THE DEVELOPMENT OF A MICROCOMPUTER PROGRAM FOR THE THREE-DIMENSIONAL NON-LINEAR FEM [1]

Ma. Mingjun Liao. Guohua Fang Zulie
University of Science Technology of Beijing, Beijing 100083, China

ABSTRACT

This paper presents a microcomputer program system for the three-dimensional non-linear finite element method (FEM), which is developed for applications in mining engineering, such as excavation, filling, caving, damming, etc. Our attention is focused on some of the key aspects of the programing techniques for an applicable microcomputer program, such as the structure of the program, data processing in the core, communication between the core and mass storage, input and output systems, etc. In the last part of this paper, two test examples are presented verifying the computational accuracy of this programing system and illustrating its applicability in practical engineering.

Key words: Finite Element Method (FEM) Microcomputer Mining Engineering

1 INTRODUCTION

Microcomputers have been applied extensively in many engineering fields. In recent years, numerical modelling using microcomputers in geotechnical engineering has been paid a great deal of attention[1-3]; it is not only convenient to the user, e.g. the computation can be carried out at all times, but also cuts down the computation costs greatly as the main expense is merely on electric power and printer paper. In addition, the calculation results can be expressed objectively and vividly, utilizing the powerful graphic purpose of microcomputers. Hence, microcomputers have provided a substantial base for the further development and application of numerical methods in geomechanics. As pointed out by Liu Huaiheng[1], with the development and popularization of microcomputers, numerical methods in geomechanics will make a break through in their advance and practical engineering applications in the foreseeable future. Therefore, the development of applicable microcomputer programs for geomechanical numerical methods is essential to the advancement of computational geomechanics.

By comparision with large or mid-sized computers, the main disadvantages of microcomputer are limited memory, and low operation speed. Hence, the key to developing an applicable microcomputer program is to improve the structure of the program and algorithms so as to adapt it to the above characteristics of a microcomputer program system for the 3-D nonlinear finite element method (3-D FEM), which was recently developed for applications in mining engineering by the authors. The focus here is on some of the key aspects of the programming techniques for an applicable microcomputer program, since the basic theory, formulas, algorithms, and procedures for the three-dimensional non-linear finite element method can be found in the relevant lite- ratures[4-6].

[1] Manuscript recieved October 10, 1992

2 STRUCTURE OF THE PROGRAM

The chain program structure that used to be employed in traditional computer programs for the finite element method(FEM)[7-8] is shown in Fig. 1. From here, it can be seen that when a program is run, all modules performing the various basic FEM steps need to be loaded onto the computer at the same time and occupy the memory of computer while it is running. However, as the program usually consists of thousands of statements, and it is often necessary to carry out the computation on over a thousand elements in practical engineering applications, an applicable multiuse program employing that structure is hardly possible to compile and run on a microcomputer.

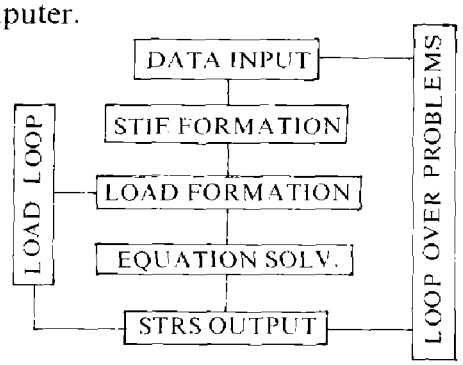

Fig. 1 The chain structure of a traditional program

As shown in Fig. 2, a parallel block structure program, in which each basic FEM procedure may be designed into an independent block program with input and output, has been adopted in our program system. While it is running, the block program makes data exchanges with the mass storage (e.g. fixed disk) by means of data files. Before the program terminates control information is returned to the CCM(Command Control Manager) which decides the next step. All the functional block programs performing the basic FEM steps and subsidiary processing such as preprocessings, post processing, drawing, etc, form an organic system, which may be systematically managed by a CCM.

Fig. 2 shows the parallel block structure of the program From Fig. 2, it can be seen that as the parallel block program system needs only one program block at a time to be loaded and run, the memory of the computer can be used much more efficiently than in program employing a chain structure. The parallel block program system is a the solution limitations posed by microcomputer practical engineering problem.

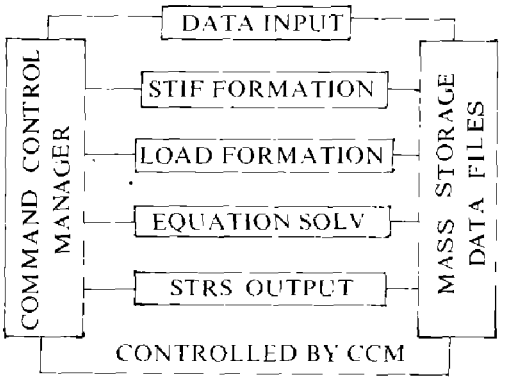

Fig. 2 The parallel block structure of the program

In addition, this program system is highly flexible, conveniently debugged and easily expanded according to the user's requirements. If necessary, it is easy to add a new independent program block to it.

3 DATA PROCESSING IN THE CORE

To improve the problem solving capacity of a program, it is also important to optimize the data processing in the core. Generally, the following reguirement need to be met.

(1) Make maximal use of core memory for a block program.

(2) Improve the efficiency of data exchange within a block program.

(3) Avoid using too many common statements.

(4) Delete temporary arrays automatically.

In the master segment of a block program, set a no-name common block in that a one-dimensional mass is stored. The form is:

```
PROGRAM MAIN
COMMON MT, NP, IA(100000)
MT = 100000
```

in which IA is a one-dimensional mass array and its dimension is adjustable according to the user's

needs.

All operational arrays in the block program will be stored in IA, and their locations in IA are determined by using the statements below

CALL DEFINI(NAME, LC, M, N)

CALL DEFINR(NAME, LC, M, N)

CALL DEFINH(NAME, LC, M, N)

Here, NAME is the name of an integer, real, or character array to be stored. M and N are respectively the row and column number of the array. LC is the first position that the array will occupy in IA, and it is calculated by the above three subroutines. The array name is sequentially recorded at the end of IA to form the array directory. The relative positions of the stored arrays are shown in Fig. 3.

The temporary arrays can be deleted using the below call:

CALL DELETE(NAME)

The purpose of this subroutine is to free the core memory occupied by the "Name" array so that other arrays can be stored.

4 DATA COMMUNICATION BETWEEN THE CORE AND MASS STORAGE

In a parallel block program system, because the data exchanges between any two independent program blocks have to use the mass storage as a medium, communication between the core and mass storage is the key for realizing the success of the parallel block program system. For this purpose, the following problems need to be resolved.

(1) Data must be transmitted from the core to mass storage.

(2) Data must be transmitted from in mass storage to the core.

(3) Connections between the core and mass storage must be established

In our program, the following subroutines are used for the transmission of data from the core to mass storage, their call forms are:

CALL FILEI(II, EXTN, LUN, M, N)

CALL FILER(RR, EXTN, LUN, M, N)

CALL FILEH(HH, EXTN, LUN, M, N)

Here, II, RR, and HH are respectively the names of the integer, real and character arrays to be transmitted. EXTN is the extension name of the data files in mass storage. LUN is the channel number of the files. M and N are the row and column number of the array respectively.

Conversely transforing data from the mass storage to the core may be carried out by the following subroutines, the call forms of which are

CALL RFILEI(II, EXTN, LUN, M, N)

CALL RFILER(RR, EXTN, LUN, M, N)

CALL RFILEH(HH, EXTN, LUN, M, N)

in which the meanings of the parameters are the same as above.

Before those calls are carried out, the channels between the core and mass storage must be established. This can be done through a number of related calls CALL ROSET, the function of which is to define the number of channels, and the byte number of an integer, real and character variable. CALL POPEN (LUN, EXTN, IT), the function of which is to open the formatted sequential file, if IT = 0, the file is old, if IT = 1, it is new. CALL FOPEN(LUN, EXTN, IT), open a new (IT = 0) or old (IT = 1) unformatted sequential file. CALL DOPEN(LUN, EXTN, MI, MR, MH, IT), open a new (IT = 0) or old (IT = 1) direct file, ML, MR, MH are respectively the numbers of integer, real and characteristics variables. CALL FCLOSE(LUN), close the file channel so that it may be used by the other files.

It can be seen the above approach to data communication between the core and mass storage is clear, simple, and efficient. A program written by using these calls is easy to read and free of errors and confusion resulting from too many inconsistent statements. If the subroutines frequently called in a program system are assembled into a separate, common aid block, the efficiency of FEM programming can be improved greatly.

A MASS ARRAY IA IN NO-NAME COMMON BLOCK							
STORAGE OF ARRAY AA	STORAGE OF ARRAY BB	STORAGE OF ARRAY CC	RESIDUAL MEMORY IN THE MASS ARRAY IA	ARRAY DIRECTORY			
				AA	BB	CC	IA

Fig. 3 Dynamic location for operational arrays in IA

5 INPUT AND OUTPUT

Large amounts of data preparation and the resuting processing have greatly restricted the practical engineering applications of 3-D FEM. The most traditional FEM programs with a fixed-format input system[3,9,10], generally require the user to prepare the input data carefully and strictly in accordnce with the formats fixed by the user instructions. Such work is cumbersome and uninteresting, while mistakes are easily make and especially difficult for a beginner. Although more recent free-format input systems have avoided some of the shortcomings of the fixed-format input system, they are still unable to give the user "true free dom". Because most traditional programs usually just yield output data such as stress, strain and displacement, to obtain more interesting results the user carry out further processing, or put the output data files into a computer again for drawing. Despite having cut down much of the manual work, the amount of working is still quite large. In order to improve the input and output, a character-data input system and a user-computer output system have been employed in our program.

5.1 *Input System*

In the character-data input system, the input data are divided into many data sections in light of their purposes. Each data section is led by a characteristic name (a character string) given by the user. When some section is required, the program will automatically seek the characteristic name from beginning to end in the input data file and then read in the data following it. The call forms are as follows.

CALL FIND(NAME, KN) searches for a leading character string "NAME" in the input data file. If not found, return KN=1, and the program stops to check for an error. Otherwise, return KN=0, and carry out some or all of the following calls are carried out

CALL REASI(HEAS, IFL, NI, I1, I2, I3, · · ·) CALL READR(HEAD, IFL, NR, R1, R2, R3, · · ·) CALL READH(HEAD, IFL, NH, H1, H2, H3, · · ·) read the data behind "HEAD", and respectively assign them to the integer variables I1, I2, I3, · · · ; the real variables R1, R2, R3, · · · ; and the character variables H1, H2, H3, · · · .

CALL READIA(HEAD, IFL, NI, IARR) CALL READRA(HEAD, IFL, NR, RARR) CALL READHA(HEAD, IFL, NH, HARR) read the data behind "HEAD", and respectively assign them to the integer array IARR and the real array RARR; the character array HARR.

The following points should be noted (1) This data input system has no restrictions input data type (e.g., integer, real). Data will be automatically transformed into the required forms on execution; thus it is convenient to the user. (2) It has no restriction of "column format" and "row sequence" to the input data file, which may be formed by the assembly of many data sections; hence it is able to improve the efficiency of input data. (3) The input data file is self-explanatory, so that the meanings of various variables and data sections are clear at a glance. This will greatly decrease the probability of making a mistake in preparing input data. If unfamiliar with this program, the user can prepare input data for his(her) problems at all times with no user instructions.

5.2 *Output System*

According to the control parameters given by the user from the keyboard or mass storage, this user-computer output system can immediately search for the results of interest to the from the unformatted data files recording such basic results as stresses, strains, displacements, equivalent node forces and coordinates, and display or plot their drawings after transformation and interpolation. If subsidiary results such as energy released safety factor, plastic zone, tensile zone etc., are required, the program evaluates them based on the related formulas and the above data, then display to plot them.

This output system, an independent program block, consists of five basic modules, i.e., search module, transformation module, interpolation module, view module, and plot module, which are called by the master segment.

6 FUNCTIONS AND RUN ENVIRONMENT

6.1 *Functions of Program*

It is well known that mine rock mass is a nonlinear, heterogeneous geomedium in that there exist large numbers of discontinuities such as fracutures, cracks, fissures, and faults; and mining is a dynamic technological process, including stoping, caving, filling, dumping, etc. It was in light of the above features that our program system was developed for applications in mining engineering. Using this program system, these features can be simulated: mechanical properties such as elasticity, plasticity, the absence of tension, or low antitension; geometrical discontinuities such as joints and faults; and engineering activities such as excavation, filling, caving, dumping or damming. Comprehensively, it can be applied to the analysis of the stability of mining structures, the mechanical optimization of mining systems, the selection of mining scheme, etc.

Considering the basic FEM procedures and pre and postprocessing, our program system has been designed in ten independent block programs. Their functions are briefly described as follows:

(1) TMDAT1 automatically generates the geometrical data of a 3—D FEM model, e.g., mesh, node numbers, element numbers, boundary conditions, etc;

(2) TMPLOT displays or plots the mesh drawing with node and element numbers, including sectional drawing, three dimensional perspective drawing;

(3) TMDAT2 automatically generates the material data of a 3—D FEM model, and assigns initial stresses to the elements;

(4) TMBAND minimizes the band width re-numbers the nodes;

(5) TMORDE pre and processing necessary to solve large coupled equations, and checks if the size of the array has exceeded the value specified in block TMSOLV;

(6) TMLOAD generates the load data for the model and loops over load steps;

(7) TMSTIF assembles element stiffness and load terms into an overall stiffness matrix and load vector, and checks if the equilibrium condition is statisfied;

(8) TMSOLV solves equations;

(9) TMSTRS computes stresses and strains and checks to see if an element, or part of an element, has yielded;

(10) TMOUTP outputs of the results to a display, printer or plotter with drawings, if needed first evaluating subsidiary quantities such as, safety factor, etc.

6.2 *Run Enviroment of the Program*

All program of this system are written in FORTRAN 77, and can be run on IBM—PC / AT, 286, 386 series microcomputers which have a 640KB memory, a fixed disk drive with more than 20MB, and a math co-processor. It is practically shown that the run time is acceptable for

the solution of a medium of large problem. For example, the run time is about three hours for the solution of a problem of 500～6000 degrees of freedom[11].

7 TWO EXAMPLES

In order to verify the accuracy of this program system and illustrate its applicability two examples are presented in this section.

The first example is a linearly elastic half infinite body subjected to a normal concentrated force, the analytical solution of which is given in Ref. 12. Fig. 4 is a 3–D FEM mesh of quarter of this problem (because of symmetry). The comparisons of the numerical solution and the analytical solution are shown in Fig. 5, from which it is evident that there is a good agreement with the relative error below 3%.

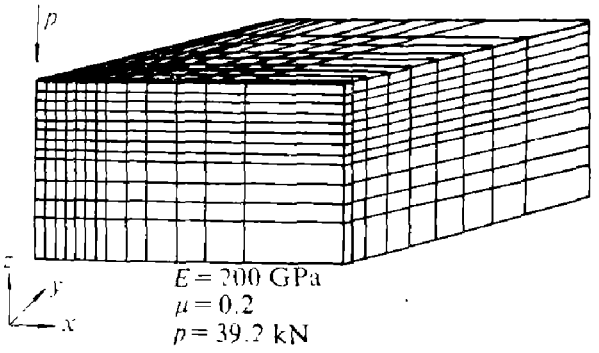

Fig. 4 FEM mesh for example one

The second example is concerned with the stability of a stope roof of an iron mine in Northern China. The orebody is locatd at a depth of 150～200 m dips at approximately 12° to the southwest, and is 500～600 m long in the strike and 300～400 m wide in the dip, and is 12 m high. The rocks contain limestone, lime rudyte, Skarn and diorite, with no large faults near the orebody.

The 3–D model was divided into 16808–node isoparametric elements with a sum of 2145 nodes. The run time was about 4～5 hours. The final distribution of the maximum tensile stress on the immediate roof is presented in Fig. 6.

From this example, it can be shown that this microcomputer program system is applicable to the solution of medium or large practical engineering problems. In addition to the great eco-

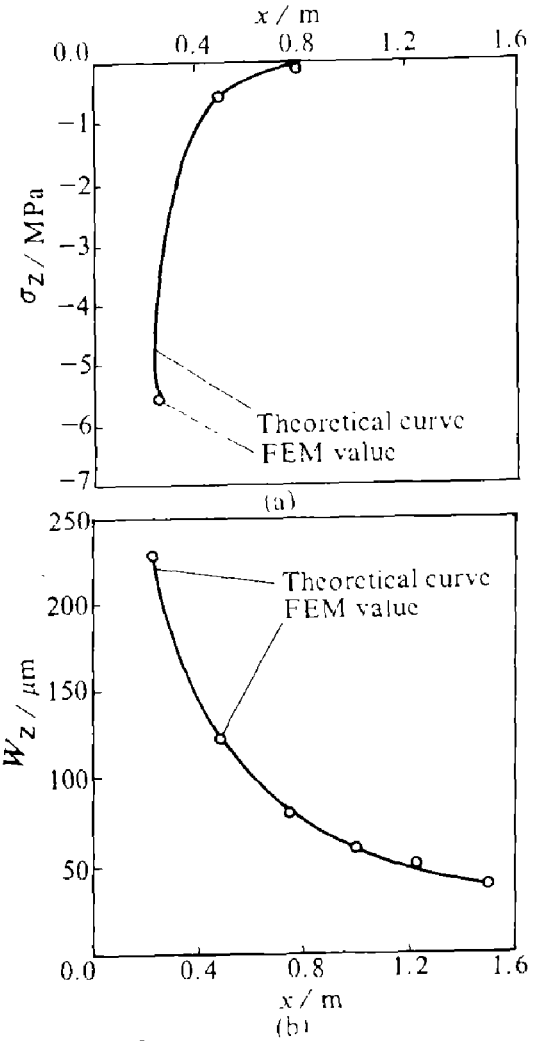

Fig. 5 Comparisons of the numerical and analytical solutions (z = −0.25 m)

(a)—Stress, σ_z; (b)—Displacement, W_z

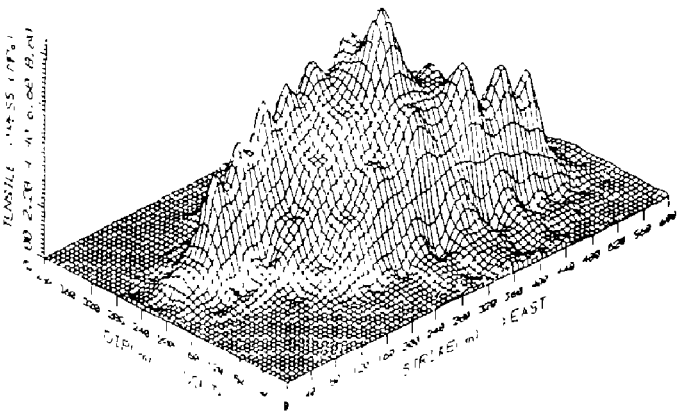

Fig. 6 The final distribution of the maximum tensile stress on the immediate roof

(To be continued on page № 26)

5 CONCLUSIONS

(1) Dry HGMS technique which includes vibration system and a new kind of disperser is effective for the purificiation of kaolin powder;

(2) The vibration of matrix propels the particles through the matrix and assists cleaning;

(3) The disperser can break down particle clusters and enable ultrafine powder to be completely dispersed;

(4) Under appropriate conditions, a product containing 0.90 % Fe_2O_3, can be produced from a koalin powder originally as sayed to contain 2.2 wt.−% Fe_2O_3. There is a recovery of 70 % of the iron free kaolin powder.

REFERENCES

1 Iannicelli, J.. In: Proc. 13th IMPC. Warsaw, 1979: 105.

2 Iannicelli, J.. Clay and Clay Minerals. 1976, 24: 64.

3 Llofthouse, C. H., Scobie, C. W.. In: Proc. 13th IMPC. Warsaw. 1979: 163.

4 Emory, B. B.. US Patent 3, 471,001.

5 Luborsky, F. E.. In: Report of Investigation. EPA−600 / 7−78−208. 1978.

6 Kolm, H. H.. US Patent 3,676,337. (1972).

7 Roux, E. H.. In: Proc. Inter Conf MINTEK 50. Johnnesburg, 1984, Paper A6 / 2.

8 Bailey, A. G.. Powder Tech. 1984, 37: 71.

(*Continued from page* № 15)

nomical benefits, the runtime is also acceptable.

REFERENCES

1 Liu, H. H.. Rock and Soil Mechanics. 1989. 10(2).

2 Cai, Y. J.. Doctoral Thesis. CSUT. 1991.

3 Wu L., Qu S., Deng C.. A Microcomputer Program for Structural Analysis. Beijing University Press. 1988.

4 Zienkiewicz O. C.. The Finite Element Method. McGraw-Hill. 1977.

5 Oden J. T.. Finite Elements of Non-linear Continua. McGraw-Hill. 1972.

6 Desai C. S., Abel J. F.. An Introduction to the Finite Element Method. Van Nostrand Reinhold. 1972.

7 Hinton E., Owen D. R. J.. Finite Element Programming. Academic Press. 1977.

8 Owen D. R. J., Hinton E.. Finite Elements in Plasticity. Pineridge Press. 1980.

9 Bathe K. J. et al. SAP5−A Structural Analysis Program for Static and Dynamic Response of Linear Systems. Univ of Calıforlia. Berkely. 1976.

10 Bathe K. J.. ADINA−A Finite Element Program for Automatic Dynamic Incremental Nonlinear Analysis. M I T. Cambridge. 1981.

11 Ma M. J.. Doctoral Thesis. Univ of Sci and Tech of Beijing. 1992.

12 Qian W. C., Ye K. Y.. Elasticity (2nd edition). Science Press. 1980.