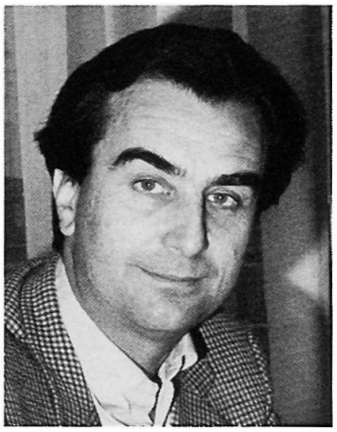
Nonlinear Structural Analysis: the Glass-Box Approach

Analyse structurale nonlinéaire interactive

Interaktive nichtlineare Tragwerkanalyse

E. ANDERHEGGEN
Prof. Dr.
Swiss Inst. of Technol.
Zurich, Switzerland



Prof. Dr. E. Anderheggen, born 1939, Diplom and Ph.D. at the ETH in Civil Engineering. Since 1976 Professor for Applied Computer Sciences at the ETHZ.

SUMMARY

A new approach to incremental nonlinear static and dynamic analysis as implemented in a recently developed general purpose finite element program is presented. The approach is new not because new algorithms or element models are implemented, but because full interaction between the program and its user during the step-by-step analysis process is possible. This means total involvement of the program user with the algorithms and the element models he is using, i.e. the opposite of the "black box" approach generally adopted.

RESUME

Une nouvelle façon de traiter les problèmes d'analyse structurale nonlinéaire est présentée. La nouveauté consiste non pas dans les modèles ou les algorithmes utilisés mais dans le fait que l'utilisateur a la possibilité de suivre en temps réel le développement du calcul sur l'écran de son terminal et d'intervenir à chaque instant pour changer les paramètres et les stratégies utilisés pour la solution du problème.

ZUSAMMENFASSUNG

Eine neue Arbeitsweise bei der Lösung nichtlinearer Probleme der Tragwerksanalyse wird vorgestellt. Neu sind dabei nicht etwa die verwendeten Modelle oder Algorithmen, sondern die dem Programmbenützer gebotenen Möglichkeiten, den Verlauf der Berechnungen auf seinen Bildschirmterminal zu verfolgen und die Lösungsparameter jederzeit interaktiv zu verändern.

1. INTRODUCTION

Many difficulties encountered when applying sophisticated numerical procedures to real-life structural design problems stem from the fact that the engineer responsible for the design and the computer scientist who developed the FE program used for the analysis can not be the same person. While in the (good) old days all needed know-how was concentrated in one person, today at least two highly specialized but unfortunately totally unrelated professionals are needed. Sometimes a third person comes in to play. This happens when the structural designer seeks the help of a "stress analyst" for modeling his structural problem and running the FE program. In fact structural designers, in many cases, do not have the time and the know-how needed for using a FE program or even for checking that somebody else has used it correctly.

This situation is particularly unfavorable in nonlinear analysis which can only make sense if the approximations involved in the numerical model are understood, if the workings of the iterative algorithms used and their sensitivity to the many parameters involved are known and if the basic design problems the analysis is supposed to solve are clearly formulated. When some of these conditions are not satisfied design and analysis tend to diverge from each other : the designer bases his decisions, just like in the old days, only on common sense, experience and simple hand calculations while the "sophisticated" numerical results obtained by the stress analyst are often accepted only if they confirm the expectations of the designer. Sometimes they are considered a waste of time just needed to make contractors and regulators happy.

What can the computer scientist do to enhance the confidence of the designer in FE analysis ? An obvious answer is that adequate teaching and "user fiendliness" of the computer programs are of great importance. In addition, at least for non-linear analysis, it is certainly necessary for the program user to truly understand the algorithm and the element models he is using.

Based on these considerations a new general purpose FE-program called FLOWERS capable of linear and nonlinear static and dynamic analysis for different types of structures was developed. The most remarkable feature of this program is the approach adopted in nonlinear static and dynamic analysis: the program allows full interaction between the computer and the program user during the execution of the interactive step-by-step procedures needed in nonlinear analysis. This means that the program user can sit in front of a terminal and follow on the screen the development of the numerical computations in real-time. At any time he can stop execution, look at different partial results, change solution strategies, go back to some predefined restart point, repeat some parts of the analysis with different solution parameters, compare results, etc. It is the aim of this paper to discuss this approach as implemented in the program FLOWERS.

2. NONLINEAR ANALYSIS WITH THE PROGRAM "FLOWERS" : GENERALITIES

The program FLOWERS consists of eight independent programs, called modules, which communicate with each other through secondary storage. For nonlinear analysis the following four modules are used:

SYSIN (SYStem INput module) for inputting system, element and load data for both linear and nonlinear static and dynamic problems

NOMBA (NONlinear BAtch module) for inputting additional specification needed for nonlinear static and dynamic analysis

NONIN (NONlinear INteractive module) for the step-by-step solution of nonlinear static and dynamic problems

GRAPH for graphical output.

The FLOWERS modules are made up of two different kinds of interconnected FORTRAN routines: the module routines and the element routines. The module routines build the main body of each module. They perform the processing steps needed for any kind of finite element model independently of specific applications. The modules NONBA and GRAPH are application independent being made up exclusively of module routines. The element routines are related to specific library element models used for implementing a variety of finite element procedures for different applications. The main task is to compute the numerical coefficients describing the physical behaviour of each element (i.e. the local element matrices) to be transmitted to the module routines for further processing. Library element routines are attached to the modules SYSIN and NONIN. Therefore these modules are application dependent. In fact for different classes of applications (like plane trusses, space frames and shells, three dimensional continua, etc.) separate versions of these modules comprising different element routines identified by their library names exist.

All FLOWERS modules with the exception of NONIN are batch programs in the sense that they communicate with their users only through input and output textfiles. The syntax and, to a large extent, the semantics, i.e. the physical meaning of the free-format, problem-oriented input language used for specifying the input data (i.e. for writing the input file) are defined and described by syntax diagrams similar to those often used for defining programming languages (e.g. PASCAL). Fig. 1 shows the syntax diagrams of the module NONBA. The very simple conventions needed to understand how to use these diagrams shall not be explained here (see (1)). It should be stressed however, that we believe, and practical experience with FLOWERS and other programs confirm, that syntax-diagrams are an extremely effective way, in fact probably the only acceptable way if real user friendliness is sought, for defining and describing batch oriented input specifications.

Contrary to all other FLOWERS modules, the module NONIN is an interactive program expecting all needed input data to be typed by a user sitting in front of a terminal.

During the execution of the module NONIN the nonlinear computations occur. That is, for a given time history of the external loads, the joint displacements and all related state variables (e.g. the element stress components) are evaluated time-step-by-time-step. In static analysis (where time represents a load parameter) the modified Newton-Raphson algorithm ranging from the initial stress method to the original, non-modified Newton-Raphson method is implemented. For nonlinear (or linear) dynamic analysis the explicit central difference method as well as two implicit algorithms, the Newmark method and the so-called ϑ - method developed within our group (see (2)) are at the present time implemented.

These algorithms are implemented at the global system level, i.e. by means of the application independent module routines. These are interfaced to the element routines which must provide all numerical coefficients describing the nonlinear

element behaviour as needed by the particular algorithm being used. Therefore all kinds of nonlinear effects (including e.g. contact problems) are treated at the local element level by the element routines. These, as explained below ("Single value output state variables", "Direct element output"), also have to provide different kinds of output data (e.g. stress components, plastic work, etc.) needed to inform the program user about the current state of the elements.

3. PREPARING FOR NONLINEAR ANALYSIS

The step-by-step solution of a nonlinear static or dynamic problem for a given time history of the external loads requires the execution of the preparatory batch modules SYSIN (SYStem INput) and NONBA (NONlinear BAtch), followed by the interactive module NONIN (NONlinear INteractive).

The module SYSIN is used both in linear and nonlinear analysis for specifying global system data (joint coordinates, element incidences, etc.), element data (e.g. material properties) and time independent load data. After the execution of the module SYSIN (and perhaps of the module GRAPH for plotting the element mesh) some additional data specifically needed for nonlinear analysis have to be inputted by executing the batch module NONBA. Its input statements are described by the syntax diagram of Fig. 1. The type of input data they refer to can be summarized as follows:

1) ANALYSIS TYPE (static and/or dynamic) and algorithm(s) to be used. It should be noted that static analysis can precede dynamic analysis and that explicit and implicit dynamic integration algorithms can be mixed in the time domain (however not within the element mesh).

2) TIME HISTORY of the external loads, i.e. the time functions associated with each of the independent loadcases previously defined.

3) INITIAL DYNAMIC CONDITIONS refering to nodal discplacements, velocities and accelerations at the beginning of dynamic analysis.

4) DAMPING coefficients building the global damping matrix used for taking linear viscous damping into account. Nonlinear and nonviscous damping is to be treated at the element level.

5) CONVERGENCE TOLERANCE DEFINITION: with the exception of the explicit central difference method, all algorithms mentioned above have to check for convergence after each equilibrium iteration within each time-step. The vector norms used to do this have to be defined and identified by names ('ident') chosen by the program user. The actual numerical tolerance values associated with these norms are specified (resp. modified, activated or disactivated) interactively during the execution of the module NONIN.

6) SINGLE VALUE OUTPUT STATE VARIABLES associated either with nodal information (displacements, unbalanced loads, etc.) or with element information provided by the element routines (stress components, plastic work, etc.) have to be specified and identified by names ('ident') chosen by the program user. These variables can then be displayed on terminal, printed or saved for graphical postprocessing by the module GRAPH.

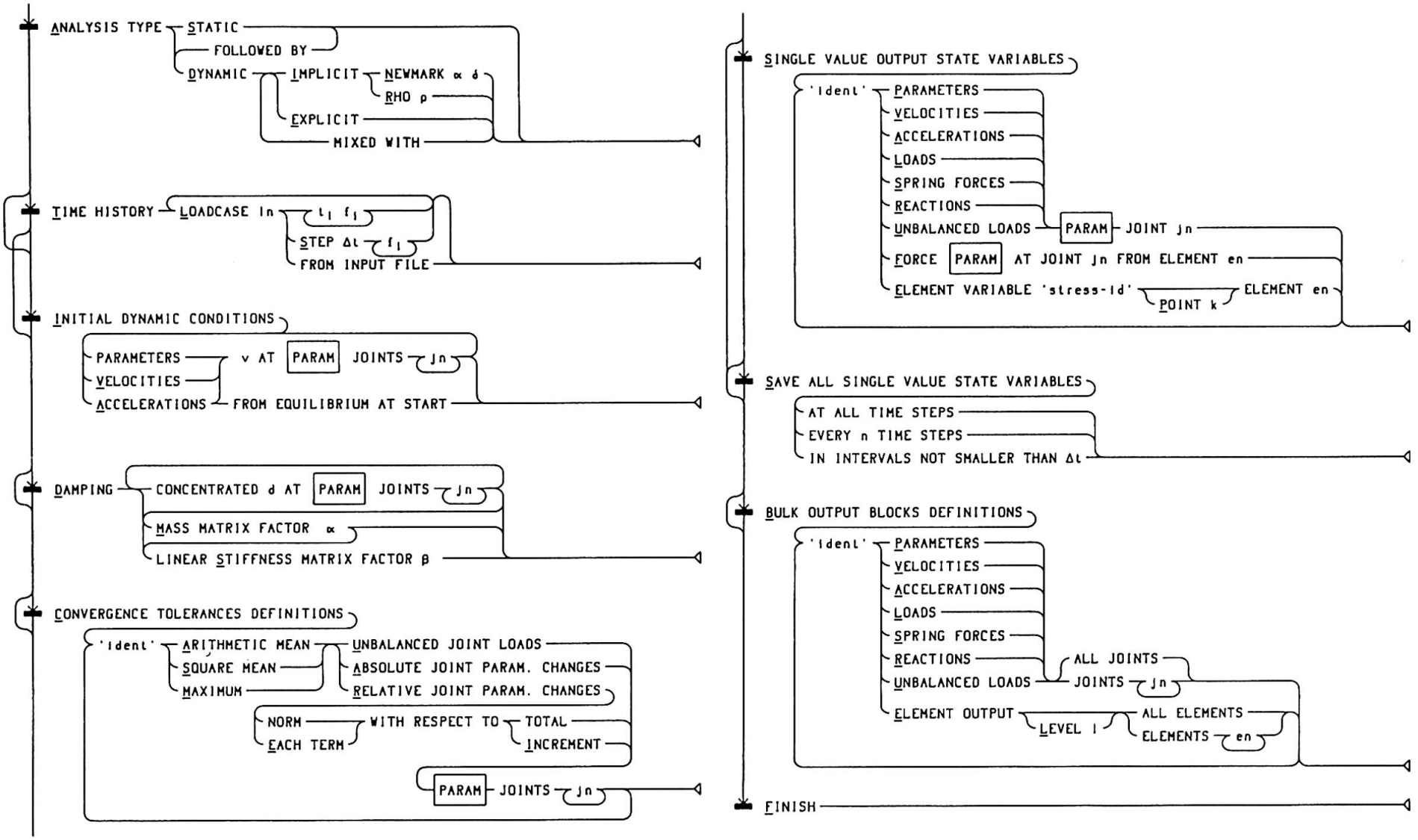


Figure 3 Syntax diagram of the module NONBA

7) BULK OUTPUT BLOCKS DEFINITIONS refering to nodal or element data to be printed when requested by the program user during the execution of the module NONIN. Again these data blocks have to be identified by names ('ident') chosen by the program user.

4. RUNNING NONLINEAR ANALYSIS

Fig. 2 shows the workings of the interactive module NONIN. Whenever requested by the program user, as well as at the beginning and shortly before the end of the interactive session, the main input menu, called the MONITOR menu, appears on the terminal screen. The program user has then the options of pressing on his

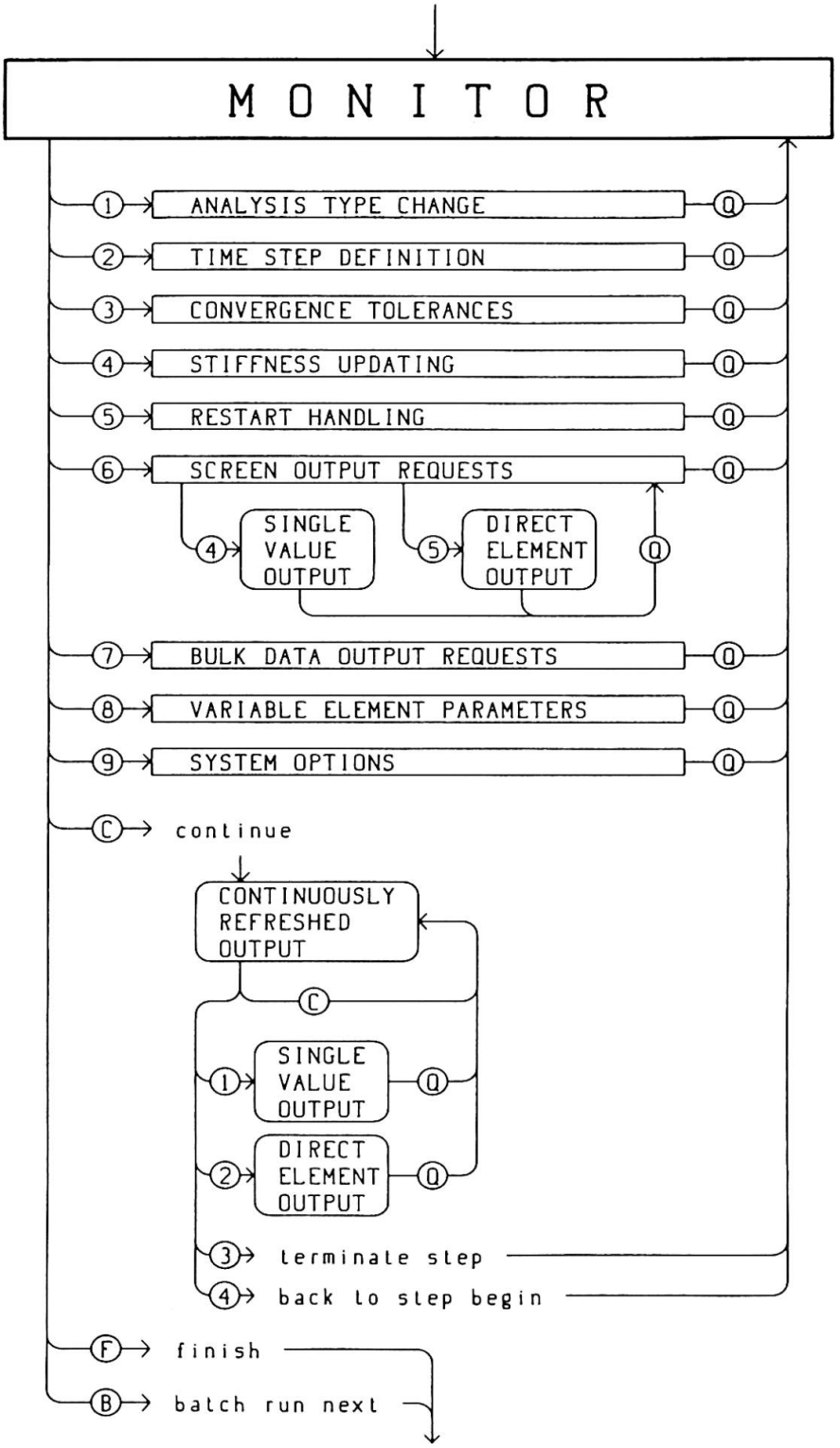


Figure 5 Interactive I/O of the module NONIN

keyboard either one of the digits "1" to "9" for calling the corresponding input sub-menu or one of the letters "C", "B" or "F" for continuing or terminating execution (these digits and letters appear on the left side of Fig. 2 where they are enclosed in circles).

The input data the user can specify by means of the sub-menus 1 to 7 can be summarized as follows (for brevity the sub-menus 8 and 9 shall not be considered here):

1) ANALYSIS TYPE CHANGE menu used for switching from static to dynamic analysis or for changing integration algorithms during dynamic analysis (e.g. from implicit to explicit or vice-versa).

2) TIME STEP DEFINITION menu used for specifying the length and the number of load history time steps until the next adjustment in time step length is needed or until the last load history step is reached. An elaborate procedure for automatically reducing the step length when convergence difficulties occur has also been introduced.

3) CONVERGENCE TOLEARANCES menu used for specifying numerical tolerance values to be compared with the tolerance vector norms defined by the module NONBA. Convergence checks can also be selectively activated or deactivated.

4) STIFFNESS UPDATING menu used for specifying the strategy to be adopted for rebuilding the global stiffness matrix (e.g. every 3 time steps at the beginning of the step, or after 7, 13 and 25 unsuccessful equilibrium iterations within any step, etc.). These options are relevant both in static analysis by the modified Newton-Raphson algorithm and in dynamic analysis when implicit algorithms are used.

5) RESTART HANDLING menu used both for defining so-called "restart points" where all relevant problem data are saved on secondary storage, and for repeating one portion of the analysis starting from a previously defined restart point. This is an essential feature in nonlinear analysis as it allows to compare results obtained by different procedures (e.g. different time steps, different stiffness updating strategies, different dynamic integration algorithms) and also to overcome difficulties due to an unfavorable choice of solution strategies (e.g. a singular tangent stiffness matrix).

6) SCREEN OUTPUT REQUEST menu used for specifying which of the single value output state variables are to be dsplayed continuously on the terminal screen. This menu is also used for displaying the current values of all these variables (SINGLE VALUE OUTPUT) or for requesting the element routines of some elements to display their so called "DIRECT ELEMENT OUTPUT" containing appropriate information on the current state of the element.

7) BULK DATA OUTPUT REQUESTS menu used for specifying the data blocks to be printed or saved for graphical postprocessing and the load history times when this has to occur.

When from the MONITOR menu the program user types the letter "C" the nonlinear analysis process is continued (or started). On the terminal screen different data are then displayed and updated at the end of each equilibrium iteration within each time step. These data include the values of the active convergance tolerance vector norms expressed as a percentage of the corresponding tolerance values. The single value output state variables selected in the sub-menu no. 6

(up to 12 of them) are also continously displayed as well as error and non-error messages sent by the element routines. All this information is displayed without interrupting execution and thus at a speed which depends greatly on the problem size and on the performance of the computer used (the program FLOWERS is being developed on a DEC-10 computer). According to different system options specified in the sub-menu no. 9, execution can be stopped at the end of each equilibrium iteration, at the end of each time step when convergence is reached, when an element sends a message, or only when all time steps defined in the sub-menu no. 2 have been analysed. However the program user also has the option to stop execution simply by pressing any key on his keyboard.

When execution is stopped different options are open. It is possible just to continue execution, to have all single value state variables displayed (SINGLE VALUE OUTPUT), to require some elements to display their "DIRECT ELEMENT OUTPUT" or to go back to the MONITOR menu. In this last case, if convergence was not reached, the program user has the option of either terminating the present step and proceding to the next step just as if convergence was reached, or to go back to the step beginning, change solution parameters and repeat the same step.

Two different ways exist for terminating the interactive session. By typing the letter "F" the interactive session is terminated. At a later time a new interactive session beginning from any of the defined restart points is then possible. By typing the letter "B" the interactive session or rather a batch job, without any input or output from the terminal, i.e. with only printer and perhaps post-processor output. When this option is chosen the program user has to specify different data (e.g. a sufficient number of time steps) so that during the following batch job no input data will be needed. After terminating an interactive session in this way, the subsequent batch job does not require the physical presence of the program user in front of the terminal. Upon termination of this batch run a new restart point is automatically set. The next run will then have to be an interactive session again.

5. CONCLUSION

A new approach to nonlinear structural analysis called the "glass box" approach was described. Its principal goal is to get the program user involved with the solution algorithms and the element models he uses by allowing him to follow and to steer the internal workings of the program on a terminal screen. In nonlinear analysis this seems to make sense. In fact its success depends on many parameters which in most cases can only be determined by trial and error, and trial-and-error procedures are much better implemented in an interactive environment than with a batch program. Of course the program user must know exactly what he is doing, but this is certainly necessary whatever program he uses. The black-box approach can not work in nonlinear analysis. Finally it is obvious, that for real-life problems, if the response times during interaction are to remain acceptable, a large amount of computer power will be needed. This however, should become less and less of a problem in the future.

REFERENCES

(1) E. Anderheggen, G. Bazzi, H. Elmer, T. Friedrich, H. Maag, J. Theiler: "FLOWERS User's Manual", 2nd edition, April 1983, Institut für Informatik, ETH-Zürich.

(2) G. Bazzi, E. Anderheggen, "The ϑ-Family of Algorithms for Time-Step Integration with Improved Numerical Dissipation, J. of Earthquake Engineering and Structural Dynamics, Vol. 10, 537-550, 1982.