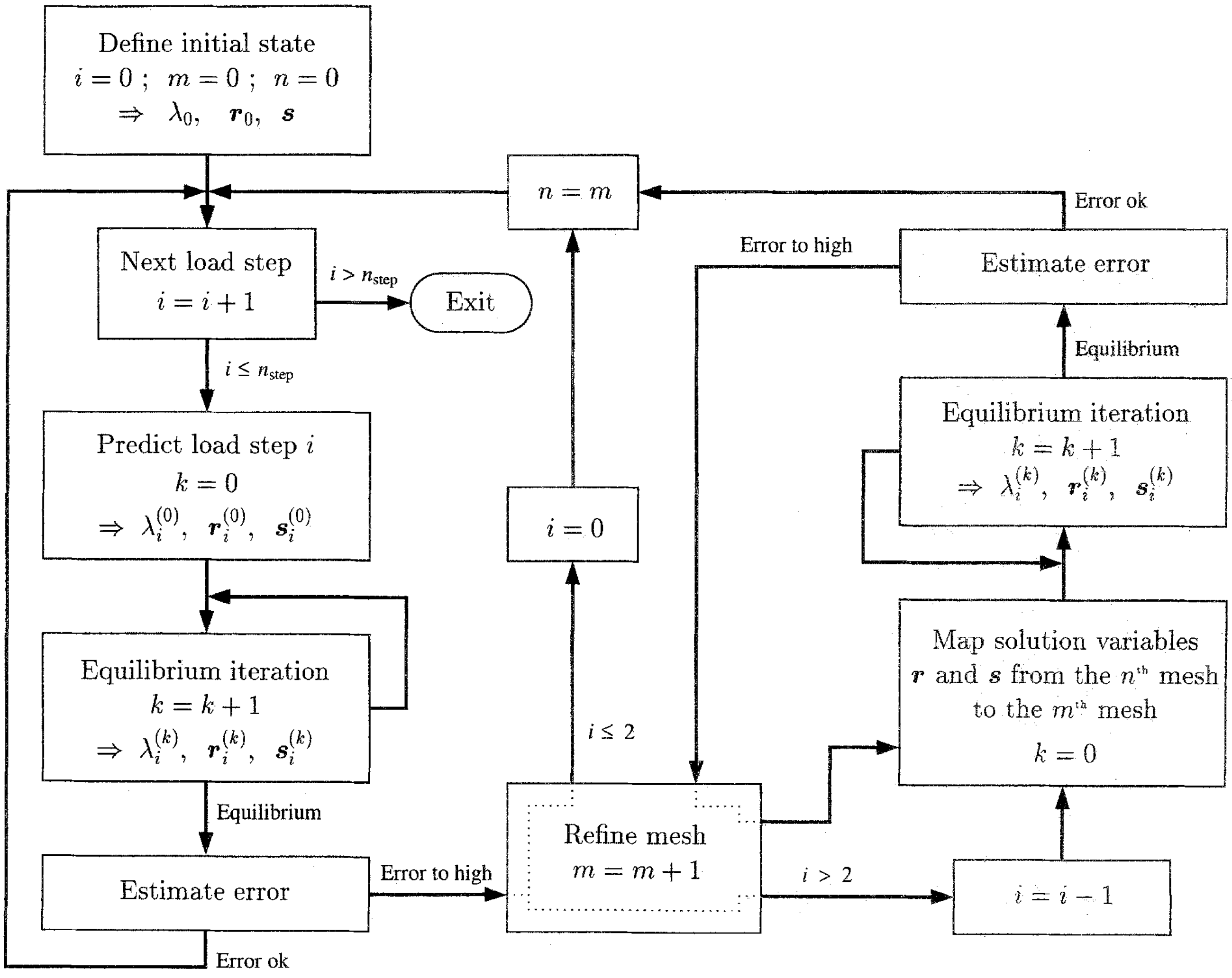


Fig. 1. Algorithmic overview of the adaptive nonlinear solution procedure.

constraint equation relating the load increment, $\Delta\lambda$ and the displacement increment $\Delta\mathbf{r}$. Several alternatives exist for this additional constraint. Herein, it is taken as

$$\|\Delta\mathbf{r}\|^2 + \Delta\lambda^2 = \Delta s^2 \tag{3}$$

where Δs is a prescribed increment in the arc-length of the generalized load-displacement curve. Using Eq. (3) results in a solution procedure commonly known as the arc-length method [20, 21].

3.1. Adaptive Solution Procedure

In linear FEA, adaptivity is usually introduced through a mesh refinement iteration loop on top of the solution algorithm itself. That is, the FEA is performed on successively refined meshes until the prescribed tolerance requirements are satisfied. The mesh refinement is guided by element level error indicators which reflect the distribution of the error over the FE mesh. This concept may be employed in NFEA as well. However, due to their incremental nature, adaptive nonlinear solution procedures should have the mesh refinement iteration loop on the load step level.

Figure 1 gives an algorithmic overview of an adaptive nonlinear FE solution procedure based on standard incremental–iterative methods. Here, s denotes some solution dependent state variables, e.g., the stresses, i is the load step counter, k is the iteration counter and m and n are mesh counters.

After convergence has been obtained on a certain load step, an error estimation is performed before continuing on the next load step. However, if the prescribed error tolerance is violated at a certain load step the normal execution is halted and the FE mesh is refined based on the computed elemental error indicators.

Before the solution procedure continues with the new mesh, the values of the state variables must be transferred from the old to the new FE mesh. This process is in the literature known as *rezoning* and is discussed in some more detail below. Note that it is the results of the load step just before the one that violated the error tolerance that should be transferred, since these are the latest results with acceptable error. The load step counter, i, is therefore decremented by one. This load step is then recalculated by means of equilibrium iterations with the new FE mesh and the solution procedure resumes normal execution starting with the load step that triggered the mesh refinement. Note that if the error violates the prescribed tolerance in the first or the second load step no solution transfer is needed. In these cases the analysis is restarted from the beginning on the first load step with the new mesh.

The term 'Refine mesh' in Fig. 1 is assumed to cover mesh *coarsening* also. The ability of mesh coarsening is perhaps even more important in NFEA than in the linear case, since the optimal computational model may change during a nonlinear analysis as a result of the 'time variation' of the applied loads, or due to geometric changes in large deformation analyses. A region of the mesh that has been refined at an early stage of the analysis may thus become less critical on a later stage. The refinement procedure should then be able to change to a coarser mesh in this region in order to provide the most economical analysis.

In the predictor step indicated in Fig. 1, a first estimate of the load parameter increment, $\Delta\lambda_i^{(0)}$, and the displacement increment, $\Delta\mathbf{r}_i^{(0)}$, are first determined and then the state variables are updated through $\lambda_i^{(0)} = \lambda_{i-1} + \Delta\lambda_i^{(0)}$ and $\mathbf{r}_i^{(0)} = \mathbf{r}_{i-1} \oplus \Delta\mathbf{r}_i^{(0)}$. The displacement increment $\Delta\mathbf{r}_i^{(0)}$ is found by solving the linearized equilibrium Eq. (2) at the previous computed configuration, i.e.,

$$\Delta\mathbf{r}_i^{(0)} = \Delta\lambda_i^{(0)}\bar{\mathbf{r}}_i^{(0)} + \tilde{\mathbf{r}}_i^{(0)} \tag{4}$$

where $\bar{\mathbf{r}} = \mathbf{K}^{-1}\mathbf{P}$ and $\tilde{\mathbf{r}} = \mathbf{K}^{-1}\mathbf{R}$ are respectively the tangential and residual displacement vectors. The load increment $\Delta\lambda_i^{(0)}$ is found by imposing the arc-length constraint (3) on the incremental displacement (4), while neglecting the residual $\tilde{\mathbf{r}}_i^{(0)}$. This leads to an expression

$$\Delta\lambda_i^{(0)} = \pm\frac{\Delta s_i}{\sqrt{1 + \dfrac{1}{n_{\text{dof}}^i\,\|\bar{\mathbf{r}}_i^{(0)}\|^2}}} \tag{5}$$

The arc-length increment of the first load step, Δs_1, is determined by specifying $\Delta\lambda_1^{(0)}$ and then using the inverse of (5). For the succeeding load steps, the arc-length increments are adapted based on the number of iterations spent on the previous steps in order to obtain convergence, i.e.,

$$\Delta s_i = \sqrt{\frac{n_{\text{iter}}^{i-1}}{n_{\text{iter}}^{\text{d}}}}\,\Delta s_{i-1} \tag{6}$$

where n_{iter}^i is the number of iterations spent on load step i, and $n_{\text{iter}}^{\text{d}}$ denotes the desired number of iterations per step.

The sign of $\Delta\lambda_i^{(0)}$ may be determined based on the characteristics of the tangential stiffness matrix, $\mathbf{K}$, or alternatively by claiming that the current predictor step must form a positive dot-product with the previous predictor step. Note that the Euclidean norm of the tangential displacement vector in (5) is scaled by the factor $1/\sqrt{n_{\text{dof}}^i}$ to account for a varying number of degrees of freedom during the analysis. If this scaling factor is omitted, the use of (6) will result in a too small step size in the first load step after a mesh refinement, since $\|\bar{\mathbf{r}}\|$ typically grows with increasing number of degrees of freedom.

The equilibrium iterations (often referred to as the corrected step) consist of determining iterative corrections $\delta\lambda_i^{(k)}$ and $\delta\mathbf{r}_i^{(k)}$ which update the state variables through $\lambda_i^{(k)} = \lambda_i^{(k-1)} + \delta\lambda_i^{(k)}$ and $\mathbf{r}_i^{(k)} = \mathbf{r}_i^{(k-1)} \oplus \delta\mathbf{r}_i^{(k)}$. The displacement correction is determined in a similar way as for the predictor step, i.e.,

$$\delta\mathbf{r}_i^{(k)} = \delta\lambda_i^{(k)}\bar{\mathbf{r}}_i^{(k)} + \tilde{\mathbf{r}}_i^{(k)} \tag{7}$$

The load parameter correction, $\delta\lambda_i^{(k)}$, may be determined by utilizing the constraint equation (3), either directly [22] or in a linearized form [23]. Herein, however, we use an orthogonal trajectory method [24], where the idea is to claim that the iterative correction $(\delta\mathbf{r}_i^{(k)}, \delta\lambda_i^{(k)})$ should be orthogonal to the current tangent $(\bar{\mathbf{r}}_i^{(k)}, 1)$, i.e.,

$$\begin{Bmatrix}\delta\mathbf{r}_i^{(k)}\\ \delta\lambda_i^{(k)}\end{Bmatrix}^{\mathrm{T}}\begin{Bmatrix}\bar{\mathbf{r}}_i^{(k)}\\ 1\end{Bmatrix} = 0 \tag{8}$$

When combined with Eq. (7) this orthogonality condition results in

$$\delta\lambda_i^{(k)} = -\frac{\bar{\mathbf{r}}_i^{(k)\mathrm{T}}\tilde{\mathbf{r}}_i^{(k)}}{1 + \|\bar{\mathbf{r}}_i^{(k)}\|^2} \tag{9}$$

The main advantage of this approach compared to methods based on the arc-length constraint (3) is that there is no need to calculate the updated displacement increment $\Delta\mathbf{r}_i^{(k)} = \Delta\mathbf{r}_i^{(k-1)} \oplus \delta\mathbf{r}_i^{(k)}$. This is essential when considering the restart of the solution procedure on a new mesh after a mesh refinement has been performed. In these cases the initial displacement increment, $\Delta\mathbf{r}_i^{(0)}$, on the new mesh is unknown since the predictor step then is represented by the solution transfer operation.

3.2. Error Estimation

The spatial error in the FE solution at a certain load step is herein estimated by the widely used

Zienkiewicz–Zhu [24] error estimator. This simple estimator is based on comparison of the FE stresses, $\mathbf{s}^h$, which are typically discontinuous across the element interfaces, with a recovered continuous stress field, $\mathbf{s}^*$. The error is measured on the element level through an energy norm expression of the form

$$\|\mathbf{e}_e^*\|_E^2 = \frac{1}{2}\int_{\Omega_e} (\mathbf{s}_e^* - \mathbf{s}_e^h)^T \mathbf{C}^{-1} (\mathbf{s}_e^* - \mathbf{s}_e^h)\, d\Omega \quad (10)$$

where $\mathbf{C}$ is the constitutive matrix and Ω_e denotes the element domain. An estimate of the global error in energy norm is then obtained by summation of the local elemental contributions

$$\|\mathbf{e}^*\|_E^2 = \sum_{e=1}^{n_{el}} \|\mathbf{e}_e^*\|_E^2 \quad (11)$$

where n_{el} denotes the number of elements.

For the example problems presented in this paper, the stress recovery is based on a global least squares fit projection of the FE shell stress resultants onto a set of C^0 continuous interpolation functions [26]. However, we are currently working with alternative recovery procedures for shell problems based on the superconvergent patch recovery technique [27, 28] and plan to present results with such methods in future publications.

3.3. Adaptive Mesh Refinement

Output from the error estimation procedure is a refinement indicator for each finite element telling how much the size of that element should be reduced (or increased) in order to obtain the optimal mesh meeting the desired accuracy. In our work we use the refinement indicator

$$\phi_e = \left(\frac{1}{\eta_p}\sqrt{\frac{n_{el}\|\mathbf{e}_e^*\|_E^2}{\|\mathbf{u}_e^h\|_E^2 + \|\mathbf{e}_e^*\|_E^2}}\right)^{1/p} \quad (12)$$

where p is the polynomial order of the finite element type used, $\|\mathbf{u}_e^h\|_E$ is the energy norm of the FE solution (the strain energy), and η_p is the permissible relative error in energy norm. Expression (12) is obtained by assuming that the error has a rate of convergence equal to p and by claiming that the global error being equally distributed among the elements in the optimal mesh [25, 29]. If the characteristic size of an existing element is h_e, the size of the new elements in the region covered by that element should then be

$$h_{new} = \frac{h_e}{\phi_e} \quad (13)$$

When the estimated global error is higher than the permissible error, the analysis program halts at that load step and passes the control to the mesh generator that will generate a new mesh guided by the computed refinement indicators, ϕ_e. That is, in regions where $\phi_e > 1$ the mesh is refined whereas in other regions with $\phi_e < 1$ the mesh is coarsened or de-refined.

In the present investigation an unstructured meshing approach based on the finite octree technique [17] is used for the generation of meshes with triangular shell elements. When quadrilateral elements are desired, a conversion of the triangular mesh into a quadrilateral mesh is performed at the end.

Mesh refinements and de-refinements are carried out by complete regeneration of the whole mesh. The element level refinement indicators are first mapped onto the underlying octree and the old FE mesh is removed. The octree is then refined/de-refined according to the refinement indicators and a new mesh is created based upon the refined octree. For further details about this mesh generator, the interested reader should consult Reference 17. The refinement/de-refinement procedures are fully described in Reference 30.

3.4. Solution Transfer Between Finite Element Meshes

An important but crucial step of an adaptive nonlinear solution process is the transfer of solution variables from the old mesh to the new mesh after a mesh refinement. Basically, this consists of the following four steps, of which the first three are illustrated in Fig. 2.

1. Define the state variables in the old mesh (i.e., the mesh from which the solution is to be transferred) by means of discrete variables located at the nodes, i.e., convert the element-wise quantities into the nodal ones.
2. For each node in the new mesh (i.e., the mesh to which the solution is to be transferred), interpolate values of the state variables at that node based on the nodal values of the old mesh.
3. Evaluate the necessary integration point values (including assumed strain point values for elements having such points) of the state variables for each new element by interpolation of the nodal values of the new mesh.
4. Perform equilibrium iterations on the new mesh to ensure that the transferred solution is in equlibrium with the external load.

The first step is trivial for geometrically nonlinear analyses where only the displacements and stresses need to be transferred. The displacement field is already defined through its nodal values from the FE solution itself, whereas nodal stress values are

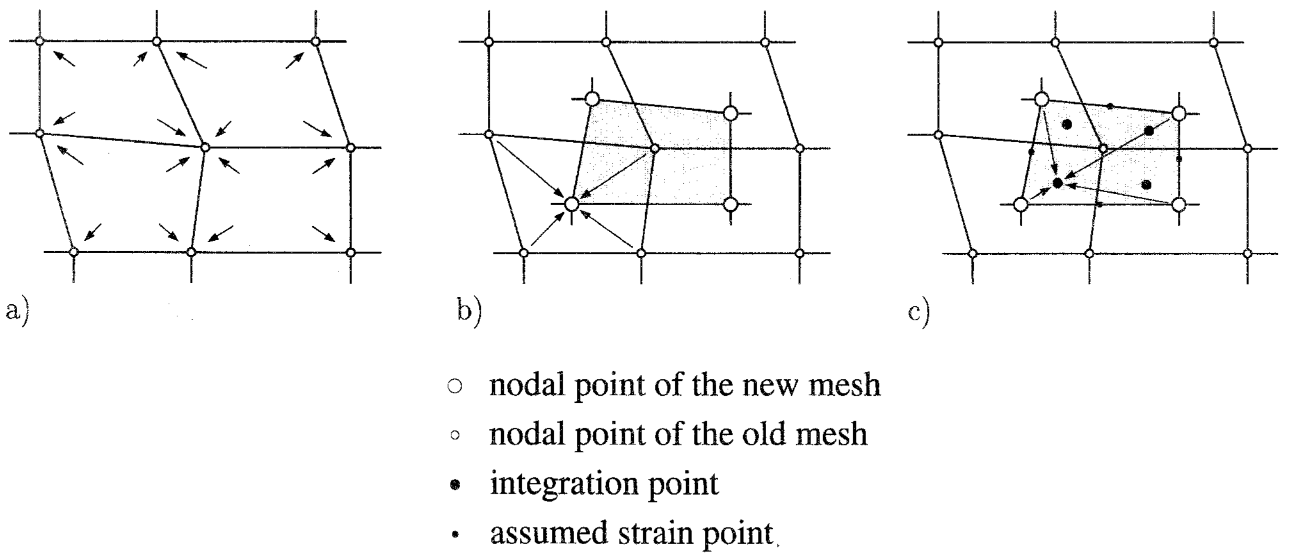


Fig. 2. The first three steps of the solution transfer process; (a) definition of nodal values in the old mesh, (b) calculation of nodal values in the new mesh, and (c) evaluation of integration (and assumed strain) point values.

computed by the stress recovery procedure needed in the error estimation process.

The second step of the solution transfer process is perhaps the most critical one when considering curved shell problems. Interpolating field variables from the old mesh to the new mesh requires a point in the new mesh to be associated with the element in the old mesh that contains the same material point. This is realized through an element search procedure in which an element in the old mesh that matches the location of the point in the new mesh is determined. Since the element shape functions are used to define the interpolation function of the field variables, a parametric inversion is needed to find the local parameters of the point with respect to the element in the old mesh it resides in. Full details on the second and third step of the solution transfer operation is given in References 30 and 31.

The equilibrium iterations that constitute the final step of the solution transfer operation are similar to those used in the original incremental–iterative procedure, but with one important modification. Recall that the tangential stiffness matrix, **K**, of a nonlinear FE problem is obtained through consistent linearization of the internal virtual work, and may formally be defined through

$$\delta\mathbf{r}^{\mathrm{T}}\mathbf{K}\,\Delta\mathbf{r} = \int_{\Omega} \{\delta\mathbf{e}{:}\mathbf{C}{:}\Delta\mathbf{e} + \Delta(\delta\mathbf{e}){:}\mathbf{s}\}\,\mathrm{d}\Omega \qquad (14)$$

where **e** and **s** are respectively the strain and stress tensors and **C** is the fourth order material tensor. The second term in the right-hand side of Eq. (14) represents the geometric stiffness and depends directly upon the current stress state **s**. Following a standard procedure these stresses are calculated based on the current displacement state. In the first iteration cycle after a remeshing this means the displacement field that is transferred from the old mesh. However, the present study has revealed that these stresses often are spuriously high, leading to erroneous geometric stiffnesses and failure of the iteration procedure. Therefore, we also transfer the recovered stress field, **s***, onto the new mesh and use these stresses in the geometric stiffness calculation in the first few iteration cycles.

4. Design, Implementation and Examples of an Interactive–Adaptive Integrated System

To verify the practical feasibility of the concept discussed in this paper, a fully integrated interactive–adaptive system has been designed and implemented. The adaptive nonlinear solution procedure presented above has been implemented in a fully vectorized version of the general purpose nonlinear FE program FENRIS (Finite Element NonlineaR Integrated System) [6, 32]. This program has a highly modular structure which allows for combination of different types of integrated nonlinear analysis, with particular capabilities for offshore structures. In order to improve the computational efficiency, a true sparse matrix solver has been implemented within FENRIS as another part of the present investigation [8, 9].

The developed interactive–adaptive environment for NFEA is based on full coupling of the extended

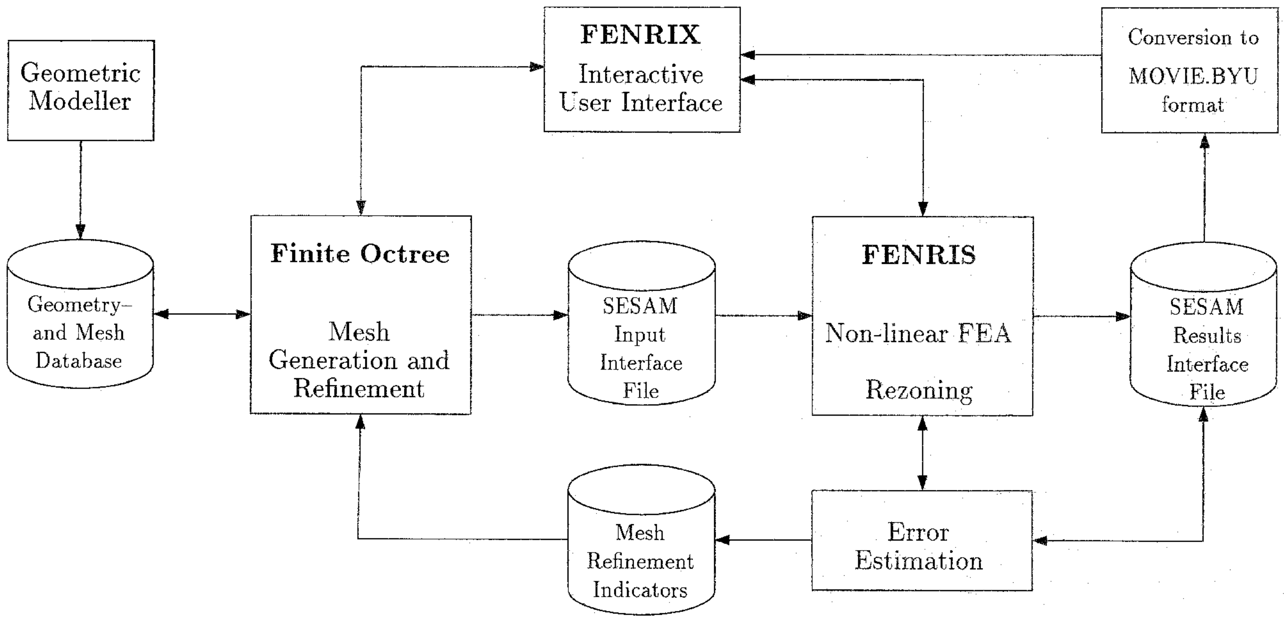

FENRIX
Interactive User Interface
Conversion to MOVIE.BYU format
Geometric Modeller
Geometry- and Mesh Database
Finite Octree
Mesh Generation and Refinement
SESAM Input Interface File
FENRIS
Non-linear FEA
Rezoning
SESAM Results Interface File
Mesh Refinement Indicators
Error Estimation

Fig. 3. Overview of the adaptive nonlinear FE analysis system.

version of FENRIS, the Finite Octree mesh generator [17], a high performance 3D visualization tool GLview [33] and a Motif-based graphical user interface FENRIX (Finite Element NonlineaR Integrated X-environment) [34], see Fig. 3. FENRIS and Finite Octree communicate through a well-defined interface, the SESAM Interface File format (SIF) [35], while GLview is compatible with the MOVIE.BYU file format. The communication between FENRIS and the user interface FENRIX is based on socket communication while between GLview and FENRIX it is based on UNIX *piping*. The data are transferred exclusively as pure ASCII text. Therefore, no extra data translation is needed when FENRIX and FENRIS are run on different hardware platforms. Most of the FE software and the mesh generator used herein is written in FORTRAN, while the user interface is implemented in C.

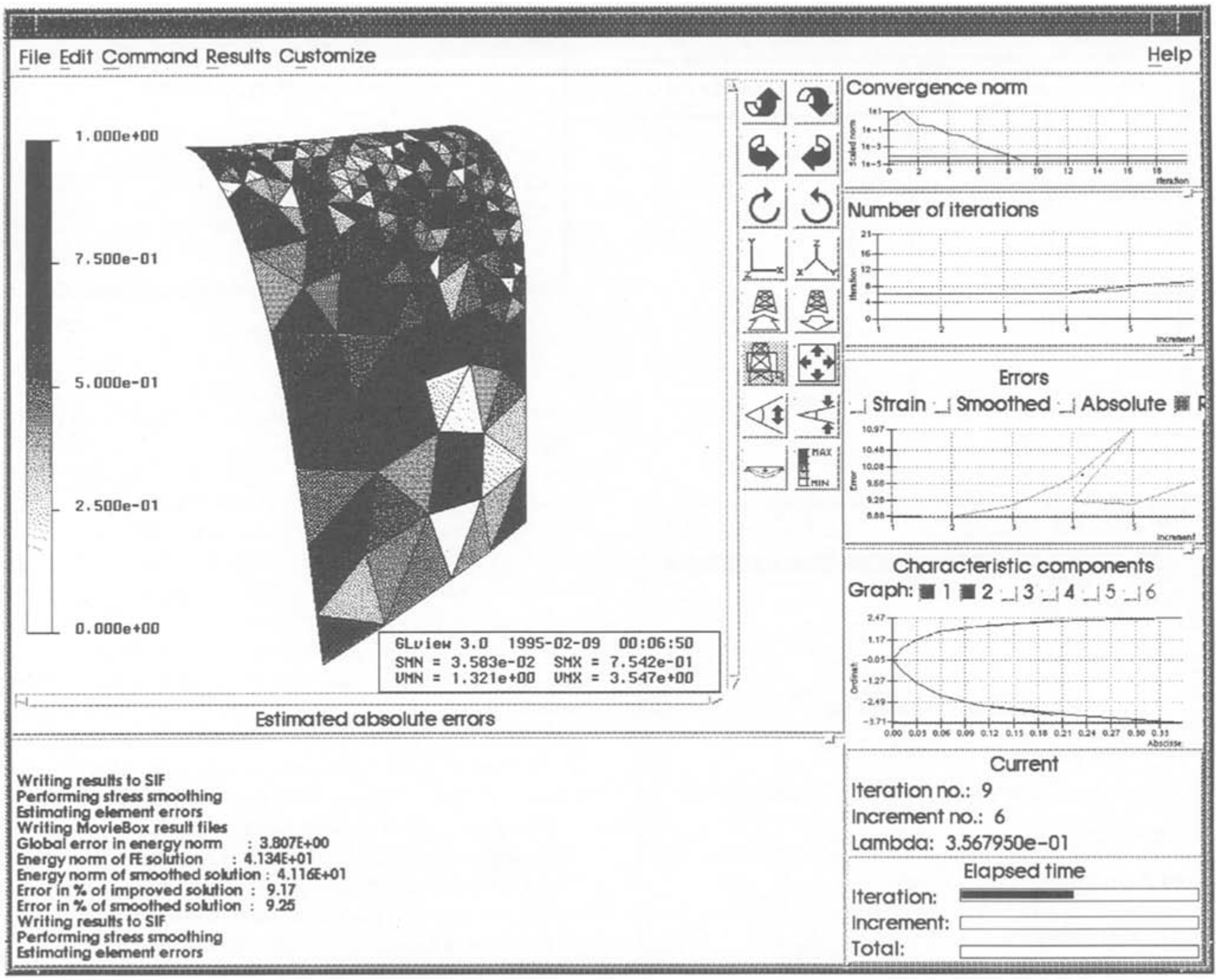
The main window of the Motif-based graphical user interface is presented in Fig. 4. On the right-hand side, six different windows are designed to monitor the performance of the solution process. The upper window shows the actual convergence norm versus the iteration number computed during current load increment. The prescribed convergence criterion is indicated in the same window as a full-drawn horizontal line. During the iteration process, many complex problems need to be closely monitored and have certain parameters adjusted in order to assure stability, convergence and good performance. By monitoring the performance of the convergence process an experienced user may interactively adjust the needed parameters to achieve better stability and speed up the solution process.

The second window from upper right, shows the number of Newton–Raphson iterations spent per load step. This information is crucial in order to select the most appropriate time/load step as the solution process develops. The next window shows the development of the total strain energy computed based on the FE stress field (Strain) and the recovered stress field (Smoothed) during the solution process, as well as the development of the estimated global error measured in the energy norm (Absolute and Relative). The user may select one or more of these quantities to be presented by simply pushing the 'push buttons' in this window.

The fourth control window is used to plot certain time varying quantities (such as displacement or reaction force components in specified points, etc.) versus time or any other time varying quantity. Up to six quantities may be monitored simultaneously. The fifth window contains information on current iteration number, load step number and load level. The last window simply shows an estimate of execution time (CPU time) relative to the iteration, load step and total solution time, respectively.

Each user may want to monitor different parameters. The control windows have been designed such that each user may customize his own environment by simply hiding any (or even all) of the control windows in order to increase the work space for the other windows.

The upper left window, the image window, is



```
GLview 3.0  1995-02-09  00:06:50
SMN = 3.583e-02  SMX = 7.542e-01
VMN = 1.321e+00  VMX = 3.547e+00
```

```
Writing results to SIF
Performing stress smoothing
Estimating element errors
Writing MovieBox result files
Global error in energy norm      : 3.807E+00
Energy norm of FE solution      : 4.134E+01
Energy norm of smoothed solution : 4.116E+01
Error in % of improved solution :  9.17
Error in % of smoothed solution :  9.25
Writing results to SIF
Performing stress smoothing
Estimating element errors
```

```
Current
Iteration no.: 9
Increment no.: 6
Lambda: 3.567950e-01
Elapsed time
Iteration:
Increment:
Total:
```

Fig. 4. The FENRIX user interface.

intended for 3D color graphic visualizaton in real computing time. This includes animation, fringe plots, vector plots, isocurves, etc. The lower left window reports messages from FENRIS during the execution. The amount of information received from FENRIS may be customized. Typically, this window reports what is going on in the analysis program at the moment, more detailed information regarding the iterative solution process and an echo of computed quantities (e.g., computed error estimates at the current load step, etc.).

The user may control the execution of the analysis program interactively by selecting any of the available options (Start, Stop, Step, Continue, Restart, Abort, Kill) in the **Command** menu shown in Fig. 5. A new analysis is started by selecting the **Start** option after having created the geometry in the geometric modeler and defined the analysis setup for the problem on an input file. The user may then choose to execute FENRIS on any available node in the network, see Fig. 6. Depending on the size of the FE model and the number of increments in the analysis, it may be preferable to execute FENRIX and FENRIS on two different computers. As seen from Fig. 6 the current load on all computers available in the network is monitored, and the user may choose the most appropriate node at the moment for his problem.

The analyst must indicate if he wants to run the analysis adaptively by pressing the **Adaptive** button in the middle of the host selection box. FENRIX will then first invoke the Finite Octree mesh generator in order to generate the initial FE mesh based upon data stored in the geometry database, before it invokes the FEA program. The FEA is now continued until the estimated global error violates the prescribed error tolerance. When this happens the analyst is prompted

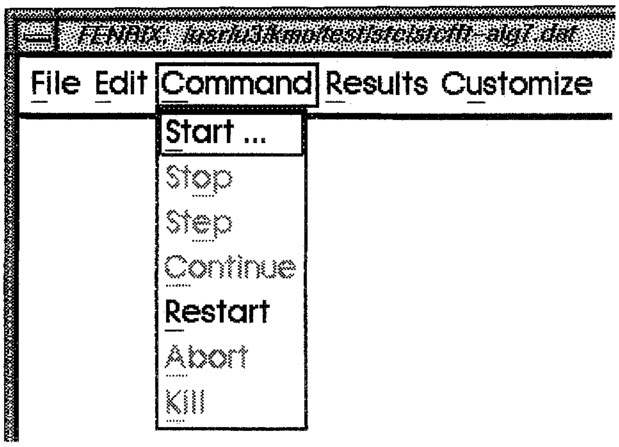
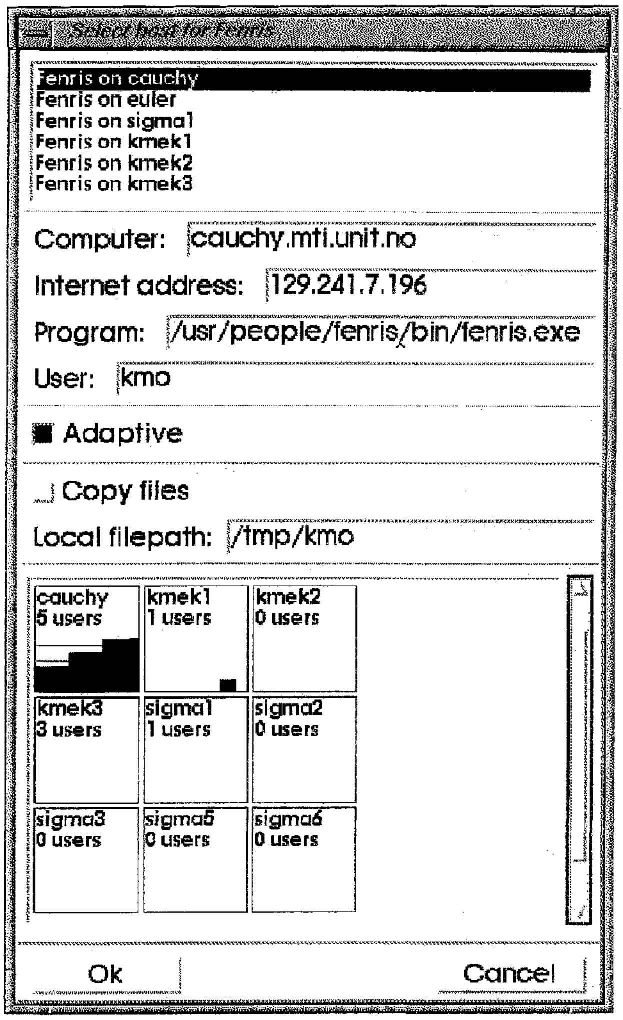


Fig. 5. The Command menu.



Fenris on cauchy
Fenris on euler
Fenris on sigma1
Fenris on kmek1
Fenris on kmek2
Fenris on kmek3

Computer: cauchy.mti.unit.no
Internet address: 129.241.7.196
Program: /usr/people/fenris/bin/fenris.exe
User: kmo

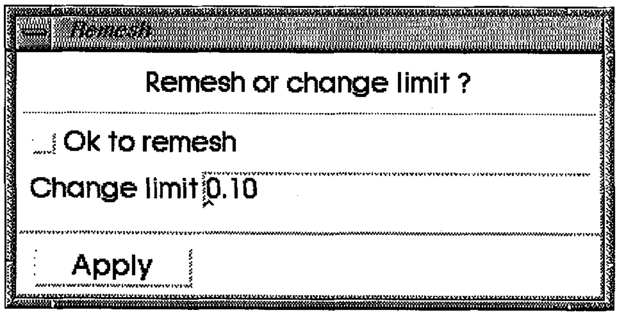
■ Adaptive

☐ Copy files

Local filepath: /tmp/kmo

cauchy 5 users	kmek1 1 users	kmek2 0 users
kmek3 3 users	sigma1 1 users	sigma2 0 users
sigma3 0 users	sigma5 0 users	sigma6 0 users

Fig. 6. The host selection box.



Remesh or change limit ?

☐ Ok to remesh

Change limit 0.10

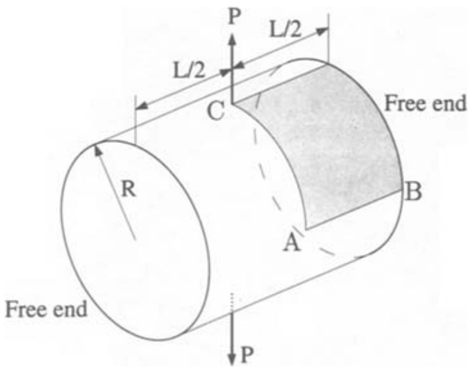
Fig. 7. The Remesh prompt box.

if he wants the FE mesh to be regenerated based on the computed error distribution, or if he wants to increase the error tolerance and continue with the same mesh, see Fig. 7. If he chooses the first alternative, the FEA is terminated at the current load step while saving all relevant data such that a restart is possible, and the mesh generator is invoked for generation of the new mesh. FENRIS is then restarted and the transfer of solution variables from the old mesh to the new mesh is performed before normal execution resumes with the new FE mesh.

During execution, the deformed FE model is displayed in the image window. Various results from the last computed increment may be fringed on the image using the **Results** menu. The results are automatically updated after the next increment is completed. If more time is needed to study the results of an increment than it takes to compute the next one, the analysis may be halted while studying the results. This is done by using the **Stop** and **Step** options in the **Command** menu. By choosing the **Stop** and **Restart** options, the user is also allowed to change the analysis parameters (e.g., analysis type, step length, convergence tolerance, error tolerance, etc.) and/or modify the problem definition by changing element type, loads, boundary conditions, material models, etc.

In the following, the use of the developed interactive–adaptive FE system is demonstrated by solving three benchmark shell problems. Two types of shell elements are employed in this study: (1) A three-noded triangular (FFT) and a four-noded quadrilateral (FFQ) Kirchhoff type of element based on the 'Free Formulation' theory [36, 37], and (2) a four-noded quadrilateral based on the Geometrically Exact Shell (GES) model due to Simo and Fox. The latter element, denoted GES42, utilizes a five-parameter Hellinger–Reissner consistent formulation for the membrane and bending fields, while the assumed natural coordinate strain concept is adopted for the transverse shear strain part. A thorough description of the underlying theory and the implementation of this element may be found in References 38–40.

The path solution strategy employed to solve the arising nonlinear systems of equations incorporates arc-length control with true Newton–Raphson



Radius	:	R	=	4.953
Length	:	L	=	10.35
Thickness	:	t	=	0.094
Elasticity	:	E	=	$1.05 \cdot 10^7$
		ν	=	0.3125
Load	:	P	=	$60.0 \cdot 10^3$

Fig. 8. The stretched cylinder problem: Geometry and properties.

equilibrium iterations. An energy norm is used as convergence criterion, with a convergence tolerance $\varepsilon_E = 0.0001$. The desired number of iterations per load step is set to $n_{\text{iter}}^{\text{d}} = 5$ and the size of the load increments, measured in terms of the incremental arc-length, Δs, are adjusted automatically by means of Eq. (6).

4.1. Stretched Cylinder with Free Ends

The geometry, the material characteristics and the loading conditions for the first example are illustrated in Fig. 8. One eighth of the depicted cylinder is analyzed using symmetry boundary conditions.

The cylinder is herein analyzed using the triangular Free Formulation element (FFT). The analysis is performed with $\eta_p = 10\%$ as the prescribed tolerance on the estimated relative error and with the initial load step size set to $\Delta\lambda_1^{(0)} = 0.01$. This resulted in the sequence of refined meshes shown in Fig. 9. The meshes 1–3 are here depicted in their deformed configuration corresponding to the load steps at which they are generated, i.e., at load step 1, 5 and 7, respectively. In addition, the final configuration (step 12) is shown for mesh 3.

In Fig. 10, the horizontal displacements at the points A and B and the vertical displacement under the point load (point C) are plotted versus the applied load for the adaptive analysis. The present solution for the displacement at point C is compared with the solution obtained by Perić and Owen [41] who used a uniform mesh with 10×20 triangular elements. It appears that the present solution is slightly stiffer than Perić and Owen's.

At the load level $\approx 0.35 \times P$ the cylinder experiences a snap-through in the area that deflects inwards (compare the deformation of mesh 2 and mesh 3 in Fig. 9 and study the curves A and B in Fig. 10). At this load level also the third refinement occurs and from Fig. 10 we see that the finer mesh captures the snap-through better than the coarser mesh does.

In Fig. 11 the estimated discretization error and the CPU-time consumption for the adaptive analysis is compared with a non-adaptive analysis using a uniform 20×40 mesh. We observe that the total time consumption for the adaptive analysis is only a small fraction of the time consumption for the single mesh analysis. Moreover, the single mesh analysis satisfies an accuracy of $\eta_p = 16\%$ only, compared with $\eta_p = 10\%$ for the adaptive analysis.

4.2. Hinged Cylindrical Shell

The geometry, material characteristics and loading conditions for the second example are illustrated in Fig. 12. The straight edges are hinged whereas the curved edges are free. Assuming a symmetric response, only the shaded quarter of the depicted shell is analyzed using symmetry boundary conditions. This classical snap-through problem is herein analyzed using the quadrilateral Free Formulation element (FFQ) and with the initial load step size set to $\Delta\lambda_1^{(0)} = 0.2$.

An interesting aspect with this example is that the discretization error appears to be highest at the first part of the FE solution path. Therefore, an adaptive analysis would normally create a relatively fine mesh at the very first load step and then stick with that mesh throughout the analysis. This mesh would, however, be unnecessarily fine for the last part of the analysis. Instead, we therefore introduce a *lower* limit on the global relative error and perform a mesh coarsening when the estimated error drops below that limit. This lower limit is here set to $\eta_l = 2\%$ whereas the permissible relative error is set to $\eta_p = 4\%$. The resulting sequence of meshes are depicted in Fig. 13, in their deformed configurations.

In Fig. 14, the vertical displacements at the points A and B are plotted versus the applied load for the adaptive analysis. The present solution is compared with the solution obtained by Nygård [37] using the same element and a uniform 8×8 mesh throughout the analysis.

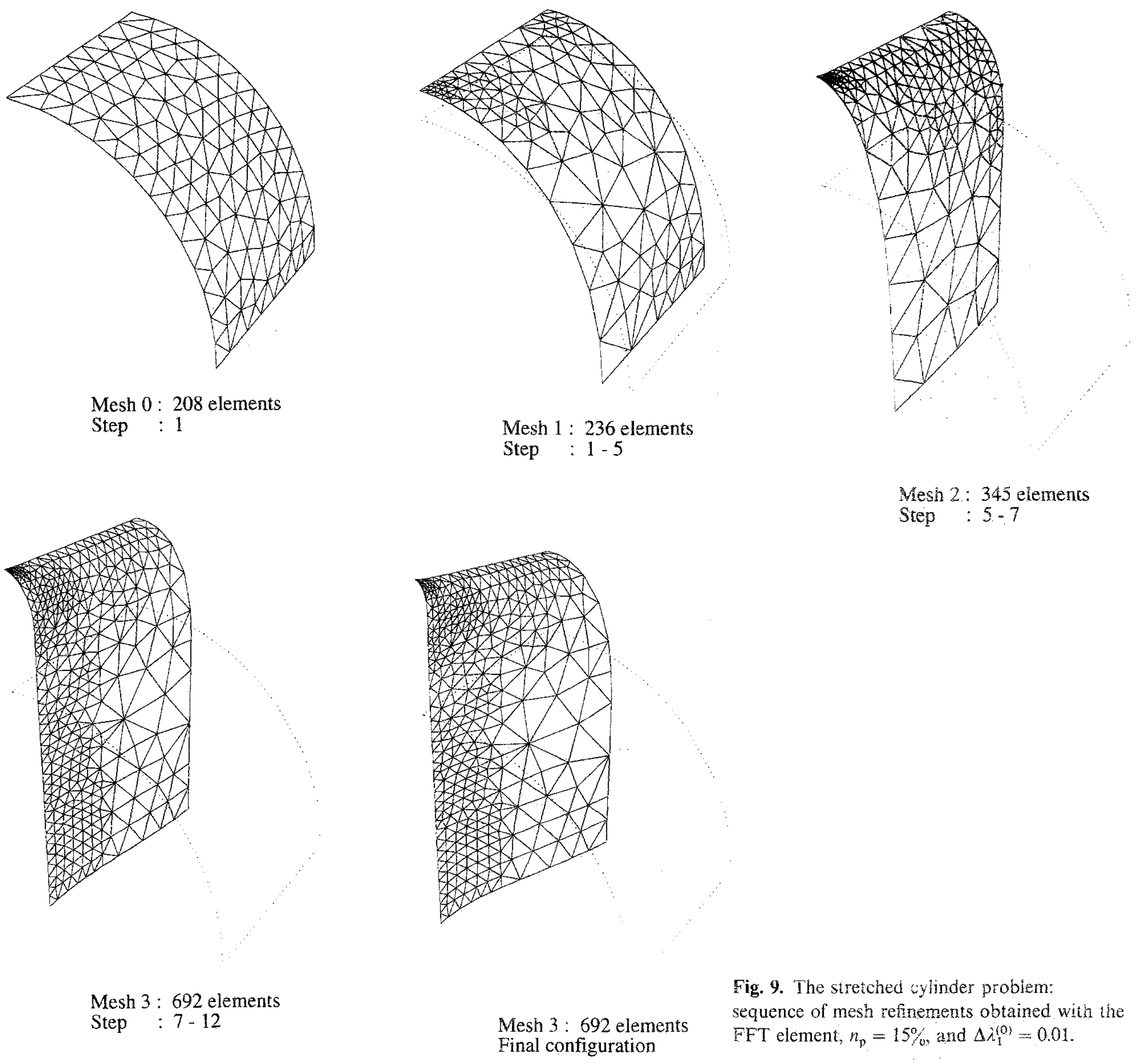


Fig. 9. The stretched cylinder problem: sequence of mesh refinements obtained with the FFT element, $n_p = 15\%$, and $\Delta\lambda_1^{(0)} = 0.01$.

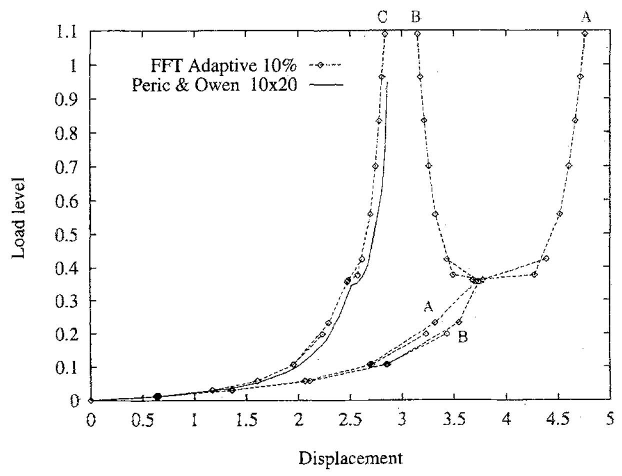
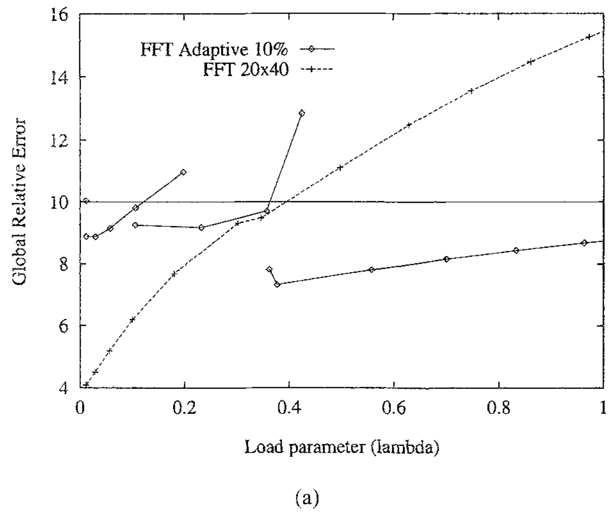


Fig. 10. The stretched cylinder problem: Load–deflection curves for the different mesh configurations at the three points A, B, and C.

In Fig. 15 the estimated discretization error and the CPU-time consumption are plotted as functions of the load step number for the two cases $\eta_l = 2\%$ (labeled 4%–2%) and $\eta_l = 0$ (labeled 4%). We observe that the total time consumption is reduced by approximately 20% when introducing the lower error limit $\eta_l = 2\%$.

4.3. Compressed Pear-Shaped Cylinder

The pear-shaped cylinder shown in Fig. 16 has been studied by many researchers, see e.g. References 42–46. The cylinder is simply supported at each end and is subjected to an uniform end-shortening. The response becomes nonlinear at low values of the applied end-shortening and the flat portions of the cylinder deflects outward as the load is increased. Due



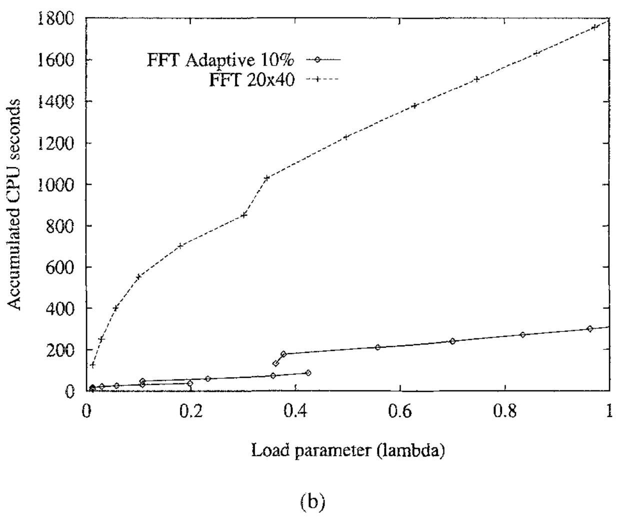
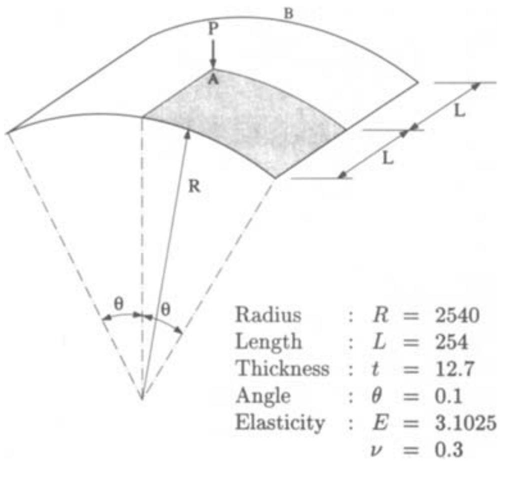


Fig. 11. The stretched cylinder problem: (a) Development of the global relative error during the analyses, and (b) accumulated CPU-time consumption.



Radius	: R	=	2540
Length	: L	=	254
Thickness	: t	=	12.7
Angle	: θ	=	0.1
Elasticity	: E	=	3.1025
	ν	=	0.3

Fig. 12. The hinged cylindrical shell problem: Geometry and properties.

to symmetry, one quarter of the cylinder is analyzed as indicated by the gray area in Fig. 16.

The cylinder is herein analyzed in the load level range $0.0 < \lambda < 0.002$, where the load parameter λ here corresponds to the prescribed end-displacement in the longitudinal direction of the cylinder. The initial size of the first load increment is specified equal to $\Delta\lambda_1^{(0)} = 0.0004$.

Four different analyses are considered in this example, using the quadrilateral GES42 shell element: An adaptive analysis with the error tolerance $\eta_p = 5\%$, and three single mesh analyses using 50×4, 100×8 and 200×16 elements, respectively. The mesh sequence obtained in the adaptive analysis is shown in Fig. 17. A total of eight mesh refinements are required in order to satisfy the rather strict error tolerance, $\eta_p = 5\%$, throughout the entire analysis.

Hartung and Ball [43] reported an elastic collapse load of 2372 for this problem using a finite difference version of STAGS [46]. Several years later, Almroth and Brogan [44] performed a convergence study using the FE version of STAGS and estimated the collapse load to be between 2300 and 2400. These results, along with the present results, results obtained with the NASA CSM Testbed [45] as well as the result of McCleary and Knight's study [42], are summarized in Table 1. The latter, which also agrees very well with the converged value obtained with the CSM Testbed, is within 0.5% of the value obtained with the present adaptive procedure. It therefore seems more likely that the collapse load is around 2450 and not between 2300 and 2400 as claimed by Almroth and Brogan.

In Fig. 18 the total axial load (i.e., the sum of the nodal reaction forces in the axial direction along the edge with the prescribed displacement) is plotted versus the normal deflection, w_{180°, for each of the present analyses. Observe that the load of collapse is highly dependent on the degree of fineness of the FE mesh employed, whereas the response prior to collapse appears to be quite insensitive to that. It is therefore important to use a relatively strict error tolerance η_p in an adaptive analysis of this problem, in order to get an accurate prediction of the collapse load.

In Fig. 19 the estimated error and the accumulated CPU-time are plotted as functions of the load level. The 100×8 analysis stays below the 5% error limit up to the load level $\lambda = 0.00155$, which is just after the collapse load has been reached. We observe that the adaptive analysis reaches this load level with only 75% of the time-consumption required by the 100×8 analysis. Both the 200×16 analysis and the adaptive analysis stay below the 5% error limit up to reaching the second turning point in the post-collapse phase.

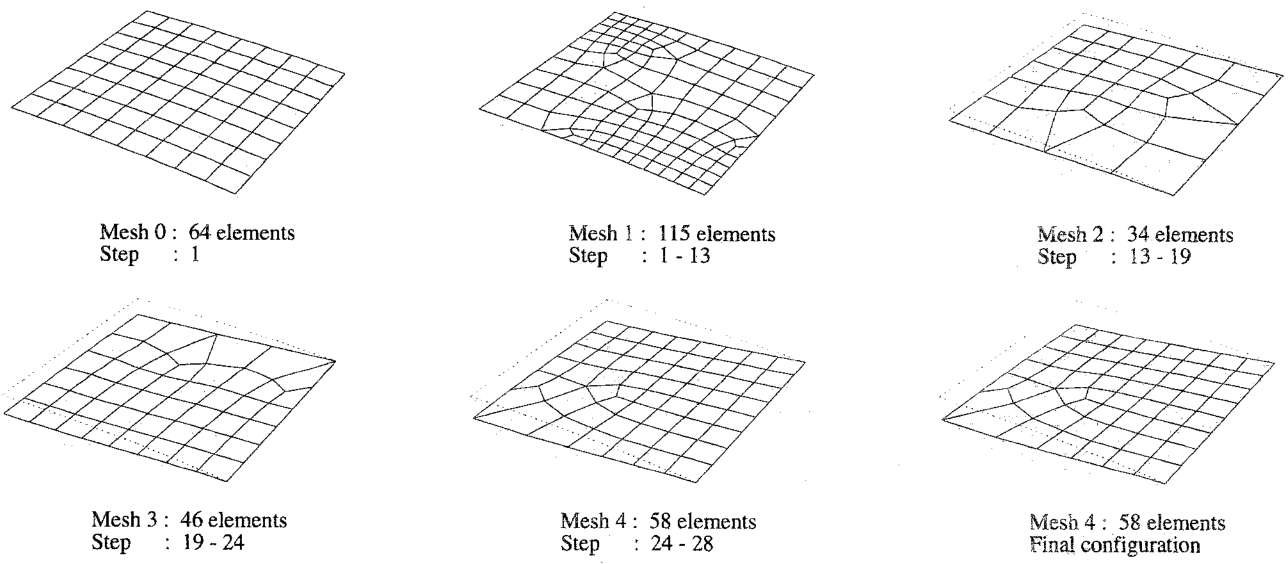


Fig. 13. The hinged cylindrical shell problem: sequence of mesh refinements obtained with the FFQ element, $n_p = 4\%$, $n_l = 2\%$ and $\Delta\lambda_1^{(0)} = 0.4$.

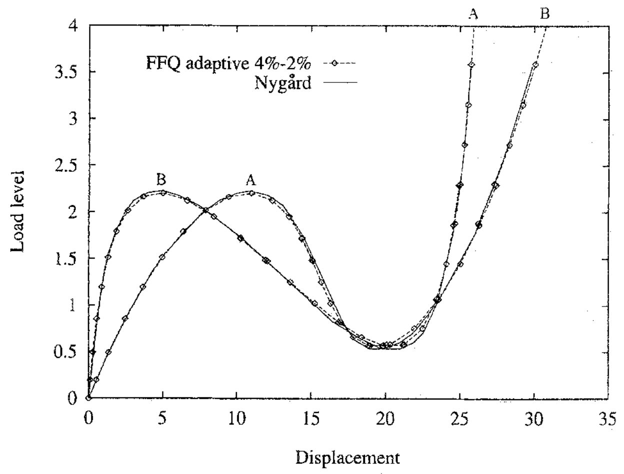


Fig. 14. The hinged cylindrical shell problem: Load–deflection curves for the different mesh configurations at the two points A and B.

However, the 200 × 16 analysis spends 2.8 times more CPU time than the adaptive analysis to reach this point.

5. Conclusions

The theoretical developments and practical applications presented here confirm that efforts toward automatic adaptive NFEA in CSM are definitely feasible and promise great payoff in practical as well as research communities. Once implemented and available, the interactive–adaptive automated version will greatly improve quality, reliability and time efficiency by providing:

(1) An efficient way to monitor the results of analysis in real computing time. This capability can also be used to allow the user to control the flow of the analysis, intervene when problems appear,

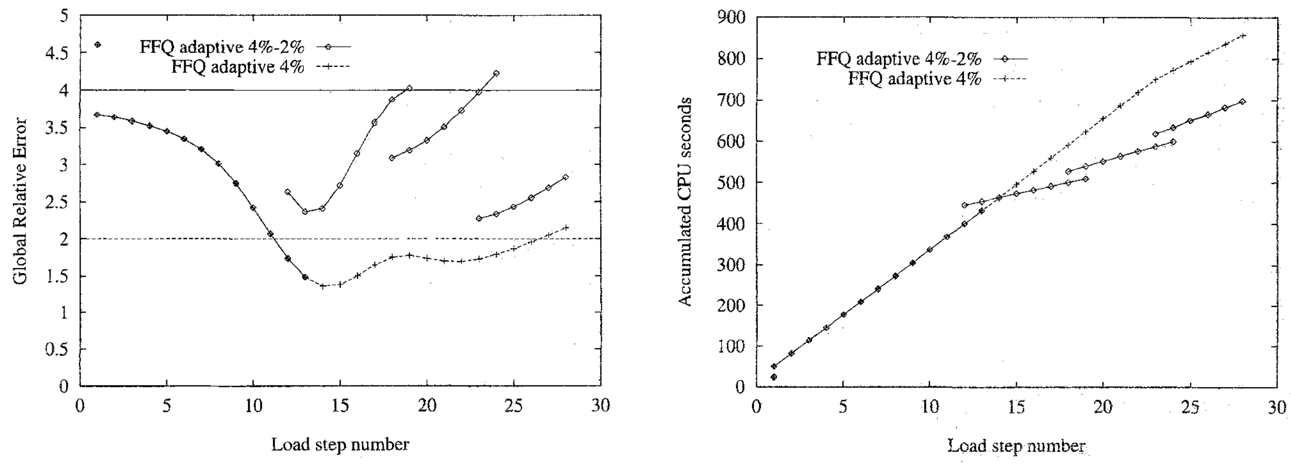
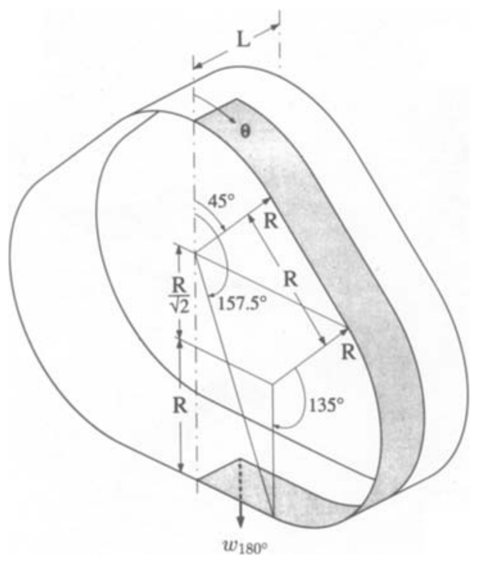


Fig. 15. The hinged cylindrical shell problem: (a) Development of the global relative error during the analyses, and (b) accumulated CPU-time consumption.

Table 1. The pear-shaped cylinder problem: Elastic collapse loads.

Source	*Element*	*Mesh size*	*Collapse load*
Present study	GES42	50×4 Elements	2685
		100×8 Elements	2515
		200×16 Elements	2475
		Adaptive 5%	2460
McCleary and Knight	9-node ANS C^0-Element	Adaptive 10%	2471
CSM Testbed	4-node ANS C^0-Element	26×3 Nodes	3945
		37×5 Nodes	2696
		51×7 Nodes	2568
	9-node ANS C^0-Element	37×5 Nodes	2475
		51×7 Nodes	2466
Almroth and Brogan	4-node flat $C1$-Element	25×3 Nodes	3586
		37×5 Nodes	2731
		47×7 Nodes	2586
	4-node flat C^1-el. w/drilling DOF's	37×5 Nodes	2657
		47×7 Nodes	2530
Hartung and Ball		40×4 Nodes	2372



Radius	:	R	=	1.0
Length	:	L	=	0.8
Thickness	:	t	=	0.01
Elasticity	:	E	=	10^7
		ν	=	0.3

Fig. 16. The pear-shaped cylinder problem: Geometry and properties.

redefine parameters and loads through the history concept, and modify algorithms.

(2) A more simplified nonlinear analysis that is easy to comprehend. Moreover, interactive–adaptive techniques also enhance finite element research by making the overall numerical experimental process more creative.

(3) Automatic control of the quality of the computational results, in particular keeping the discretization error within prescribed limits.

(4) A minimum computational effort necessary to obtain good quality results by automated design of the optimal mesh, optimal step length (time step in dynamics) and by carrying the analysis out on the most appropriate hardware.

Although the interactive–adaptive integrated system developed here was proven to be very useful and effective, the development of a fully automated CAE environment will require extensive further developments. This includes the following, and will be dealt with in our future work:

(1) Knowledge-based expert systems that provides assistance for inexperienced engineers.

(2) Automatic handling of all kinds of computational difficulties, by automatically switching to the most appropriate solution algorithm at any time during the analysis.

(3) Automatic learning capabilities, with the system's own experience growing with the number of solved problems.

(4) Further development and improvement of adaptive computational techniques and error estimates.

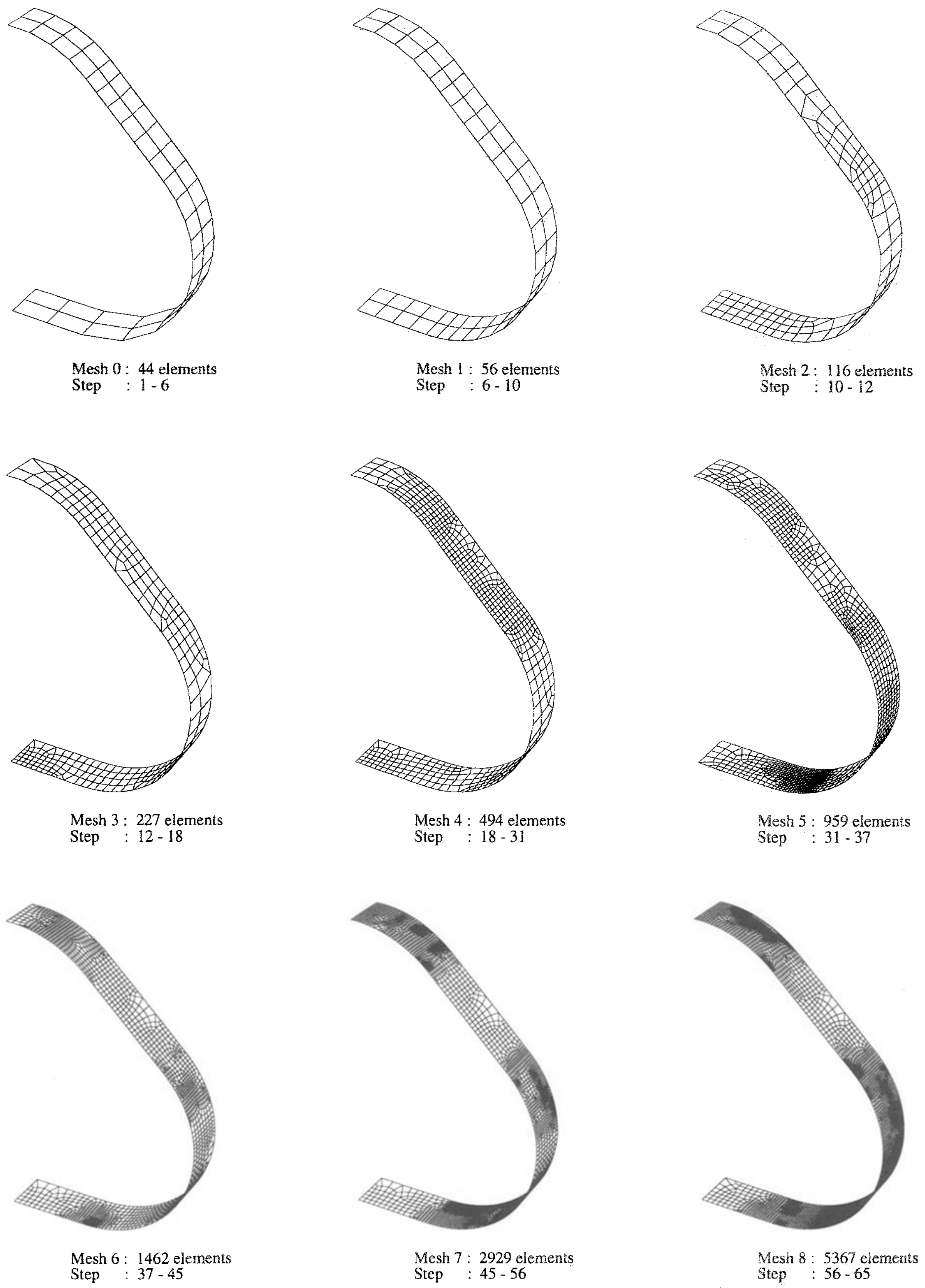
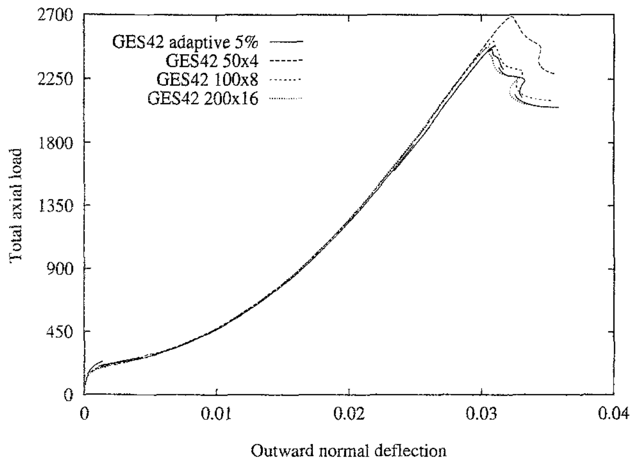
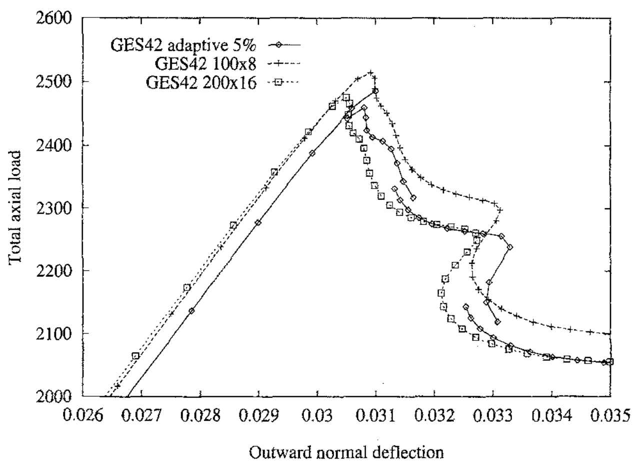


Fig. 17. The pear-shaped cylinder problem: Sequence of mesh refinements obtained in the adaptive nonlinear analysis using the GES42 element, $\eta_p = 5\%$ and $\Delta\lambda_1^{(0)} = 0.0004$.

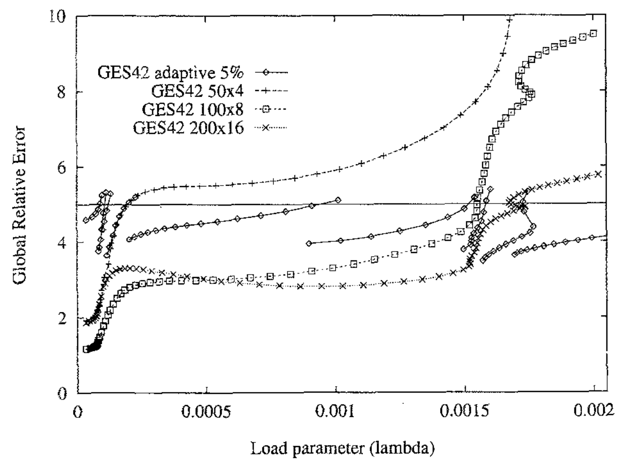


(a)

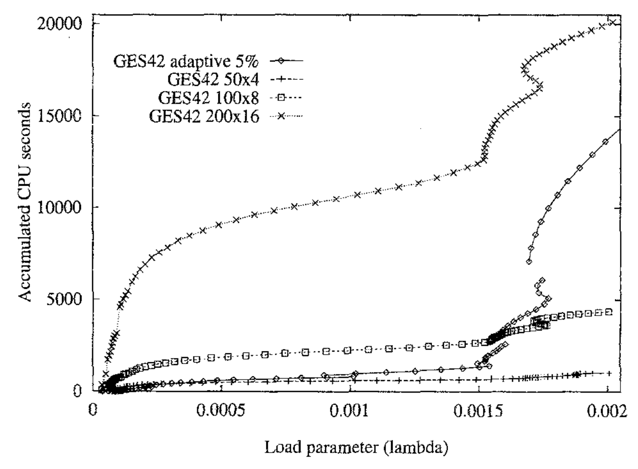


(b)

Fig. 18. The pear-shaped cylinder problem: (a) Load–deflection plots for the different mesh configurations and (b) detailed look at the top of the load–deflection curves.



(a)



(b)

Fig. 19. The pear-shaped cylinder problem: (a) Development of the global relative error during the analyses, and (b) accumulated CPU-time consumption.

Acknowledgments

The authors would like to thank their colleagues in the FENRIS project for developing parts of the computer program.. Siv.Ing. Asbjørn Mangerud and Øyvind A. Syljuåsen is especially acknowledged for developing parts of the FENRIX program. The authors are also very much obliged to Dr. Ing. Arne Chr. Damhaug for the implementation of the Sparse Matrix Package in FENRIS, and to Professor Mark S. Shephard for providing the Finite Octree mesh generator. They also acknowledge the support, financial or otherwise from NTH and from Tungregneutvalget, NRF.

References

1. Fredriksson, B.; Mackerle, J. (1984) Structural Mechanics: Finite Element Computer Programs, 4th edn, Linkøping, Sweden, Advanced Engineering Corp.
2. Noor, A. K.; Babuška. I. (1987) Quality assessment and control of finite element solutions, Finite Elements in Analysis and Design, 3, 1–26.
3. PDA Engineering, Costa Mesa, California (1988) PATRAN Plus User Manual.
4. Structural Dynamics Research Corporation, Milford, Ohio (1992) I-DEAS (Integrated Design Engineering Analysis Software) User Manuals.
5. Det Norske Veritas Sesam A.S., Høvik, Norway (1993) SESAM User's Manual, SESAM System.
6. Haugen, B.; Mathisen, K.M. (1989) Large-scale vectorized three-dimensional geometric nonlinear shell analysis on a CRAY X-MP/28 utilizing sparse matrix methods. In Proceeding of the First International Conference on Applications of Supercomputers in Engineering (ASE89), volume FFSAA, pp. 189–205, Southampton, England
7. Farhat, C.; Crivelli, L. (1989) A general approach to nonlinear FE computations on shared-memory multiprocessors, Comp. Meths. Appl. Mech. Engrg. 72, 153–171.

8. Damhaug, A.C. (1992) Sparse solution of finite element equations, Dr. Ing. Dissertation, Department of Structural Engineering, The Norwegian Institute of Technology, Trondheim, Norway.
9. Damhaug, A.C.; Mathisen, K.M.; Okstad, K.M. (1994) The use of sparse matrix methods in finite element codes for structural mechanics applications, Comp. Syst. Engrg., 4, 355–362
10. Bathe, K.-J., Dvorkin, E.N. (1983) On the automatic solution of nonlinear finite element equations, Comp. Struct., 17, 871–879.
11. Lee, S.H.; Hsieh, S.S. (1991) Applications of a self-adaptive algorithm to non-linear finite element analysis, Int. J. Numer. Meths. Engrg, 32, 1057–1077.
12. Shephard, M.S. (1985) Finite element modeling with an integrated geometric modeling environment: Part II Attribute specification, domain differences and indirect element types, Engrg. Comp., 1, 73–85.
13. Weiler, K.J. (1986) Topological structures for geometric modelling. PhD Thesis, Center for Interactive Computer Graphics, Rensselaer Polytechnic Institute, Troy, New York.
14. Delmas, J.E.; Tembulkar, J.M. (1990) Geometry based automated finite element modeling and analysis. In FEM in the Design Process, Proceedings of the 6th World Congress on Finite Element Methods, Banff, Alberta, Canada, Robinson J. (editor), pp. 280–286, Great Bidlake Manor, UK, Robinson and Associates.
15. Shephard, M.S. (1988) Approaches to the automatic generation and control of finite elements meshes, Appl. Mech. Rev., 41, 169–185.
16. Peraire, J.; Vahdati, M.; Morgan, K.; Zienkiewicz, O.C. (1987) Adaptive meshing for compressive flow computations, J. Comp. Phys., 72, 449–466.
17. Shephard, M.S.; Georges, M.K. (1991) Automatic three-dimensional mesh generation by the finite octree technique, Int. J. Numer. Meths. Engrg., 32, 709–749.
18. Leal, D. (1991) Progress towards a standard for finite element data. In Proceedings of the 3rd International Conference on Quality Assurance and Standards in Finite Element Analysis, Stratford-upon-Avon, UK, September, NAFEMS.
19. Hunten, K.A. (Editor) (1991) FEA Reference Model, ISO TC184/SC4/WG3 Document N21.
20. Riks, E. (1972) The application of Newton's method to the problem of elastic stability, J. Appl. Mech., 39, 1060–1066.
21. Wempner, G.A. (1971) Discrete approximations related to nonlinear theories in solids, Int. J. Solids Struct., 7, 1581–1599.
22. Crisfield, M.A. (1981) A fast incremental/iterative solution procedure that handles snap-through, Comp. Struct., 13, 55–62.
23. Schweizerhof, K.H.; Wriggers, P. (1986) Consistent linearization for path following methods in nonlinear FE analysis, Comp. Meths. Appl. Meth. Engrg., 59, 261–279.
24. Fried, I. (1984) Orthogonal trajectory accession to the nonlinear equilibrium curve, Comp. Meths. Appl. Mech. Engrg., 47, 283–297.
25. Zienkiewicz, O.C.; Zhu, J.Z. (1987) A simple error estimator and adaptive procedures for practical engineering analysis, Int. J. Numer. Meths. Engrg., 24, 337–357.
26. Hinton, E.; Campbell, J.S. (1974) Local and global smoothing of discontinuous finite element functions using a least squares method, Int. J. Numer. Meths. Engrg., 8, 461–480.
27. Zienkiewicz, O.C.; Zhu, J.Z. (1992) The superconvergent patch recovery and *a posteriori* error estimates. Part 1: The recovery technique, Int. J. Numer. Meths. Engrg., 33, 1331–1364.
28. Blacker, T.; Belytschko, T. (1994) Superconvergent patch recovery with equilibrium and conjoint interpolant enhancements, Int. J. Numer. Meths. Engrg., 37, 517–536.
29. Zhu, J.Z.; Zienkiewicz, O.C. (1988) Adaptive techniques in the finite element method, Commun. Appl. Numer. Meths., 4, 197–204.
30. Okstad, K.M. (1994) Adaptive methods for nonlinear finite element analysis of shell structures, Dr. Ing. Dissertation, Department of Structural Engineering, The Norwegian Institute of Technology, Trondheim, Norway
31. Okstad, K.M.; Mathisen, K.M. (1994) Towards automatic adaptive geometrically nonlinear shell analysis. Part I: Implementation of an *h*-adaptive mesh refinement procedure, Int. J. Numer. Methods. Engrg., 37, 2657–2678.
32. Fyrileiv, O. (1991) FENRIS—Finite Element NonlineaR Integrated System, user's manual, Veritas Sesam Systems A.S., Høvik, Norway.
33. Aamnes, K. (1993) GLview Command Summary, SINTEF Industrial Mathematics, Trondheim, Norway.
34. Mangerud, A.; Mathisen, K.M.; Okstad, K.M.; Syljuåsen, Ø.A. (1993) FENRIX—Finite Element NonlineaR Integrated X-environment, Users Manual, Technical Report R-8-93, Department of Structural Engineering, The Norwegian Institute of Technology, Trondheim, Norway.
35. Arnholm, C. (1989) SIF Results Interface File, File Description, Veritas Sesam Systems A.S., Høvik, Norway.
36. Bergan, P.G.; Nygård (1985) Nonlinear shell analysis using free formulation finite elements. In Proceedings of the Europe–US Symposium on Finite Element Methods for Nonlinear Problems, Trondheim, Norway, Bergan, P.G.; Bathe, K.-J.; Wunderlich, W. (Editors) New York and Tokyo, Springer-Verlag.
37. Nygård, M.K. (1986) The free formulation for nonlinear finite elements with applications to shells, Dr. Ing. Dissertation, Division of Structural Mechanics, The Norwegian Institute of Technology, Trondheim, Norway.
38. Simo, J.C.; Fox, D.D. (1989) On a stress resultant geometrically exact shell model. Part I: Formulation and optimal parametrization, Comp. Meths. Appl. Mech. Engrg., 72, 267–304.
39. Simo, J.C.; Fox, D.D.; Rifai, M.S. (1989) On a stress resultant geometrically exact shell model. Part II: The linear theory: computational aspects, Comp. Meths. Appl. Mech. Engrg, 73, 53–92.
40. Simo, J.C.; Fox, D.D.; Rifai, M.S. (1990) On a stress resultant geometrically exact shell model. Part III: Computational aspects of the nonlinear theory, Comp. Meths. Appl. Mech. Engrg., 79, 21–70.
41. Perić, D.J.; Owen D.R.J. (1991) The Morely thin shell finite element for large deformations problems: Simplicity versus sophistication. In Proceedings of the 4th International Conference on Nonlinear Engineering Computations (NEC-91), Split, Yugoslavia, Biaćanić, N.; Marović, P.; Owen, D.R.J.; Mihanović, A. (Editors).
42. McCleary, S.L.; Knight, N.F. Jr. (1990) Error detection and control for nonlinear shell analysis. In FEM in the Design Process, Proceedings of the 6th World Congress on Finite Element Methods, Banff, Alberta, Canada, Robinson, J. (Editor) pp. 192–201, Great Bidlake Manor, UK, Robinson and Associates.
43. Hartung, R.F.; Ball, R.E. (1973) A comparison of several computer solutions to three structural shell analysis problems, Technical Report AFFDL-TR-73-15, U.S. Air Force.
44. Almroth, B.O.; Brogan, F.A. (1981) Computational efficiency of shell elements. In Nonlinear Finite Element Analysis of Plates and Shells, Hughes, T.J.R.; Pifko, A.; Jay, A. (Editors), volume 48, pp. 147–165, ASME.

45. Knight, N.F. Jr.; McCleary, S.L.; Macy, S.C.; Aminpour, M.A. (1989) Large-scale structural analysis: The structural analyst, the CSM testbed and the NAS system, Technical Memorandum 100643, NASA Langley Research Center, Hampton, Virginia.

46. Almroth, B.O.; Brogan, F.A.; Stanley, G.M. (1979) Structural Analysis of General Shells, Vol. II: User Instructions for the STAGS(C-1) Computer Code, Lockheed Report LMSC-D33873.