NASA Contractor Report 4544

Application of Symbolic and Algebraic Manipulation Software in Solving Applied Mechanics Problems

Wen-Lang Tsai and Noboru Kikuchi

CONTRACT NCA2-136
August 1993

National Aeronautics and
Space Administration

NASA Contractor Report 4544

Application of Symbolic and Algebraic Manipulation Software in Solving Applied Mechanics Problems

Wen-Lang Tsai and Noboru Kikuchi

University of Michigan
Department of Mechanical Engineering
Ann Arbor, Michigan

Prepared for
Ames Research Center
CONTRACT NCA2-136
August 1993

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

 INTENTIONALLY BLANK

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

NOMENCLATURE

SAM	symbolic and algebraic manipulation
V	strain energy; potential energy; velocity
Ω	domain of integration
N	stress resultant; shape function
$N_{\alpha\beta}$	stress tensor
M	moment resultant; mass matrix
$M_{\alpha\beta}$	moment tensor
K	stiffness matrix
ε_{ij}	strain
ε	smoothing parameter
κ	curvature of curve
x, y, z	Cartesian coordinates
u, v, w	displacement components
R	radius of curve
L	linear operator; Lagrangian; length
t	time
I	imaginary symbol
q	load per unit; rotation tensor
P	position vector
k	spring constant
θ	angle
T	kinetic energy
$a_{\alpha\beta}$	metric tensor
D_α	operator of covariant derivative
D	material property matrix; flow matrix
$d_{\alpha\beta}$	curvature tensor
$E_{\alpha\beta}$	strain tensor; Young's modules
F_α	force components in surface of shell
$p_{\alpha\beta}$	displacement gradient
$K_{\alpha\beta}$	bending tensor
ϑ	determinant of displacement gradient
ν	poisson's ratio
[J]	Jacobian matrix

w_i	weight parameter
s_i	area coordinates of triangular element
ξ, η	natural coordinates of quadrilateral element
σ_{ij}	Cauchy stress
$\bar{\sigma}, \bar{\varepsilon}$	effective stress and strain rate
$\tilde{f}$	friction stress
U	velocity vector
λ	Lagrange multiplier
u^d	die velocity
Δ	area
ΔU	difference between U_i and U_j
d	Generalized displacement vector
h	thickness of shell
m	friction factor
g	yield shear stress

SUMMARY

Compared to numerical analysis, symbolic and algebraic manipulation is unfamiliar to people in the research field. As its name implies, symbolic and algebraic manipulation can be simply interpreted as a computerized operation which can retain symbols throughout computations and express results in terms of symbolic forms. For example, the coefficients *a*, *b*, and *c* in the quadratic polynomial equation $ax^2+bx+c=0$ do not need to be known in order to find its roots. The equation itself can be directly input to a computer and the results will be

$$x_1 = \frac{(-b+\sqrt{b^2-4ac})}{2a} \quad \text{and} \quad x_2 = \frac{(-b-\sqrt{b^2-4ac})}{2a}$$

If the numerical values are required, three coefficients can be specified and solutions will be expressed as numbers.

From the example above, at least two unique characteristics of symbolic and algebraic manipulation can be observed. First unlike numerical analysis, the solutions from symbolic and algebraic manipulation are exact and therefore no round-off error is introduced. Second, the solutions are the same as those derived by hand. Therefore the extension of human capability to handle more sophisticated fomulations becomes feasible by computer.

In the first chapter of this report, the history of symbolic and algebraic manipulation is introduced. The sencond chapter chronologically reviews the literature regarding the application of symbolic and algebraic manipulation in the engineering field. The capabilities of symbolic and algebraic manipulators are demonstrated in chapter three by selected examples. Chapters four through six demonstrate applications of symbolic and algebraic manipulation. Chapter four describes the automatic formulation of applied mechanics problems, chapter five covers the materially nonlinear, rigid-plastic ring compression problem, and chapter six discusses plate problems. The final chapter summarizes the overall conclusions of this report.

It is well known that there are some difficuilties existing in the symbolic and algebraic field. The report proposes a remedy to avoid the difficulties and successfully accomplishes the applications. Due to this breakthrough, the solution of some previously insolvable problems become available. In addition, one of the advantages found in this research is believed to be crucial for improving the execution efficiency of numerical programs.

CHAPTER I

HISTORY OF SYMBOLIC AND ALGEBRAIC MANIPULATION

1.1 Introduction

Symbolic and Algebraic Manipulation (abbreviated as SAM) software is one of the new products of modern technology for use with highly developed digital computers. Traditionalists might say that SAM software is a misuse of modern computers. This is true if the viewpoint is adopted that a computer is a machine which only counts numbers. This viewpoint, however, severely limits the emerging artificial intelligence capabilities of computers. For instance, in addition to numbers, there are many symbols which define appropriate mathematical relations in a calculus book. Can we ask a computer to do these analytical derivations for us? This is a great question which finally led to the birth of SAM and added "soul" to the computer, to make it think more like a human brain. This idea, which originated before 1953, has had an impact on a variety of fields, such as science, industry and education. Therefore, although its history is not as long as that of the classical sciences, its impact has been so large that a record of its history is deserving.

At the initiation of this dissertation effort in 1986, there were already a number of SAM systems available on the market. Some of them were more than ten years old, such as FORMAC, REDUCE and MACSYMA. Others were just being developed, such as muMATH and MATHEMATICA. Ironically, most relevant documents either ignore the history of these systems or just skim over them briefly. Only one book, written in 1969 by Jean E. Sammet [1], includes historical details, however, it is too old to cover recent developments. Most of the major systems used today have been produced since then. Therefore, it is necessary to collect, rewrite, and update the history of the SAM systems. It is hoped that the interested researchers will get a complete picture of the development of SAM systems. Through an understanding of the history they will be able to grasp the direction of the field and devote themselves towards making a further contribution. This is the major purpose of the chapter. It is much more important than just knowing how to run the SAM systems.

The first idea for using computers to do SAM can be traced back to two master theses published in 1953 ([2],[3]). Three years later, what is believed to be the earliest SAM system, called PM, was developed at IBM [4]. Now there are many SAM systems on the market for various computers. Some of them are designed for general purpose usage, while others have been developed for particular applications. Generally speaking, the evolution of symbolic and algebraic manipulation can be classified into three stages. They are :

1. The first generation (1953-1965)---software to appear in this generation was PM, ALGY, FORMAC, MATHLAB and ALTRAN. Because of the limitation of hardware capacity, the systems in this stage were small in size and immature in content. Therefore, most of them became obsolete or were revised.

2. The 2nd generation (1966-1975)---software to appear in this generation was REDUCE and MACSYMA. These systems took advantage of the improvement of hardware memory capacity. They contain many built-in functions, are large and are for general purpose usage. All of them run on the mainframe.

3. The 3rd generation (1976- present)---some representatives of more recent systems are muMATH, MATHEMATICA, and DERIVE. Unlike the systems of the second generation, the systems in this generation are designed to run on microcomputers. This has been possible due not only to the improvement of memory capacity in microcomputers, but also due to the requirement of most users who just need quick checks or moderate manipulations.

Details of the histories of the systems will be described in the following subsections individually. For the sake of clarification, a summary is also included in Table 1.1.

1.2 History of SAM systems

1.2.1 PM

PM is believed to be the earliest computerized algebraic system in the world. It was developed by George E. Collins at the IBM research center in Yorktown Heights, New York. Although the first document was published in 1966 [4] its beginning dates back to 1956. Written in assembly language for the IBM 701 computer, PM contained the subroutines for addition, subtraction and multiplication of multiple-precision integers, and subroutines for performing the same operations on multivariate polynomials with multiple-precision integer coefficients. Between 1956 and 1966 PM was reprogrammed for the newer IBM computers

(e.g. 709, 7090 and 7094), and augmented to include new operations (such as integer gcd) with various improvements (e.g. the incorporations of list processing and dynamic storage allocation). In 1966 Dr. Collins became a professor of computer science at the University of Wisconsin. With the aid of graduate students, the PM system was converted to the SAC-1 system in 1973. In spite of its eventual replacement, PM, as the first SAM system, was still very significant in the field.

1.2.2 ALGY

Although ALGY has few functions, it was one of the earliest SAM systems in the world. The developmental work was started at Western Development Lab-Philco Co. in Palo Alto, California by Bernick, Callender and Sanford, around 1961 [5]. It was interactive and allowed expressions written in a notation similar to FORTRAN as input, with some deviations. For example, the *$* was used instead of ** to represent exponentiation, and all natural integers were expressed as fractions, for example, *0* and *1* were denoted by *0/1* and *1/1*, respectively. It only contained a few commands, such as :

- OPEN : expanding the expression in the parenthesis
- SBST : making substitution
- FCTR : factoring a given expression
- TRGA : expanding the *sin(a+b)* into *sin(a)cos(b)+sin(b)cos(a)*

Which is why the authors said that only two hours instruction was enough to use it. Although ALGY didn't come into extensive use, some of its ideas were succeeded by the FORMAC system, which is still popular today.

1.2.3 FORMAC

FORMAC is an acronym of FORmula MAnipulation Compiler. It was developed by J. E. Sammet and Robert G. Tobey at IBM's Boston Advanced Programming Department in July, 1962 [1]. Five months later (December, 1962), the first complete draft of language specifications was prepared and implementation design started immediately thereafter. After 18 months of extensive experiments, the first complete version was successfully running on the IBM 7090/94 computer in April, 1964. For the sake of obtaining feedback from users to make further improvement to the system, FORMAC was released for public use by the authors themselves (not by IBM) in November, 1964. This version of FORMAC was written in

FORTRAN IV. Three years later (November, 1967), the new version of FORMAC written in PL/I was released by the authors for use on IBM/360 systems.

The FORTRAN version of FORMAC kept most commands and notations of FORTRAN IV. In addition, there were a couple of new commands added to allow it to do algebraic manipulations. For example, LET assigns symbols to variables instead of numbers in FORTRAN, SUBST makes substitution, EXPAND removes all parentheses in expressions and COEFF obtains the coefficients of variables. The major PL/I FORMAC capabilities can be divided into the following categories :

1. User control of simplification : EXPAND for expanding the parentheses expression, DIST for applying the distributive law to all products of sums, etc.

2. Substitution : EVAL(expr,a,b) replaces a in expr by b.

3. Differentiation : DERIV performs partial differentiation.

4. Expression analysis : COEFF(expr1,expr2) returns the coefficient of expr1 in expr2. NUM and DENOM return the numerator and denominator, respectively. HIGHPOW and LOWPOW return the highest and lowest power.

5. Storage allocation : SAVE(var) for storing the var to secondary storage.

6. Output : PRINT_OUT(expr) to print out the required expressions.

7. Built-in functions : these include trigonometric, logarithm, exponentiation, square root, hyperbolic function, etc.

8. User defined function : the user can define functions as needed.

FORMAC is now one of the most popular SAM systems. It is the first reasonably general purpose system to receive extensive usage worldwide. With the advantages of longer history and larger numbers of users, its accumulated contributions to SAM field are remarkable. In 1977, the new version, called FORMAC 73, was released to replace the old one.

1.2.4 MATHLAB

MATHLAB[1] system was developed by C. Engelman and his employees at MITRE Co. in 1964 [6]. Its source language is LISP, but the commands are defined as English words. For example, PLEASESIMPLIFY(x,y) is the command to simplify x and name it as y. In the fall of 1967, the first version of MATHLAB was replaced by the second version, MATHLAB 68, which operated on a PDP-6 machine with 256 K core memory. The input and output were through a teletype-like keyboard with a fixed character display scope. The notations in the second version were more ALGOL-like. MATHLAB was the first complete on-line system.

1.2.5 ALTRAN

ALTRAN is a system developed at the BELL TELEPHONE Laboratory in Murray Hill, New Jersey by W. S. Brown, M. D. McIlroy, D. C. Leagues and G. S. Stoller [1]. It was running in late 1964 on the IBM 7090/7094, 7040/44, etc. The basic languages which ALTRAN adapted were a mixture of FORTRAN II and FORTRAN IV. Since it was limited to use in the BELL Lab., its contributions to the SAM field were small.

1.2.6 REDUCE

REDUCE was developed by A. C. Hearn of Rand Corporation, California, in 1963 [7]. At that time, he met Dr. John McCarthy, an inventor of the LISP language, who suggested the use of LISP for the problems of elementary particle physics. Since then, Dr. Hearn, as a theoretical physicist, has worked in the SAM area. In August 1966, the first publication was issued [8]. This paper only talked about the specific application of SAM techniques to elementary particle physics. Two years later (1968), the first paper describing a general algebra system, "REDUCE", was published [9]. The name of REDUCE originated from this paper. Its name is not an acronym. According to the description from the author himself, its name was actually intended as a wit. He said "algebra system then as now, tended to produce very large expressions for many problems, rather then reduce the results to a more manageable form". The system at this time was called REDUCE for distinction from the new version, REDUCE 2, which appeared in 1970. The big improvement was that the whole system was written in an ALGOL-like dialect (call RLISP), rather than the parenthesized notation of LISP in which REDUCE was written. At this time, the REDUCE 2 system was also released to users, making the beginnings of a user community. Thereafter, REDUCE 2 was implemented successfully on

[1] MATHLAB is not to be confused with MATLAB. MATLAB is the numerical software for matrix operations, while MATHLAB is another symbolic and algebraic manipulator.

the Michigan Terminal System (MTS) of the University of Michigan by Mike Alexander. After a long silence, REDUCE 3 was distributed in 1983. Several significantly new packages were added in this version, such as analytic integration, multivariate factorization, arbitrary precision real arithmetic and equation solving. Following REDUCE 3, upgraded versions were also released. They were REDUCE 3.1 released in 1984, REDUCE 3.2 in April, 1985, REDUCE 3.3 on July 15, 1987. Each of them contains bug fixes and additional capabilities. Instead of implementation on MTS, the REDUCE 3.3 was first implemented on the APOLLO workstation in the Computer Aided Engineering Network (CAEN) of the University of Michigan. REDUCE 3.3 was also updated once in January 15, 1988.

REDUCE system has become one of the most well-known SAM systems. Its general purpose design makes it possible to be used in a wide variety of areas. Its contributions are confirmed by the number of papers published in different fields.

1.2.7 SCHOONSCHIP

SCHOONSCHIP was designed by M. Veltman at CERN, Switzerland in 1964 [10]. Its major applications are in the field of high energy physics, but it is sufficiently general to be used for other calculations. It can deal easily with expressions of 10^4 to 10^5 terms on the CDC 6000 computer. It was limited to use within CERN.

1.2.8 ANALITIK

ANALITIK was developed at the Institute of Cybernetics in Kiev, Soviet Union, by the direction of the well known Soviet cybernetician and academician V. M. Glushkov [11]. The first paper discussing the system features was published in 1964. The language it used was ALGOL-like and close to that of traditional mathematical notation and natural language. It possessed interactive and batch processing modes. Since its implementation is highly machine dependent, ANALITIK has only run on the MIR-2 computer.

1.2.9 FLAP

FLAP was written in LISP 1.5 by A. H. Morris, Jr. at the U.S. Naval Weapons Laboratory in Dahlgren, VA. prior to 1967 [1]. Obviously, the FLAP system wasn't released to the public.

1.2.10 SAC

The SAC system was developed by Dr. George E. Collins at the University of Wisconsin, Madison. The first version, SAC-1, was distributed in 1967 [4]. This was a highly portable general purpose system, developed to replace one of the very earliest computer algebra system, PM in IBM [see 1.2.1]. SAC-1 was replaced by SAC-2 in July, 1980. The SAC-2 was programmed in ALDES language, which was designed by Rudiger Loos and G.E. Collins in 1973 to 1974. The SAC-2 system also provided the translator from ALDES to standard FORTRAN to maintain its portability.

1.2.11 MACSYMA

MACSYMA is an acronym of project MAC's SYmbolic MAnipulator. It was originally designed by C. Engelman, W. Matin, J. Moses for project MAC at M.I.T. in 1968. The implementation of it began in July, 1969. The system has quintupled in size since the first paper describing it appeared in 1971 ([12] [13] [14] [15]). It was made available over the ARPA networks in May, 1972. MACSYMA has a lot of built-in mathematical functions and graphic facilities which have made it one of the most powerful SAM systems in the world. Unfortunately, the University of Michigan didn't have it until September, 1988. The one implemented on the APOLLO workstation in CAEN of the University of Michigan still doesn't have a graphics package.

1.2.12 SCRATCHPAD

Although the name of SCRATCHPAD was chosen in 1970, the initial work on it can be traced back to 1965. The SCRATCHPAD system was designed principally by James H. Griesmer, Richard D. Jenks, Fred Blair, David Yun, and their colleagues, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York ([16] [17] [18]). Unfortunately, the name of SCRATCHPAD was not used for the first paper, presented in Bonn in 1970. One year later, a revised version by Dick Jenks, called SCRATHPAD/1, was demonstrated at SYMSAM/II in March 1971. After combining some new features, such as history file (allowing users to backtrack), and system commands, the first completed SCRATCHPAD/1 manual was eventually published in 1975. After this, there seemed a stagnation in the progress of the SCRATCHPAD system due to personnel changes. Jim Griesmer left the group to be a manager of education at IBM research and Dick Jenks went to the University of Utah for a sabbatical. When Dick Jenks returned to Yorktown Heights in the fall of 1977, David Yun agreed to organize the "mode-base" ideas originated by Dick Jenks in 1973. This led to the

NEWSPAD which thereafter was renamed to SCRATCHPAD 84 at the New York conference in 1984. However, the name of SCRATCHPAD 84 was not quite appropriate since it would take more than one year to finish the system. Therefore it was changed into SCRATCHPAD II, which is the name used now. It became available in 1985 for test and evaluation to a limited number of users from an IBM owned mainframe via telenet, CSNET and ARPANET. As yet, it is not commercially available.

1.2.13 CAMAC

The CAMAC system was designed by Vera Pless in 1973 at M.I.T. [19]. The first version of it ran interactively and was written in FORTRAN with sections in assembler language. When Vera Pless moved to Chicago Circle in 1975, the CAMAC system was transferred to the Circle's IBM 370-158 by William Pattern. The name of CAMAC is an acronym of Combinatorial and Algebraic Machine Aided Computation. As the name implies, it was for a specific application.

1.2.14 SHEEP

SHEEP was designed by I. Frick at the University of Stockholm, Sweden in 1975 [(20]. [21]) It was specialized for manipulating components of tensors. The source language it uses is MACRO-10. It runs on DEC PDP 10 and PDP 20. The first version of SHEEP, now called SHEEP 1, was written in assembler code for the DEC-10/20 computer. Unlike the first version, SHEEP 2 is written in standard LISP.

1.2.15 ORTOCARTAN

ORTOCARTAN is written in LISP. It was designed by Andrzej Krasinski in Poland in 1977 [22]. Its name is an acronym and is due to the specific application to the calculation of Riemann, Ricci, Einstein and Weyl tensors from a given metric tensor using an ORTHonormal set of CARTAN forms. Although the author said it could be relatively easily extended for other uses, such as inverting matrices of arbitrary rank, it did not come into wide use.

1.2.16 MAPLE

The MAPLE system was designed by Bruce Char, Keith Geddes, W. Morven Gentleman and Gaston Gonnet at University of Waterloo, Canada in December 1980 ([23] [24]). The name "MAPLE" is not an acronym but rather it was simply chosen as a name with a Canadian identity. There were two goals which oriented MAPLE's design. The first was to be

used on a time sharing mainframe computer. The second was to run it on a microprocessor-based workstation. This was the major difference between MAPLE and REDUCE (or MACSYMA).

1.2.17 muMATH

Written in muSIMP (a LISP-like language), muMATH was designed and developed by David R. Stoutmeyer and Albert Rich at the University of Hawaii in 1977 [25]. The first version was called muMATH-77 and was experimental. Two years later, the Software House, Inc., was founded by David Stoutmeyer and the first product, muMATH-79, was distributed to users for the CP/M-80 operation system or Apple II family Z80 machine with 64 K bytes core memory required. The second product called, muMATH-83, was not released for the IBM personal computer until 1983. The muMATH-83 needs 256 K bytes RAM memory. Recently, DERIVE has taken over the place of muMATH-83. The significant improvement is that DERIVE combines the numerical, algebraic and graphical functions together, rather than just algebraic functions of muMATH. The DERIVE system requires 512 K memory space for normal execution.

1.2.18 MATHLIB & SMP

MATHLIB is an interactive general purpose SAM system. It was originally designed and developed under the auspices of the Department of Mathematics at Harvey Mudd College, California, in 1978. It was one of the products of Innosoft International Inc., of Claremont, California and became commercially available in 1983. It can perform numerical and symbolic operations. In addition, its graphical output is device-independent and allows it to be processed by over 150 different graphics devices. The PRS subroutine embodied in the algebraic subsystem of MATHLIB has more than 250 built-in functions for manipulation of mathematical expressions.

SMP is another product of Innosoft International Inc.. It was designed at the California Institute of Technology. It's written in C language and was originally developed to run on VAX/780 under the UNIX operating system. In addition, there is a special design character in SMP to allow for easy conversion between operating systems. It needs at least 2.5 megabytes of memory space for typical usage.

1.2.19 MATHEMATICA

MATHEMATICA is a recent product of Wolfram Research, Inc. There are several versions of MATHEMATICA for a variety of computers, such as for Apple Macintosh, DEC VAX, IBM ,Cray, and so forth. It was designed and implemented by Stephen Wolfram, Daniel Grayson, Roman E. Maeder, and their colléagues at the University of Illinois in 1988 [26]. It integrates the algebraic manipulation, numerical computation, and graphical functions together and allows the resultant expressions to be outputed in C code, FORTRAN code, and text form. Its source language is C. The memory requirement for normal operation is about 3.7 mega bytes. The MATHEMATICA as well as DERIVE are expected to be two dominant systems in the coming decade.

1.2.20 Mathcad

Mathcad is developed by Mathsoft, Inc. at Cambridge, Massachusetts. The earlier version appeared on market around 1987. This system adopted the core functions of MAPLE and extented itself by including the graphic capacity. It is written in C language. The latest version 3.1 is available in 1992. This newest version can run in IBM, Macintosh PCs and unix based machines. The minimum space requirements are two megabyte RAM and seven megabyte hard disk. Unlike most of the SAM systems, the command inputs in this system are menu driven. This allows users to communicate with machine by simply picking and clicking. This unique feature not only saves users lots of efforts in typing but also reduces human errors which sometimes turn out a unmanageable, hard-to-be-debugged results.

1.3 Conclusion

From a history of the SAM system, we can draw the following conclusions :

- The SAM systems evolved from small, immature systems to well-designed, multi-function systems, to compact systems which can be used on microcomputers.
- At present, there is no unique best system. The definition of the best SAM system depends on many variables, such as computer availability, availability of software, familiarization with software, the problem to be solved, software contents, circumference facility, and so forth.

system	year	remarks
PM	1956	IBM
ALGY	1961	WDLP Co.
FORMAC	1962	IBM - Boston
MATHLAB	1964	MITRE Co.
ALTRAN	1964	BELL Lab.
REDUCE	1963	Rand Co.
SCHOONCHIP	1964	CERN
ANALITIK	1964	Soviet Unions
FLAP	1967	U.S. Navy
SAC	1967	Uni. of Wisconsin
MACSYMA	1968	M.I.T.
SCRATCHPAD	1965	IBM-Yorktown Heights
CAMAC	1973	Vera Pless
SHEEP	1975	Sweden
ORTOCARTAN	1977	Poland
MAPLE	1980	Canada
muMATH & DERIVE	1977	Uni. of Hawaii
MATHLIB	1977	Harvey Mudd College
SMP	1977	Caltech
MATHEMATICA	1988	Uni. of Illinois
Mathcad 3.1	1992	Mathsoft Inc.

Table 1.1 : List of symbolic and algebraic systems

1.4 References

[1] Jean. E. Sammet, "Programming Languages : History and Fundamentals", Prentice-Hall inc., 1969.

[2] H. G. Kahrimanian, "Analytic differentiation by a digital computer", M.S. thesis, Temple University, Philadelphia, Pennsylvania, 1953.

[3] J. Nolan, "Analytic differentiation on a digital computer", S. M. thesis, M.I.T., cambridge, Massachusettes, 1953.

[4] George E. Collins, "PM, A system for polynomial manipulation", ACM, Vol. 9, No. 8, Aug. 1966.

[5] M. D. Bernick, E. D. Callender, J. R. sanford, "ALGY-An Algebraic manipulation program",Proc. WJCC, vol. 19, pp 389-392, 1961.

[6] C. Engelman, "MATHLAB-A program for on-line machine assistance in symbolic computations", Proc. FJCC, vol. 27, pt.2, pp. 117-126, Nov., 1965.

[7] Gerhand Rayna, " REDUCE : software for algebraic computation", Springer-Verlag, 1987.

[8] A. C. Hearn, "computation of Algebraic Properties of Elementary Particle Reactions using a digital computer", comm. of the ACM, 9, pp 573-577, 1966.

[9] A. C. Hearn, "REDUCE-A user oriented interactive system for algebraic simplification", Interactive systems for Experimental Applied Mathematics, pp 79-90, (edited by M. Klerer and J. Reinfelds), Academic Press, New York, 1968.

[10] H. Strubbe, "Presentation of the SCHOONSCHIP system", EURUSAM 74, pp 55-59, 1974.

[11] Nisse Husberg, Jouko Seppanen, "ANALITIK : principal features of the language and its implementation", EUROSAM 74, pp 24-25, 1974.

[12] Joel Moses, "MACSYMA - The fifth year", ACM SIGSAM bulletin EUROSAM'74, Stockholm, pp 105-110, Aug 1974.

[13] "MACSYMA reference Manual",version ten, Mathlab Group, MIT, 1983.

[14] Richard Pavelle, " MACSYMA : capacities and applications to problems in engineering and the sciences", Symbolics Inc., Cambridge, MA,1985.

[15] W. A Martin and R. J. Fateman, "The MACSYMA system", proc. 2nd symposium on symbolic and algebraic manipulation, pp 59-75, ACM, March 1971.

[16] Robert S. Sutor, "The Scratchpad II computer algebra language and system", EUROCAL 85, pp 32-33, vol.12, 1985.

[17] James H. Griesmer, "A history of the SCRATCHPAD Project (1965-1977)", IBM newsletter, vol.1, no. 2, Jan. 15, 1986.

[18] Richard D. Jenks, "A history of the SCRATCHPAD Project (1977-1986)", IBM newsletter, Vol. 1, no. 3, May 15, 1986.

[19] Jeffrey S. Leon, Vera Pless, "CAMAC 1979", symbolic and algebraic computation EUROSAM 79, pp 249-257, 1979.

[20] Lars Hornfeldt, "A system for automatic generation of tensor algorithms and indicial tensor calculus, including substitution of sums", Symbolic and algebraic computation, EUROSAM 79, pp 279-290, 1979.

[21] I. Frick, "The computer algebra system SHEEP, What it can and cannot do in general relativity",Inst. of Theoretical physics, University of Stockholm, 1977.

[22] Andrzej Kransinski, "ORTOCARTAN---A program for algebraic calculation in general relativity", ACM SIGSAM, vol. 17, pp 12-18, 1983.

[23] Bruce Char, Keith Geddes, Gaston Gonnet, "The MAPLE symbolic computation system", ACM SIGSAM, vol. 17, pp 31-42, 1983.

[24] Bruce Char, Keith Geddes, W. Morven Gentleman, Gaston Gonnet, "The design of MAPLE : a Compact portable and powerful computer algebra system", Computer Algebra, proceedings Eurocal'83, pp 101-115, 1983.

[25] C. Wooff, D. Hodgkinson, "muMATH : A microcomputer algebra system", Academic press, 1987.

[26] Stephen Wolfram, "MATHEMATICA---a system for doing mathematics by computer", Addison-Wesley, 1988.

CHAPTER II

SURVEY OF THE LITERATURE ON SYMBOLIC AND ALGEBRAIC MANIPULATION

2.1 Introduction

The documents published in the symbolic and algebraic manipulation field are not as plentiful as those in the area of numerical analysis. However, after a careful classification of the existing documents, one finds that the developmental history is closely related to research directions and content of the publications. In general, the documents about SAM may be divided into four categories. They are :

1. About SAM system itself --- More than half of the existing papers belong to this class. Most of them were published in the period of the first generation. The contents are focused on the following topics :

   (a)The introduction of the new SAM system, including the capacities, functions, etc. [1][2][3].

   (b) The technical reports of softwares [4][5][6][7] .

   (c) The data structure, language and implementation [8][9].

2. Applications to science --- This class of publications is the second largest of the existing SAM papers. One of the major impetuses in developing SAM systems was due to the requirements from scientists, especially in the fields of elementary particle, general relativity and celestial mechanics. Some of the famous examples were collected in the paper by Hearn [10]. One of them is the recalculation of Delaunay's moon coordinates by Deprit, Henrard and Rom in 1970 [11] The others are such as Campbell and Hearn's analysis of the Feyman diagram [13], Rudiger Loos' work about Archimedes' cattle problem [14], and Roberts' and Boris' on the solution of partial differential equations [15].

3. Applications to engineering --- In fact, most engineering problems are not solvable analytically. Therefore, numerical approximation usually predominates in the solution of engineering problems. This is one of the reasons why the engineering applications of SAM has not been as popular as those on science. However, there are two factors which necessitate the use of symbolic and algebraic manipulation in engineering. The first is that the accuracy requirement of a solution of numerical approximation becomes more and more strict today. The second is that the problem to be solved usually involves more sophisticated algebraic manipulation due to the more strict requirements of solutions. Due to these factors, the applications of symbolic and algebraic manipulation to engineering problems has became more popular recently. Some of the publications, such as those written by Madson, Smith and Hoff [16], Levi [17], Wilkins [18], Noor and Anderson [19], Korncoff and Fenven [20], Steinberg and Roache [21], will be discussed in more detail in the next paragraph.

4. Application in the other fields --- In addition to science and engineering, Symbolic and algebraic manipulation has been applicable in other fields, such as information management [22], education [23][24] and business [25].

As time goes on, more and more applications will be reported in various fields. This is due to the fact that :

1. The ongoing improvement in the memory space of hardware systems, especially personal computers.

2. The availability of variously sound SAM software systems.

However, in order to see that scratch paper is replaced by the computer screen in all areas, the people in the educational field should assume the responsibility of utilizing this new tool. As the discussion in the paper, written by Richard Pavelle in 1985, points out [25], only about 20 percent of people in the related field are aware of the existence of SAM system and less than a quarter of these actually use them.

2.2 Reviews of SAM applications in engineering

(1) Computer algorithms for solving non-linear problems

A paper [16] published in 1965 is believed to be the earliest document which employed computerized symbolic and algebraic manipulation to solve an engineering problem. The

authors, W. A. Madson, L. B. Smith and N. J. Hoff, developed their own software to find the solution for the post-buckling behavior of thin-walled circular cylindrical shells under axial compression. The major commands developed by them were :

*SERIESMULT : To expand the expressions, e.g. $(a*sin(x)+b*cos(y))^n$.

*TRIGSPAND : To treat non-double trigonometric terms, such as $sin^2(x)$, $sin(x)*sin(2x)*cos(y)$ etc., into double trigonometric terms like $cos(y)*cos(x)$.

*SEARCHSTORE : To search and collect the coefficients of like trigonometric function, then store them.

*NEWTNRAPH : To solve the nonlinear system equations obtained from the calling of the last three commands by using the Newton-Raphson iteration method.

The application of the above commands to the shell post-buckling problem started at the assumption of radial displacement,

$$w=t\ \Sigma A_{ij} cos(i\pi x/\lambda x)cos(j\pi y/\lambda y)$$

Then by the strain-displacement relationship and Hook's law, the stresses could be obtained. As the stresses (therefore the Airy stress function) were known, the membrane energy, bending strain energy and the potential of axial load could be derived. The resultant total potential energy was then minimized with respect to the coefficients of radial displacement w. The system equations obtained after the minimization then could be solved by calling the NEWTNRAPH command.

(2) Symbolic algebra by computer-applications to structural mechanics

One of the earliest publications of symbolic manipulation application in the engineering field was in 1971, when only a few SAM systems existed. Only REDUCE and FORMAC were mentioned in this paper. At the beginning of the paper [17] by I. M. Levi, he described the story of SAM application in seeking the minimum theoretical post-buckling load for a thin circular cylindrical shell under axial compression. Starting from 1941, Von Karman and Tsien indicated that a low post-buckling load could be found with only two terms included in the series expression of normal displacement. This inconsistency in the Von Karman-Tsien's solution was not found until J. Kempner increased the series into three terms and found a further lowering of the post-buckling load in 1954. Since then, additional investigations were conducted as new terms were added into the series solution. But finally everybody was limited

by their inability to solve the complex algebraic equations without error. The drive to seek the minimum load did not end until 1965 when Madson, Smith and Hoff of Stanford University wrote the special program in ALGOL to increase the series into 14 terms and found the minimum load approached zero, which revealed the basic fallacies in the application of the Karman-Tsien procedure.

The second topic in the paper talked briefly about the derivation of a stiffness matrix for a compatible triangular plate bending element by symbolic and algebraic manipulation. Then an example of the calculation of creep strain rate in plate and shell problems was demonstrated using SAM. This computation started from the x, y components of stress which were expressed as a double trigonometric series, followed by the calculation of equivalent stress, and ended in the substitution of the above quantities into strain rate equations. The resultant fortran codes were printed out by REDUCE. The paper ended with a brief discussion on the REDUCE capacities.

(3) Applications of symbolic algebra manipulation language for composite structures analysis

This paper [18] was published in 1973 by Dick J. Wilkins, Jr. of General Dynamics/Convair Aerospace Division, Fort Worth, Texas. The author used PL/I FORMAC to calculate the strain energy for an anisotropic shell. The strain energy can be written as

$$V = \frac{1}{2}\int_{\Omega} \begin{Bmatrix} N_x \\ N_y \\ N_{xy} \\ M_x \\ M_y \\ M_{xy} \end{Bmatrix}^T \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{xy} \\ \kappa_x \\ \kappa_y \\ \kappa_{xy} \end{Bmatrix} d\Omega \qquad (2.1)$$

Where

- N : stress resultants.
- M : moment resultants.
- ε : mid-plane strain.
- κ : curvature.
- Ω: shell surface area.

• V : strain energy.

and the constitutive equations are expressed as follows :

$$\begin{Bmatrix} N_x \\ N_y \\ N_{xy} \\ M_x \\ M_y \\ M_{xy} \end{Bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & B_{11} & B_{12} & B_{13} \\ A_{12} & A_{22} & A_{23} & B_{12} & B_{22} & B_{23} \\ A_{13} & A_{23} & A_{33} & B_{13} & B_{23} & B_{33} \\ B_{11} & B_{12} & B_{13} & D_{11} & D_{12} & D_{13} \\ B_{12} & B_{22} & B_{23} & D_{12} & D_{22} & D_{23} \\ B_{13} & B_{23} & B_{33} & D_{13} & D_{23} & D_{33} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{xy} \\ \kappa_x \\ \kappa_y \\ \kappa_{xy} \end{Bmatrix} \tag{2.2}$$

Where : A_{ij}, B_{ij}, D_{ij} are the pertinent constitutive components of Hooke's Law. The equation (2.1) and (2.2) can be combined into the form

$$V = \frac{1}{2}\int_{\Omega} [\{\varepsilon\}^T [A]\{\varepsilon\} + 2\{\varepsilon\}^T [B]\{\kappa\} + \{\kappa\}^T [D]\{\kappa\}] d\Omega \tag{2.3}$$

Then the displacements are approximated as

$$w = \sum_m \sum_n C^w_{mn} X_m Y_n \tag{2.4}$$

$$u = \sum_m \sum_n C^u_{mn} \frac{\partial X_m}{\partial x} Y_n \tag{2.5}$$

$$v = \sum_m \sum_n C^v_{mn} X_m \frac{\partial Y_n}{\partial y} \tag{2.6}$$

Where the C_{ij} are constants to be determined by Rayleigh-Ritz method. By Vlasov shell theory, the strain-displacement relations are

$$\varepsilon_x = \frac{\partial u}{\partial x} \tag{2.7}$$

$$\varepsilon_y = \frac{\partial v}{\partial y} + \frac{w}{R} \tag{2.8}$$

$$\varepsilon_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \tag{2.9}$$

$$\kappa_x = -\frac{\partial^2 w}{\partial x^2} \tag{2.10}$$

$$\kappa_y = -\frac{\partial^2 w}{\partial y^2} - \frac{w}{R^2} \tag{2.11}$$

$$\kappa_{xy} = -2\frac{\partial^2 w}{\partial x \partial y} - \frac{1}{R}\frac{\partial u}{\partial y} + \frac{1}{R}\frac{\partial v}{\partial x} \tag{2.12}$$

Where R here is the radius of shell curvature.

With the assumption of a symmetric constitutive matrix, the calculation of the integrand of equation (2.3) was done by FORMAC by the substitution of equations (2.4) to (2.12) into (2.3). The Rayleigh-Ritz method was then applied to take the partial derivative with respect to all undetermined constants in the displacement series. This was also manipulated by FORMAC. The resultant expressions were the energy variation which is in terms of $\frac{\partial X_m}{\partial x}$, $\frac{\partial y_n}{\partial y}$ and their derivatives. Since FORMAC was incapable of performing the symbolic integration at that time, the informal "symbolic integration" was done by examining each term in the energy variation for a specific combinations of derivatives. Each time a certain type was found, it was replaced by a symbol and an appropriate constant to allow for the non-dimensionalization of the integrals. There were a total of twelve different integrations in the energy variation equation. The final expressions were then slightly modified into FORTRAN code by adding DO loops and suitably changing the indices by hand.

(4) Computerized Symbolic Manipulation in Structural Mechanics --- Progress and Potential

In the beginning of the paper [19], A. K. Noor and C. M. Anderson introduced the symbolic and algebraic manipulator MACSYMA. These included the brief history, basic capacities and special commands, as well as associated packages.

The second part of the paper gives three applications in the structural mechanics field by using MACSYMA. They are :

1. Generation of characteristic arrays of finite elements for a shear flexible shallow shell element --- There were three types of basic integrals for linear problems and three types of basic integrals for geometrically nonlinear problems. These were

   (a) Linear problems

$$A^{ij} = \int_{\Omega^{(e)}} N^i N^j d\Omega \tag{2.13}$$

$$B_\alpha^{ij} = \int_{\Omega^{(e)}} N^i \partial_\alpha N^j d\Omega \tag{2.14}$$

$$C_{\alpha\beta}^{ij} = \int_{\Omega^{(e)}} \partial_\alpha N^i \partial_\beta N^j d\Omega \tag{2.15}$$

   (b) Geometrically nonlinear problems

$$C_{\alpha\beta}^{ijk} = \int_{\Omega^{(e)}} N^i \partial_\alpha N^j \partial_\beta N^k d\Omega \tag{2.16}$$

$$D_{\alpha\beta\gamma}^{ijk} = \int_{\Omega^{(e)}} \partial_\alpha N^i \partial_\beta N^j \partial_\gamma N^k d\Omega \tag{2.17}$$

$$E_{\alpha\beta\gamma\rho}^{ijkn} = \int_{\Omega^{(e)}} \partial_\alpha N^i \partial_\beta N^j \partial_\gamma N^k \partial_\rho N^n d\Omega \tag{2.18}$$

The evaluation of integrals in equations (2.13) and (2.14) can be performed analytically by MACSYMA. However in general the integrand of equations (2.15) to (2.18) cannot be integrated exactly due to the existence of a Jacobian determinant in the denominator of the integrand[2] Therefore the hybrid approach (numerical quadrature plus symbolic manipulation) was proposed. The number of integrations to be performed can be substantially reduced by the help of permutative and Dihedral symmetries.

2. Evaluation of effective stiffness and mass coefficients of continuum models for repetitive lattice structures --- The symbolic manipulations by MACSYMA included the evaluation of strain components, calculation of strain energy (with the thermoelastic strain energy) and kinetic energy, computation of stiffness and thermal coefficients as well as effective mass coefficients, forming the Lagrangian of the system and finally obtaining the governing differential equations. The numerical analysis started as soon as the governing equations were obtained. This numerical analysis was also done in MACSYMA. The results of mode shapes were then plotted out by MACSYMA's graphic facility.

3. Application of the Rayleigh-Ritz technique to the free vibration analysis of laminated composite elliptic plates --- The tasks done by MACSYMA in this application were

   (a) Selecting approximation functions for each of the fundamental unknowns displacement amplitude with undetermined coefficients and developing analytic expressions for the specific strain and kinetic energies as quadratic functions of the undetermined coefficients.

   (b) Differentiating specific strain and kinetic energies with respect to the undetermined coefficients symbolically.

   (c) Evaluating stiffness and mass coefficients by performing integrations over volume.

---

[2]This has been done successfully, see the details in chapter three.

(d) Simplifying the expressions for the nonzero stiffness and mass coefficients and developing FORTRAN code.

After stiffness and mass coefficients had been evaluated, the vibration frequencies and mode shapes could be obtained numerically by using any scheme for generalized eigenvalue problems.

The last part of the paper discussed the problems which limited the applicability of computerized symbolic manipulation. The major problems mentioned in the paper were summarized as follows.

1. Production of large expressions during the computation (intermediate expressions swell).

2. Slow speed of symbolic computation.

3. Low portability of large symbolic manipulation systems.

4. Need for analyst interaction during the symbolic computation.

5. Inability to estimate the storage requirements and CPU time for symbolic computations.

6. Problems associated with interface between algebraic and numerical calculations.

In addition, the authors suggested the directions of future research in this field. They were

1. Reduction of a general (tensor) formulation of structural mechanics problem to its computational level.

2. Hybrid computations.

3. Approximate symbolic integration of rational functions.

(5) Symbolic generation of finite element stiffness matrices

As the title implies [20], the authors A. R. Korncoff (Boeing computer service, Seattle WA) and S. J. Fenves (Carnegie-Mellon University) used the symbolic processor MACSYMA to assist in the development of a software to generate the stiffness matrices for finite element analysis. These included the construction of the strain-displacement matrix, calculation of the

determinant of the Jacobian, and multiplication of relevant matrices. The integrand was then integrated symbolically if it was integrable (e.g. the constant strain triangle element). Otherwise it would be output as the function of the problem parameters for further numerical evaluation (e. g. four-node quadrilateral element)[3] . In addition, the software gave two options in the material property matrix. One was "user-supplied" material property. The other was "library supplied" which provides one-dimensional elasticity, plain stress, plain strain, axisymmetric, and 3-D linear isotropic elasticity The example of constructing the isoparametric formulation for constant strain triangle was also shown in the appendix of the paper.

(6) Symbolic manipulation and computational fluid dynamics

The authors S. Steinberg and P. J. Roache [21] employed the symbolic and algebraic manipulator VAXIMA, a VAX version of MACSYMA, to transform the physical differential equation and the boundary conditions into the rectangular region and then constructed the so called stencil coefficient matrix for a finite difference scheme. The major ideas came from the general elliptic problems. In physical coordinates, the linear elliptic equation could be expressed as

$$Lf = \sum_{i,j=1}^{n} a_{ij} \frac{\partial^2 f}{\partial x_i \partial x_j} + \sum_{i=1}^{n} b_i \frac{\partial f}{\partial x_i} + cf + d \tag{2.19}$$

Where a_{ij}, b_i, c, d were given and were the function of coordinates in general.The problem was to find a numerical approximation solution which satisfies equation (2.20) and the given boundary conditions.

$$Lf = 0 \tag{2.20}$$

Since the physical domain is not regular in general, it is necessary to transform the physical coordinates into the rectangular, computational coordinates in which the finite difference scheme could be constructed easily. This coordinate transformation involved the calculations of the Jacobian matrix, its determinant, and cofactors. The equation in new coordinates would become

$$\tilde{L}\tilde{f} = \sum_{i,j=1}^{n} \tilde{a}_{ij} \frac{\partial^2 \tilde{f}}{\partial \varepsilon_i \partial \varepsilon_j} + \sum_{i=1}^{n} \tilde{b}_i \frac{\partial \tilde{f}}{\partial \varepsilon_i} + \tilde{c}\tilde{f} + \tilde{d} \tag{2.21}$$

---

[3] The integration (2.15) for a four-node isoparametrical quadrilateral element has been obtained exactly and will be discussed in detail in chapter three of this report.

Where the tilde denoted that the quantities were functions of computational coordinates.

After the transformation of equation and boundary conditions had been done, the centered difference method was employed to construct the finite difference scheme as follows :

$$\sum_{|i|+|j|+|k|\le 3} C_{i,j,k}(\varepsilon_1, \varepsilon_2, \varepsilon_3)\, g(\varepsilon_1 + i\,\Delta\varepsilon_1, \varepsilon_2 + j\,\Delta\varepsilon_2, \varepsilon_3 + k\,\Delta\varepsilon_3) = R(\varepsilon_1, \varepsilon_2, \varepsilon_3) \quad (2.22)$$

Where the coefficient $c_{i,j,k}$ were called the stencil and was constructed by symbolic manipulation. Taking advantage of the symmetric property, the number to be computed for $c_{i,j,k}$ could be dropped to 10 from 27. The resultant expressions of $c_{i,j,k}$ then could be coded in the FORTRAN language for the next numerical scheme.

2.3 Conclusion

After making a survey of the publications on symbolic and algebraic manipulation, the following conclusions are drawn :

1. None of the papers applying SAM to engineering problems tried to get closed-form solutions. They kept traditional methodology by increasing the terms of the approximation function to get more accurate solutions. This is due to the difficulty in solving generally partial differential equations or integral equations analytically.

2. The papers discussing the application of symbolic and algebraic manipulation on the finite element analysis stop at the step of making a local stiffness matrix, local mass matrix, etc. The same situation also occurred in finite difference analysis. This was because

   (a) The finite element and finite difference methods are themselves approximation methods. The accuracy of results depends on many factors, not just on round-off error or integration error which can be cured by symbolic and algebraic manipulation. Although it was also one of the purposes to improve the accuracy of the solution, the major consideration in applying symbolic and algebraic manipulation was to help in the formulation of the tedious mathematical equations.

   (b) In general engineering problems, the stiffness matrix in FEA and stencil coefficient in FDA are huge in dimension. The limitation of memory space makes the execution of FEA's (or FDA) job impossible by symbolic and algebraic manipulation. Therefore it is necessary to be finished by numerical analysis.

(c) Although most of the SAM systems also possess the capacity of numerical analysis, the execution speed of numerical analysis in symbolic and algebraic manipulator is slower in comparison to that in pure numerical analysis. The difference of efficiencies between them is remarkable when the job is big. Therefore it is best not to have it done completely in symbolic and algebraic manipulation. As the documents showed, nobody did the whole FEA or FDA job in symbolic mode alone.

2.4 References

[1] A. C. Hearn,"REDUCE--A user oriented interactive system for algebraic simplification", interactive system for experimental applied mathematics (edited by M. Klerer and J. Reinfelds), pp 79 -90, academic press, New York, 1968.

[2] Joses Mose,'The fifth year---MACSYMA", ACM, vol. 8, No. 3, pp 105-110, Aug. 1974.

[3] J. A. Van Hulzen,"FORMAC today, or what can happen to an orphan", ACM, vol. 8, No. 1, pp 5-7, 1974.

[4] B. F. Caviness, H. I. Epstein,"A note on the complexity of algebraic differentiation", ACM, vol. 11, No. 3, pp 4-6, 1977.

[5] Steven J. Harrington,"A symbolic limit evaluation program in REDUCE", ACM, vol. 13, No. 1, pp 27-31, 1979.

[6] Edward W. Ng,"Symbolic integration of a class of algebraic functions", ACM, vol. 8, No. 3, pp 99-102, 1974.

[7] John Fitch,"A simple method of taking nth roots of integers", ACM, vol. 8, No. 4, 1974.

[8] A. C. Hearn,"Structure : The key to improve algebraic computation", symbolic and algebraic computation by computers (edited by N. Inada and T. Soma), pp 215-230, 1985.

[9] J. P. Fitch,"Implementing REDUCE on a MICRC-computer", computer algebra (edited by J. A. Van Hulzen), pp 128-136, vol 162, 1983.

[10] A. C. Hearn,"Scientific applications of symbolic computation", computer science and scientific compu. (edited by J. M. Ortega)., pp 83-108, academic press, New York, 1976.

[11] A. Deprit, J. henrard, A. Rom,"Lunar ephemeris : Delaunay's theory revisited", Science 168, pp 1569-1570, 1970.

[12] G. H. Harrison,"A compact method for symbolic computation of Riemann Tensor", J. of Computational Physics, 4, pp 594-600, 1969.

[13] J. A. Campbell, A. C. Hearn,"Symbolic analysis of Feynman Diagrams by computer, J. of Computational Physics, 5, pp 280-327, 1970.

[14] Rudiger Loos,"Amthor's solution of Archimedes' cattle problem", ACM, vol. 11, No.1, pp 4-7, 1977.

[15] K. V. Roberts, J. P. Boris,"The solution of partial differential equations using a symbolic style of Algol", J. of Computational Physics, 8, pp 83-105, 1971.

[16] W. A. Madson, L. B. Smith, N. J. Hoff, "Computer algorithms for solving non-linear problems", J. of Solids Structures, Vol. 1, pp. 113-138, 1965.

[17] I. M. Levi,"Symbolic algebra by computer-applications to structural mechanics", AIAA paper No. 71-363, pp 19-21, April 1971.

[18] Dick J. Wilkins, Jr.,"Applications of a symbolic algebra manipulation language for composite structures analysis, Computer & Structures, Vol. 3, pp 801-807, 1973.

[19] A. K. Noor, C. M. Anderson,"Computerized symbolic manipulation in structural mechanics---progress and potential", Computer & Structures, Vol. 10, pp 95-118, 1979.

[20] A. R. Korncoff, S. J. Fenves,"Symbolic generation of finite element stiffness matrices", Computer & Structure, Vol. 10, pp 95-118, 1979.

[21] S. Steinberg, P. J. Roache,"Symbolic manipulation and Computational fluid dynamics", J. of Computational Physics, 57, pp 251-284, 1985.

[22] S. Bandyopadhyay, J. S. Devitt,"The role of symbolic computation in the management of scientific information", EUROPCAL, pp 460-461, 1985.

[23] P. S. Wang,"Implications of symbolic computation for the teaching of mathematics", ACM, vol. 10, No. 3, pp 15-18, 1976.

[24] R. H. Lance,"Symbolic computation and the instruction of engineering undergraduate", ASME, HTD-vol. 105, AMD-vol. 97, pp 21-24, 1988.

[25] Richard Pavelle," MACSYMA : capacities and applications to problems in engineering and the sciences", Symbolics Inc., Cambridge, MA, 1985.

CHAPTER III

CAPABILITIES OF THE SYMBOLIC AND ALGEBRAIC MANIPULATORS

3.1 Introduction

The capabilities of the symbolic and algebraic manipulator are quite system dependent. Roughly speaking, a system which is designed for general purpose usage usually possesses the functions of differentiation, integration, matrix operation, polynomial manipulation, pattern match, variable substitution and equation solver. Some systems, such as MACSYMA and MATHEMATICA, have a lot of built-in mathematical functions which allow the users to get the answers by just calling the appropriate command once. Others, like REDUCE, may need users to write a short program to get the same answers.

This chapter will demonstrate the fundamental capabilities of the symbolic and algebraic manipulators which are available at hand by solving examples of applied mechanics. Since REDUCE is the oldest system available at The University of Michigan, most examples will be demonstrated by using REDUCE. Of course, MACSYMA will be employed to help the demonstration if it is necessary.

Unfortunately, REDUCE doesn't possess the graphic function, and the version of MACSYMA being used in The University of Michigan also doesn't include the graphics package, although it is available on the market. Therefore, the postprocessing of the results from symbolic and algebraic manipulators will be done by other graphics packages.

3.2 What can the symbolic and algebraic manipulators do ?

In this section, some of the most useful operations in symbolic and algebraic manipulation are demonstrated in detail by examples. They are differentiation, integration, matrix operation, algebraic equation solving, treatment of trigonometric function, differential equation solving, polynomial and rational operation, fortran code output, number system, substitution and built-in functions. The strategies and particular techniques are also mentioned at the place where they are necessary.

3.2.1 Differentiation

All differentiations, without exception, can be done analytically by REDUCE. If it is necessary, REDUCE knows how to apply the chain rule to solve problems. The powerful capabilities of this analytical differentiation will probably replace traditional numerical differentiation in many cases, such as the evaluations of the Jacobian and Hessian matrices However, the wrong results may be obtained by careless or naive users. For instance, in finding the first derivative of x^{x^x} with respect to x. Two different solutions may be obtained as follows :

```
1: on time;
Time: 134 ms

2: df(x**(x**x),x);
  X
 X                2
X   +X*(LOG(X) *X + LOG(X)*X + 1)
-----------------------------------------------
                    X
Time: 383 ms

3: df(x**x**x,x);
   2
  X
X   *X*(2*LOG(X) + 1)
Time: 233 ms
```

Here the first solution is correct. How to judge the correctness of the results is one of the important tasks in symbolic manipulation. A sound background knowledge in the SAM and problem-related fields is very helpful in checking them.

In some cases, the unevaluated differentiation form of functions, such as $\frac{\partial x}{\partial t}$ is desired to be retained throughout the computation. This also can be done as follows :

```
4: depend x,t;
Time: 84 ms

5: depend y,t;
Time: 83 ms

6: p:=a*x*y;
P := A*X*Y
Time: 150 ms

7: df(p,t);
A*(DF(X,T)*Y + DF(Y,T)*X)
Time: 133 ms
```

3.2.2 Integration

In REDUCE, all the integrations performed by the command INT are indefinite integrations with the integration constants discarded. If the function is not integrable by REDUCE (the closed-form solution may exist theoretically), the original form will be displayed on the screen. The definite integration may be obtained by further substituting the upper and lower limits into the results after the indefinite integration.

```
%A non-integrable case.
8: int(sqrt(a^2-x^2),x);

                2    2
INT(SQRT(A - X ),X)
Time: 950 ms

%An integrable case.
9: int(1/(a^2+x^2),x);

        X
 ATAN(---)
        A
---------------
   A
Time: 466 ms
```

In most cases, the non-integrable integrand will become integrable after appropriate manipulation. This pre-treatment involves the technique of changing the integrating variables in fundamental calculus. Sometimes the intelligent users can substantially extend the capabilities of the symbolic and algebraic manipulator by suitably combining human intelligence with the tireless and errorless advantages of computer. For example, if the x in command 8 is substituted by $a*cos(t)$, then $dx=-a*sin(t)*dt$ and the integration of $\sqrt{a^2-x^2}$ with respect to x will become the integration of $-a^2*sin^2(t)dt$ with respect to t, which is integrable by REDUCE. After the integration is done, the original variable x may be substituted back to get the desired expression in terms of x. The check may be done by skeptics by differentiating the resultant expression to get the original integrand. The following three commands demonstrate these procedures.

```
10: int(-a^2*(sin(t))^2,t);

 2
A *(COS(T)*SIN(T) - T)
----------------------
          2
Time: 2134 ms
```

```
11: sub(cos(t)=x/a,sin(t)=sqrt(1-x**2/a**2),t=acos(x/a),ws);

          X    2          2   2
  ACOS(---)*A - SQRT(A  - X )*X
          A
- ---------------------------------------
                   2
Time: 716 ms

12: df(ws,x);

    2   2
   A  - X
------------------
          2   2
 SQRT(A  - X )
Time: 450 ms
```

3.2.3 Matrix operation

Matrix operation is one of the powerful capabilities of symbolic and algebraic manipulators. These include the addition and multiplication of matrices, multiplication of matrices and scalars, inverting matrices, calculating the determinant of a square matrix, finding the trace, computing the eigenvalues and associated eigenvectors exactly if they are available and so forth.

3.2.3.1 Matrix multiplication

The three body rigid rotation 1-2-3 in dynamics is a good example of the utility of matrix multiplication. In robotics, it is necessary to find the analytical form of final orientation from which the rotation angles of each arm can be computed. The final direction cosine matrix d is obtained from the product of three consecutive direction cosine matrices a, b, c.

```
%Declaring four matrices.
13: matrix a(3,3),b(3,3),c(3,3),d(3,3);

%Inputting matrices.
14: a:=mat((1,0,0),(0,cos(q1),-sin(q1)),(0,sin(q1),cos(q1)));
A(1,1) := 1
A(1,2) := 0
A(1,3) := 0
A(2,1) := 0
A(2,2) := COS(Q1)
A(2,3) :=-SIN(Q1)
A(3,1) := 0
A(3,2) := SIN(Q1)
A(3,3) := COS(Q1)
```

```
Time: 700 ms

15: b:=mat((cos(q2),0,sin(q2)),(0,1,0),(-sin(q2),0,cos(q2)))$
Time: 367 ms

16: c:=mat((cos(q3),sin(q3),0),(-sin(q3),cos(q3),0),(0,0,1))$
Time: 383 ms

%multiplication of three matrices.
17: d:=a*b*c;
D(1,1) :=COS(Q2)*COS(Q3)
D(1,2) :=COS(Q2)*SIN(Q3)
D(1,3) :=SIN(Q2)
D(2,1) :=-(COS(Q1)*SIN(Q3)-COS(Q3)*SIN(Q1)*SIN(Q2))
D(2,2) :=COS(Q1)*COS(Q3)+SIN(Q1)*SIN(Q2)*SIN(Q3)
D(2,3) :=-COS(Q2)*SIN(Q1)
D(3,1) :=-(COS(Q1)*COS(Q3)*SIN(Q2)+SIN(Q1)*SIN(Q3))
D(3,2) :=-(COS(Q1)*SIN(Q2)*SIN(Q3)-COS(Q3)*SIN(Q1))
D(3,3) :=COS(Q1)*COS(Q2)
Time: 617 ms
```

The terminators $ in command lines 15 and 16 prohibit the printing of results and save almost half of the time compared to command line 14 which uses the other terminator.

3.2.3.2 Matrix inversion

Unlike numerical analysis in which the time-consuming operation of matrix inversion is to be avoided, to find the inverse of a matrix symbolically is one of the significant and simple tasks in symbolic and algebraic manipulation. One important application of it is in solving a system of linear equations. For example, the problem of finding a curve to fit the given set of data by the least square method results in solving a system of linear equations. The coefficient matrix here is the Hilbert matrix which is usually used to investigate the phenomenon of round-off error accumulation. REDUCE can solve this problem exactly. The numerical solution and symbolic solution are tabulated in Table 3.1, 3.2, 3.3 for three different Hilbert matrix sizes. As the tables show, the numerical solution is not capable of producing accurate results even for the case of the 7*7 Hilbert matrix. The deviation between both solutions is also plotted in Figure 3.1. The significance of symbolic and algebraic manipulation is evident.

```
8: matrix h(40,40),x(40,1)$

9: for i:=1:40 do for j:=1:40 do h(i,j):=1/(i+j-1)$
Time: 30917 ms

10: for i:=1:40 do x(i,1):=i$
Time: 583 ms

11: h:=(1/h);
```

H(1,1) :=1600
H(1,2) :=-1279200
H(1,3) :=340267200
H(1,4) :=-45113759600
H(1,5) :=3573009760320
H(1,6) :=-187583012416800
.
.
H(40,39) :=-1141149866470104951399125616277120066810096080000
H(40,40) := 58520505972825894943544903398826670092825440000
Time: 1355067 ms

12: x:=h*x;
X(1,1) := -64000
X(2,1) := 102272040
X(3,1) := -40781023920
X(4,1) := 7204667408120
X(5,1) := -712815447183840
X(6,1) := 44879235720719400
X(7,1) := -1948531047013671120
X(8,1) := 616382793215847543 60
X(9,1) := -1478390066741437724160
X(10,1) := 277069618742320247043 20
X(11,1) := -4153432059713600183 52000
X(12,1) := 50736054056483096381 80800
X(13,1) := -5126750366823480380 3526400
X(14,1) := 43383194606013382444 2926400
X(15,1) := -31056951342467710632 79900800
.
.
.
.
X(33,1) := -27514898087986919439 2659362117120
X(34,1) := 12902797074572476803 6578024940720
X(35,1) := -49525830202172298308 155472545440
X(36,1) := 15151287095628987186 696245706000
X(37,1) := -35517346849186367154 96119886400
X(38,1) := 59892339471281730744 1549441200
X(39,1) := -64662238360272798660 726511200
X(40,1) := 33563750566547554291 31776400
Time: 96000 ms

x	Exact REDUCE solution	Gauss elimination solution
x(1)	125	124.7899166408321
x(2)	-2880	-2875.997324887610
x(3)	14490	14472.49323423709
x(4)	-24640	-24613.29019994230
x(5)	13230	13216.83350607823

Table 3.1 : Comparison of solution for 5*5 Hilbert matrix

x	Exact REDUCE solution	Gauss elimination solution
x(1)	-216	-204.4675087167038
x(2)	7350	7027.565254454758
x(3)	-57120	-54968.63675793860
x(4)	166320	160780.3224850843
x(5)	-201600	-195532.2818417542
x(6)	85932	83555.64156991533

Table 3.2 : Comparison of solution for 6*6 Hilbert matrix

x	Exact REDUCE solution	Gauss elimination solution
x(1)	343	131.3201346851223
x(2)	-16128	-7941.234641235595
x(3)	177660	100320.8278414759
x(4)	-772800	-476154.4030660979
x(5)	1559250	1020735.514691367
x(6)	-1463616	-1001760.776649569
x(7)	516516	365761.5297697352

Table 3.3 : Comparison of solution for 7*7 Hilbert matrix

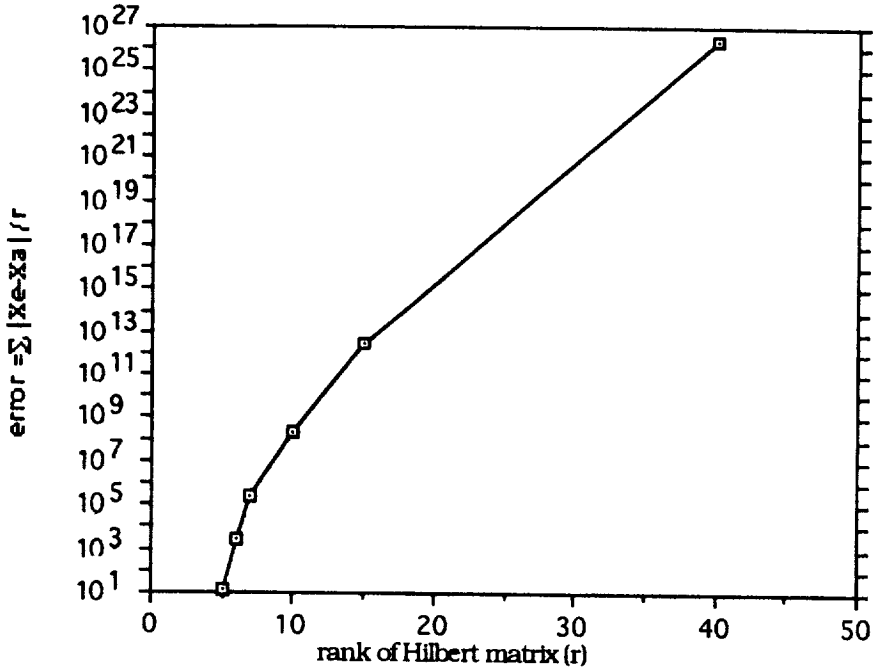

Figure 3.1 : Error between the solutions of REDUCE and Gauss elimination

3.3 Eigenvalues and Eigenvectors

To find the eigenvalues and eigenvectors for a matrix is another important task in the application of symbolic and algebraic manipulation. Since to solve the eigenvalue problems analytically for an arbitrary dimensional matrix is theoretically impossible, the following examples only show the exact solutions for a 3 by 3 matrix.

```
21: matrix s(3,3)$

22: s:=mat((sxx,sxy,sxz),(sxy,syy,syz),(sxz,sxy,szz))$

23: mateigen(s,eta);

      3    2           2          2                                    2
{{ETA -ETA *SXX-ETA *SYY-ETA *SZZ+ETA*SXX*SYY+ETA*SXX*SZZ-ETA*SXY
                                  2
-ETA*SXY*SYZ-ETA*SXZ +ETA*SYY*SZZ+SXX*SXY*SYZ-SXX*SYY*SZZ
     2          2                2
-SXY *SXZ+SXY *SZZ-SXY*SXZ*SYZ+SXZ *SYY,

 1,
```

```
                    ARBCOMPLEX(1)*(ETA*SXZ+SXY*SYZ-SXZ*SYY)
 MAT(1,1):= ------------------------------------------------------------------------
                                  2                                    2
                    ETA -ETA*SXX-ETA*SYY+SXX*SYY-SXY

                    ARBCOMPLEX(1)*(ETA*SYZ-SXX*SYZ+SXY*SXZ)
 MAT(2,1):=-------------------------------------------------------------------------
                                  2                                    2
                    ETA -ETA*SXX-ETA*SYY+SXX*SYY-SXY

  MAT(3,1):=ARBCOMPLEX(1)}}
Time: 1283 ms

24: s:=mat((5,1,0),(1,2,4),(0,4,3))$
Time: 284 ms

25:  mateigen(s,eta);

      3          2
{{ETA -10*ETA +14*ETA+53,

  1,

                 4*ARBCOMPLEX(2)
 MAT(1,1) := ----------------------------
                          2
                    ETA  - 7*ETA + 9

                4*ARBCOMPLEX(2)*(ETA - 5)
 MAT(2,1) := ---------------------------------------
                           2
                     ETA  - 7*ETA + 9

  MAT(3,1) := ARBCOMPLEX(2)}}
Time: 666 ms

26: trace(s);
  SXX + SYY + SZZ
Time: 100 ms
```

As the command lines 23 and 25 show, the solutions from REDUCE by calling MATEIGEN contain three parts. They are

(a) Characteristic equation.

(b) The number of repetition roots. It's one in the above examples.

(c) Eigenvectors.

If the eigenvalues are required, the characteristic equation needs to be solved in addition. The equation solver command SOLVE which will be demonstrated later can meet this requirement. The ARBCOMPLEX(1) which appeared in the eigenvectors is referred to as "arbitrary complex constant". Examples 24 and 25 also show results from REDUCE by substituting numbers into matrix S. The trace of the matrix is also evaluated by the command TRACE as shown in example 26.

3.2.4 Equation solver

REDUCE can solve the polynomial equation up to order three exactly. If the solution includes the imaginary part, the "I" will show up to represent the imaginary symbol. The following example solves the characteristic equation obtained above.

```
27: solve(ETA**3-10*ETA**2+14*ETA+53=0,eta);

                                                                    2/3
{ETA=-((63*SQRT(229)*I-691*SQRT(3))     *SQRT(3)*I+(63*SQRT(229)*I-691*

        2/3                                                 1/3   1/3   1/6       2/3
   SQRT(3))     -20*(63*SQRT(229)*I-691*SQRT(3))     *2    *3     -58*2    *SQRT(3)*

      1/3         2/3   1/3                                                1/3   1/3   1/6
   *3    *I+58*2     *3    )/(6*(63*SQRT(229)*I - 691*SQRT(3))     *2    *3    ),

                                                                   2/3
ETA=((63*SQRT(229)*I - 691*SQRT(3))     *SQRT(3)*I - (63*SQRT(229)*I - 691*

        2/3                                                 1/3   1/3   1/6       2/3
   SQRT(3))     + 20*(63*SQRT(229)*I-691*SQRT(3))     *2    *3     -58*2    *SQRT(3)*

   1/3         2/3   1/3                                                1/3   1/3   1/6
   3   *I - 58*2     *3    )/(6*(63*SQRT(229)*I - 691*SQRT(3))     *2    *3    ),

                                                                   2/3                                  1/3
ETA=((63*SQRT(229)*I - 691*SQRT(3))     + 10*(63*SQRT(229)*I - 691*SQRT(3))

      1/3   1/6       2/3   1/3                                                1/3   1/3   1/6
   *2    *3     + 58*2     *3    )/(3*(63*SQRT(229)*I - 691*SQRT(3))     *2    *3    )}
Time: 5717 ms
```

If the numerical mode NUMVAL, complex switch COMPLEX and FLOAT mode are turned on, the numerical solution of three eigenvalues can be obtained in sixteen digits precision by default. The imaginary parts in the following example are very small and are due to the round-off errors.

```
28: solve(ETA**3 - 10*ETA**2 + 14*ETA + 53=0,eta);
{ETA=4.830759950611553d0 + 2.991124223331416d-7*I,
```

```
 ETA=-(1.61676303063684O8d0 + 6.9600601673474 75d-8*I),
 ETA=6.786003556862447d0 + (-2.295118206596669d-7)*I}
Time: 1567 ms
```

REDUCE can solve systems of linear algebraic equations exactly. The limitation is determined only by the memory capacity of the hardware system. The following example finds the minimization of a quadratic function

*Q=k1*y1^{2}+k2*y1*y2+k3*y2^{2}+k4*y2*y3+k5*y3^{2}-k6*y3* subject to *y1+y2=2*

```
29: Q:=k1*y1**2+k2*y1*y2+k3*y2**2+k4*y2*y3+k5*y3**2-k6*y3+y4*(y1+y2-2)$
Time: 550 ms

30: a:=df(q,y1);
A := 2*K1*Y1 + K2*Y2 + Y4
Time: 167 ms

31: b:=df(q,y2);
B := K2*Y1 + 2*K3*Y2 + K4*Y3 + Y4
Time: 167 ms

32: c:=df(q,y3);
C := K4*Y2 + 2*K5*Y3 - K6
Time: 166 ms

33: d:=df(q,y4);
D := Y1 + Y2 - 2
Time: 150 ms

34: solve({a=0,b=0,c=0,d=0},{y1,y2,y3,y4});

                                                    2
          4*K2*K5 - 8*K3*K5 + 2*K4  - K4*K6
{{Y1= - ----------------------------------------------,
                                    2
          4*K1*K5 - 4*K2*K5 + 4*K3*K5 - K4

           8*K1*K5 - 4*K2*K5 - K4*K6
 Y2=----------------------------------------------,
                                  2
       4*K1*K5 - 4*K2*K5 + 4*K3*K5 - K4

              2*(2*K1*K4 - K1*K6 - K2*K4 + K2*K6 - K3*K6)
 Y3= - ------------------------------------------------------------------,
                                  2
              4*K1*K5 - 4*K2*K5 + 4*K3*K5 - K4

                                    2                    2
            16*K1*K3*K5-4*K1*K4 +2*K1*K4*K6-4*K2 *K5-K2*K4*K6
Y4= - ------------------------------------------------------------------------}}
                                  2
              4*K1*K5 - 4*K2*K5 + 4*K3*K5 - K4
```

Time: 1400 ms

After the substitution of *y1* to *y4* into the quadratic function, the minimum is found as follows :

```
35: q:=q;
                                       2                            2       2
Q:=(16*K1*K3*K5-4*K1*K4 +4*K1*K4*K6 -K1*K6 -4*K2 *K5-2*K2*K4*K6+

        2         2                                 2
  K2*K6 -K3*K6 )/(4*K1*K5-4*K2*K5+4*K3*K5-K4 )
Time: 267 ms
```

3.2.5 Treatment of the trigonometric function

REDUCE doesn't even know an equation as simple as $sin^2(q)+cos^2(q)=1$. However REDUCE does possess the potential to learn it. Due to this powerful capability, the trigonometric functions can be handled easily by just teaching REDUCE the operation rules. For example, without teaching the operation rule of trigonometry, the determinant of direction cosine matrix d in command 17 is as follows :

```
36: det(d);

        2         2         2          2         2         2          2
COS(Q1) *COS(Q2) *COS(Q3) +COS(Q1) *COS(Q2) *SIN(Q3) +COS(Q1) *
        2         2          2         2         2          2         2
COS(Q3) *SIN(Q2) +COS(Q1) *SIN(Q2) *SIN(Q3) +COS(Q2) *COS(Q3) *
        2          2         2         2          2         2         2
SIN(Q1) +COS(Q2) *SIN(Q1) *SIN(Q3) +COS(Q3) *SIN(Q1) *SIN(Q2) +
        2         2         2
SIN(Q1) *SIN(Q2) *SIN(Q3)
Time: 483 ms
```

After teaching REDUCE the appropriate operation rules, the solution becomes quite simple. Note that the time consumption in command 38 is longer than that in command 36. This is due to the extra work needed for simplification. It also reveals the phenomenon of internal swells.

```
37: let cos(q1)**2+sin(q1)**2=1,cos(q2)**2+sin(q2)**2=1,cos(q3)**2+sin(q3)**2=1;
Time: 650 ms

38: det(d);
1
Time: 584 ms
```

3.2.6 Solving differential equation

REDUCE is unable to solve the differential equation directly, while MACSYMA does possess this capability. The following example is a problem of beam deflection $w(x)$ under uniform load q [Figure 3.2]. The differential equation is of the form

$$\frac{d^2w}{dx^2} = s*w + r*x\,(x - L) \qquad (3.1)$$

Where s and r ,in general, are the function of Young's modules, moment of inertial as well as boundary conditions. In the case of small deflection with simple supported on both end, the s becomes zero, and $r=q/2EI$.

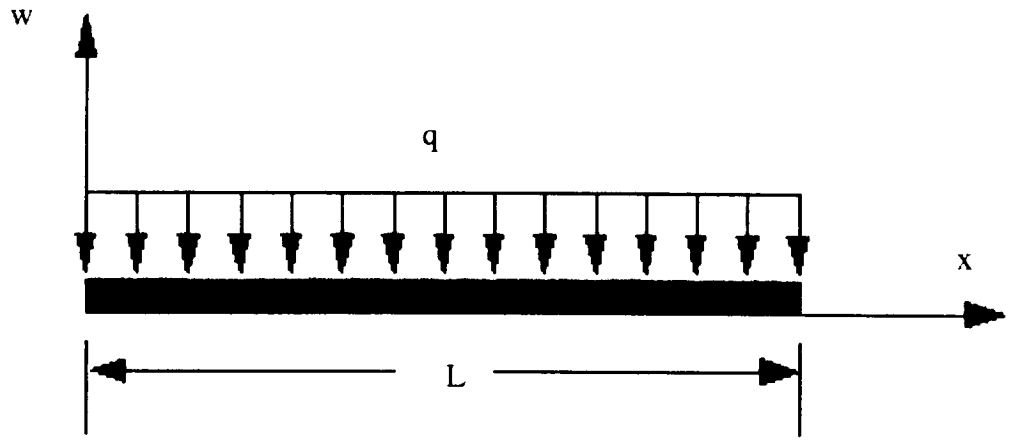

Figure 3.2 : Beam under uniform load. The boundaries are not specified to find general solution.

The (Cn) in the following examples is the MACSYMA prompt for inputting the command and (Dn) is the solution given by MACSYMA.

```
(C1) depends(w,x);
(D1)                          [W(X)]

(C2) diff(w,x,2)-s*w-r*x*(x-l)=0;

                                   2
                                  d W
(D2)          - R X (X - L) + --------- - S W = 0
                                    2
                                  dX

(C3)  ode2(d2,w,x);
Is  S  positive, negative, or zero?
p;
```

$$(D3)\ W = \%K1\ \%E^{SQRT(S)\ X} + \%K2\ \%E^{-SQRT(S)\ X} - \frac{R\ S\ X^2 - L\ R\ S\ X + 2\ R}{S^2}$$

Where the %K1 , %K2 are constants to be determined by boundary conditions. The %E here is the symbol of exponential function.

Although the next example (no specific physical problem associated with it) is more complex, it only takes 3.6 milliseconds in MACSYMA.

(C4) depends(y,x);

(D4) [Y(X)]

(C5) diff(y,x,2)+(2/x)*diff(y,x)-(2/x**2)*y-(1/x**2)*sin(log(x))=0;

$$(D5)\quad \frac{d^2Y}{dX^2} + \frac{2}{X}\frac{dY}{dX} - \frac{2Y}{X^2} - \frac{SIN(LOG(X))}{X^2} = 0$$

(C6) ode2(d2,y,x);

$$(D6)\quad Y = -\frac{3\ SIN(LOG(X)) + COS(LOG(X))}{10} + \%K1\ X + \frac{\%K2}{X^2}$$

(C7) time(d3);

Time:

(D7) [3.6d0]

3.2.7 Polynomial and rational operations

Polynomial and rational operations is one of the most important and useful functions in symbolic and algebraic manipulation. There are two occasions in employing these functions. First, in most cases the problems to be solved are not as simple as the above demonstrations. Therefore it is necessary to manipulate the formulae into a machine manageable forms before calling the appropriate REDUCE commands to solve them. Second, sometimes the solutions are restricted to specific forms for particular usage. In order to get the appropriate forms, there is no way to avoid employing this package. The capabilities of this package include the controls of the expansion, factorization, and cancellation of common factors, determining the GCD of two polynomials, obtaining the part of polynomial and rational functions, and so forth.

```
%Turning off expansion switch.
39: off exp;

%Inputting p and q polynomials. The numerical common factors
%will be automatically factored out.
40:  p:=(x-1)*(5*x-3)**2*(4*x+8)^3*(9*x-6);

                      2                    3
P := 192*(5*X - 3) *(3*X - 2)*(X + 2) *(X - 1)

41:  q:=(x-1)*(5*x-3)*(4*x+8)*(x+4);
Q := 4*(5*X - 3)*(X + 4)*(X + 2)*(X - 1)

%Getting greatest common divider of p and q.
42: gcd(p,q);
4*(5*X - 3)*(X + 2)*(X - 1)

%Turning on expansion switch and checking p, q.
43: on exp;

44: p;

            7        6        5        4        3        2
192*(75*X +235*X -163*X -723*X +392*X +664*X -624*X+144)

45: q;
       4        3       2
4*(5*X  + 22*X  - 5*X  - 46*X + 24)

%Defining a fraction r.
46: r:=p/q;

                7        6        5        4        3        2
R :=(48*(75*X +235*X -163*X -723*X +392*X +664*X -624*X+144))/

          4       3      2
       (5*X +22*X -5*X -46*X+24)

%Turning on the greatest common divider switch and rechecking r.
%The common factors have been cancelled as shown in command 47.
47: on gcd;

48: r;

          4        3        2
48*(15*X  + 41*X  - 10*X  - 52*X + 24)
-----------------------------------------------------
                    X + 4

%Getting the denominator and numerator of fraction r.
49: den(r);
X + 4
```

```
50: num(r);

                4        3       2
48*(15*X  + 41*X  - 10*X  - 52*X + 24)
```

%Getting the leading degree of numerator of fraction r.

```
51: deg(num(r),x);
4
```

%The numerator of r in command 49 also can be factored as the multiplications of each factors.

```
52: on ifactor;

53: factorize(num(r));
{2,2,2,2,3,X + 2,X + 2,3*X - 2,5*X - 3}
```

3.2.8 Fortran code output

REDUCE can automatically produce the fortran expression, natural style expression (default), and REDUCE code. The fortran code can be made as a subroutine and be directly input to the fortran main program. The natural style expression allows it to be looked as hand-written form, while REDUCE code is useful in making a REDUCE subroutine for input into the REDUCE main program. Since the results from REDUCE are generally very lengthy, the functions of code-conversion make the switch from symbolic and algebraic manipulation to numerical analysis smoother. This not only saves effort in symbolic and algebraic manipulation, but also rules out all the possibilities of error introduced by hand typing. The following examples will give a clearer understanding about these functions.

%Inputting the polynomial. The output forms are in natural style of human being writing.

```
54: p:=(a+b-c)**7;

     7    6       6         5  2      5             5  2       4  3        4  2
P := A +7*A *B-7*A *C+21*A *B -42*A *B*C+21*A *C +35*A *B  -105*A *B *C+105*

      4      2     4  3       3  4       3  3           3  2  2       3      3       3  4
     A *B*C -35*A *C +35*A *B -140*A *B *C +210*A *B *C -140*A *B*C +35*A *C

         2  5       2  4           2  3  2       2  2  3       2     4     2  5
    +21*A *B -105*A *B *C +210*A *B *C -210*A *B *C +105*A *B*C -21*A *C +7*

       6        5              4  2        3  3           2  4              5       6
    A*B -42*A*B *C+105*A*B *C -140*A*B *C +105*A*B *C -42*A*B*C +7*A*C

      7    6         5  2     4  3     3  4     2  5        6  7
    +B -7*B *C+21*B *C -35*B *C +35*B *C -21*B *C +7*B*C -C
Time: 1434 ms
```

%Turning on the fortran-code conversion switch.

```
55: on fort;
```

```
%Checking the output of fortran code.
56: p;
     ANS=A**7+7.*A**6*B-7.*A**6*C+21.*A**5*B**2-42.*A**5*B
    .*C+ 21.*A**5*C**2+35.*A**4*B**3-105.*A**4*B**2*C+105.
    .*A**4*B*C**2-35.*A**4*C**3+35.*A**3*B**4-140.*A**3*B
    .**3*C+210.*A**3*B**2*C**2-140.*A**3*B*C**3+35.*A**3
    .*C**4+21.*A**2*B**5-105.*A**2*B**4*C+210.*A**2*B**3*C
    .**2-210.*A**2*B**2*C**3+105.*A**2*B*C**4-21.*A**2*C
    . **5+7.*A*B**6-42.*A*B**5*C+105.*A*B**4*C**2-140.*A*B
    . **3*C**3+105.*A*B**2*C**4-42.*A*B*C**5+7.*A*C**6+B**
    . 7-7.*B**6*C+21.*B**5*C**2-35.*B**4*C**3+35.*B**3*C**
    . 4-21.*B**2*C**5+7.*B*C**6-C**7
Time: 850 ms

%Changing the number of continuation line in fortran code.
57: cardno!*:=10$

58: p;
     ANS1=7.*A*B**6-42.*A*B**5*C+105.*A*B**4*C**2-140.*A*B
    . **3*C**3+105.*A*B**2*C**4-42.*A*B*C**5+7.*A*C**6+B**
    . 7-7.*B**6*C+21.*B**5*C**2-35.*B**4*C**3+35.*B**3*C**
    . 4-21.*B**2*C**5+7.*B*C**6-C**7
     ANS=A**7+7.*A**6*B-7.*A**6*C+21.*A**5*B**2-42.*A**5*B
    . *C+21.*A**5*C**2+35.*A**4*B**3-105.*A**4*B**2*C+105.
    . *A**4*B*C**2-35.*A**4*C**3+35.*A**3*B**4-140.*A**3*B
    . **3*C+210.*A**3*B**2*C**2-140.*A**3*B*C**3+35.*A**3*
    . C**4+21.*A**2*B**5-105.*A**2*B**4*C+210.*A**2*B**3*C
    . **2-210.*A**2*B**2*C**3+105.*A**2*B*C**4-21.*A**2*C
    . **5+ANS1
Time: 1034 ms

%Turning off fortran-code conversion switch.
59: off fort;

%Turning off the 'natural style' function switch.
60: off nat;

%Checking the output of REDUCE code.
61: p:=p;
P := A**7+7*A**6*B-7*A**
6*C+21*A**5*B**2-42*A**5*B*C+21*A**5*C**2+35*A**4*B**3-105*
A**4*B**2*C+105*A**4*B*C**2-35*A**4*C**3+35*A**3*B**4-140*A
**3*B**3*C+210*A**3*B**2*C**2-140*A**3*B*C**3+35*A**3*C**4+
21*A**2*B**5-105*A**2*B**4*C+210*A**2*B**3*C**2-210*A**2*B
**2*C**3+105*A**2*B*C**4-21*A**2*C**5+7*A*B**6-42*A*B**5*C
+105*A*B**4*C**2-140*A*B**3*C**3+105*A*B**2*C**4-42*A*B*C**5
+7*A*C**6+B**7-7*B**6*C+21*B**5*C**2-35*B**4*C**3+35*B**3*C
**4-21*B**2*C**5+7*B*C**6-C**7$
Time: 816 ms
```

3.2.9 Number system

In addition to symbolic manipulation, the symbolic and algebraic manipulators can also do numerical analysis. There are three ways to treat numbers in REDUCE. They are

(a) Integer --- In general, there is no practical limit on the number of digits. For example, the value of 2^{1000} gives 255 digits. It only takes 383 milliseconds.

```
62: A:=2**1000;
A:=1071508607186267320948425049060001810561404811705533607443750388370
   3510511249361224931983788156951275946729175531468251871452856923140
   4043598457759857480393456777482423098542107460506237114187718215304
   6474983581941267398767559165543946077062914571 1
Time: 383 ms
```

(b) Fraction number --- Numbers that aren't integers and operated with symbols (or numbers) are represented by default by the quotient of two integers with message(s) telling users the conversion.

```
63:  a:=0.999*b*c;
*** 0.999 represented by 999/1000
       999*B*C
A := ------------
        1000
Time: 200 ms

64: a:=0.999*0.5;
*** 0.999 represented by 999/1000
*** 0.5 represented by 1/2
       999
A := --------
      2000
Time: 217 ms
```

(c) Real number --- It is also possible to ask REDUCE to work the floating point approximations to numbers with arbitrary precision with specified numbers of digit.

```
%Turning on the numerical mode.
65: on numval;

%Turning on the floating system switch.
66: on float;

67: pi;
3.141592653589793d0
Time: 150 ms

68: on bigfloat;
*** Domain mode FLOAT changed to BIGFLOAT
```

```
% Specified 50 digits.
69: precision 50$

70: pi;
3.141 59265 35897 93238 46264 33832 79502 88419 71693 993751
Time: 217 ms
```

3.2.10 Substitution

There are two substitution functions in REDUCE. One is for local substitution, and the other is for global substitution. The difference can be revealed in the following examples.

```
%Defining a function f.
71: f:=6.*A**2*R1**2*U2-3.*ATAN(U1/A)*A**2*
    R1*R2*U1-9.*ATAN(U1/A)*A**2*R1*R2*U2+3.*ATAN(U1/A)*A
    **2*R2**2*U1+3.*ATAN(U1/A)*A**2*R2**2*U2-2.*ATAN(U1/
    A)*R1**2*U1**3+6.*ATAN(U1/A)*R1**2*U1**2*U2$
Time: 1234 ms

%making a local substitution and calling it as B.
72:  b:=sub(atan(u1/a)=k,f);

          2                        2                    2       2       2
B :=-(3*A *K*R1*R2*U1+9*A *K*R1*R2*U2-3*A *K*R2 *U1-3*A *K*

       2          2  2               2  3          2  2
     R2 *U2 - 6*A *R1 *U2 + 2*K*R1 *U1  - 6*K*R1 *U1 *U2)
Time: 433 ms

%Checking the original function f after the local substitution. It's unchanged.

73: f;

            U1   2                        U1   2                        U1   2  2
-(3*ATAN(----)*A *R1*R2*U1+9*ATAN(----)*A *R1*R2*U2-3*ATAN(----)*A *R2 *U1
             A                             A                             A

            U1   2  2                U1    2  3            U1    2  2
 -3*ATAN(----)*A *R2 *U2+2*ATAN(----)*R1 *U1 - 6*ATAN(----)*R1 *U1 *U2
             A                        A                    A

       2  2
   - 6*A *R1 *U2)
Time: 367 ms

%Making global substitution.
74: let atan(u1/a)=g;
Time: 133 ms

%The original function f has been changed.
75: f;
```

```
      2               2                    2      2        2       2
-(3*A *G*R1*R2*U1+9*A *G*R1*R2*U2-3*A *G*R2 *U1-3*A *G*R2 *U2

     2   2              2   3          2   2
  - 6*A *R1 *U2 + 2*G*R1 *U1  - 6*G*R1 *U1 *U2)
Time: 283 ms
```

3.2.11 Built-in functions

The built-in functions are quite system dependent. Since REDUCE is designed for general purpose usage, there are not many built-in functions. However they can be obtained by suitably combined commands of REDUCE. On the contrary, MACSYMA has many built-in functions which allow users to simply call commands once to get the solution. Some of these MACSYMA functions are shown in the following paragraphs.

(a) Limit evaluation --- If it is necessary, the function LIMIT in MACSYMA will automatically apply L'Hospital's rule to evaluate the formulae.

```
(C8)limit(sin(x)/x,x,0,plus);
(D8)                    1

(C9) time(d8);
Time:
(D9)                [2.116d0]

(C10) limit((6*x*2+3*x-4)/(x-1),x,1,plus);
(D10)                     INF

(C11) limit((1-x)**(1/x),x,0);
                           -1
(D11)                   %E

(C12) time(d11);
Time:
(D12)               [2.75d0]
```

(b) Laplace transformation --- The LAPLACE command in MACSYMA can transform the functions in physical domain, such as EXP, LOG, SIN, COS, SINH, COSH, DELTA and ERF, into the *s* domain. In addition, it also can transform a differential equation into algebraic equation. The command for inverse of Laplace transform is also available.

```
(C13) laplace(1/sqrt(t),t,s);
                    SQRT(%PI)
(D13)               ---------------
                    SQRT(S)
```

```
(C14) time(d13);
Time:
(D14)                    [0.05d0]

(C15)laplace(((c-b)*exp(a*t)+(a-c)*exp(b*t)+(b-c)*exp(c*t))/((a-b)*(b-c)*(c-a)),t,s);

               B - C     A - C     C - B
               -------- + -------- + -------
               S - C     S - B     S - A
(D15)        ----------------------------------------
                    (A - B) (B - C) (C - A)

(C16) time(d9);
Time:
(D16)                    [0.384d0]

(C17) laplace(sin(a*t)-a*t*cos(a*t),t,s);

                                 2
               A            2 S            1
(D17)        --------- - A (------------ - ----------)
              2   2          2     2 2    2   2
             S  + A        (S  + A )     S  + A

(C18) time(d17);
Time:
(D18)                    [0.233d0]

%Inverting the Laplace transform.
(C19) ilt((s+6)/(s^2+4*s+12),s,t);

                -2T     2 SIN(2 SQRT(2) T)
(D19)       %E      (------------------------ + COS(2 SQRT(2) T))
                           SQRT(2)
(C20) time(d4);
Time:
(D20)                           [0.933d0]

%Transforming a differential equation into algebraic equation.
(C21) laplace(diff(y(x),x,2)-3*diff(y(x),x)+2*y(x)=0,x,s);

                         !
          d              !
(D21) - --- (Y(X))  !   - 3 (S LAPLACE(Y(X), X, S) - Y(0))
         dX              !
                         !X = 0

       2
   +S  LAPLACE(Y(X),X,S)+2 LAPLACE(Y(X),X,S)-Y(0) S=0

(C22) time(d21);
Time:
```

```
(D22)                          [0.15d0]

(C23) laplace(diff(y(x),x,2)+w^2*y(x)-b*sin(w*x)=0,x,s);

                          !
       d                  !     2                          2
(D23)- ---(Y(X))          ! +W  LAPLACE(Y(X),X,S)+S  LAPLACE(Y(X),X,S)
       dX                 !
                          !X=0

                    B W
                - --------- - Y(0) S = 0
                    2   2
                   W  + S

(C24) time(d23);
Time:
(D24)                                [0.217d0]
```

(c) Series expansion --- The MACSYMA version in University of Michigan provides the Taylor series and power series expansion capabilities. Although the function of Fourier series expansion is available on the market, it is not available here.

```
(C25) taylor(%e^x,[x,0,7]);

                                 2   3   4   5   6   7
                                X   X   X   X   X   X
(D25)/T/     1 + X + -- + --- + --- + ---- + ------ + ------- + . . .
                     2   6   24  120 720  5040

(C26) time(d25);
Time:
(D26)                          [0.567d0]

(C27) powerseries(%e^x,x,0);
                          INF
                          ====     I1
                          \       X
(D27)                      >      -----
                          /       I1!
                          ====
                          I1 = 0

(C28) time(d27);
Time:
(D29)                          [0.45d0]
```

3.3 References

[1] A. C. Hearn,"REDUCE users' manual", version 3.3, The RAND Co., Santa Monica, CA 90406-2138, July 1987.

[2] "MACSYMA reference manual", The mathlab group laboratory for computer science, MIT, version 10, January 1983.

[3] Stephen Wolfram,"Mathematica- A system for doing mathematics by computer", Addison-Wesley Publishing Compan, Inc.,1988.

[4] Peter V. O'Neil,"Advanced Engineering mathematics", Wadsworth Publishing Company, 1983.

[5] Richard L. Burden, J. Douglas Faires,"Numerical Analysis", Prindle, Weber & Schmidt publishers, Third edition, 1985.

[6] Gilbert Strang,"Introduction to applied mathematics", wellesley-Cambridge press, 1986.

[7] Thomas R. Kane, Peter W. Likins and David A. Levinson,"Spacecraft dynamics", MacGraw-Hill book Company, 1983.

[8] Richard Pavelle," MACSYMA : capacities and applications to problems in engineering and the sciences", Symbolics Inc., Cambridge, MA, 1985.

[9] Al Shenk,"Calculus and Analytic Geometry", Scott, Foresman and Company, 1984.

[10] Murry R. Spiegel,"mathematical Handbook of formulas and tables", Schaum's outline series in mathematics, McGraw-Hill Book Company, 1968.

CHAPTER IV

APPLICATION OF SYMBOLIC AND ALGEBRAIC MANIPULATION TO AUTOMATIC PROBLEM FORMULATION

4.1 Introduction

One of the most important advantages gained from symbolic and algebraic manipulation is to automatically formulate lengthy mathematical equations without making any errors. Modern scientists and engineers are continually being challenged with more and more complicated formulas. With the aid of symbolic and algebraic manipulators, most problems can be treated easily and correctly. This chapter will demonstrate six automatic formulation examples done by symbolic and algebraic manipulation. They are :

1. The derivation of equations of motion in dynamics.
2. Tensor formulation for the shell problem.
3. The approximation to a function by Fourier series.
4. The formulation template for the iteration method in nonlinear numerical analysis.
5. Finite element stiffness matrix and mass matrix construction for 6-node triangular element in a heat transfer problem.
6. Finite element stiffness matrix construction for 4-node isoparametrically quadrilateral element in a plane elasticity problem.

Of course, the use of symbolic and algebraic manipulators as tools to automatically formulate mathematical equations can be extended to any fields. Although the methodologies are dependent on the problem to be solved, the basic commands used in programming are similar.

4.2 Derivation of equation of motion by SAM

4.2.1 Introduction

The derivation of equations of motion by the Lagrange method for the system shown in Figure 4.1 involves finding kinetic energy T, potential energy V, and therefore the Lagrangian L.

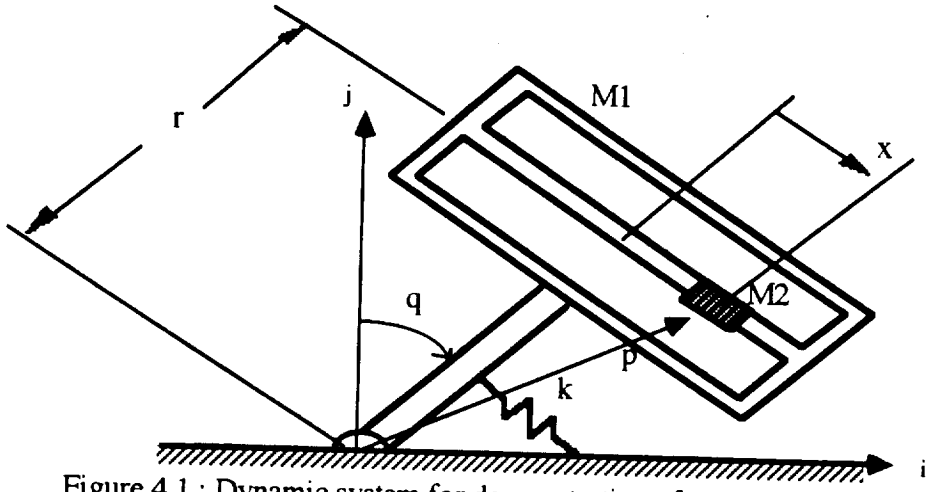

Figure 4.1 : Dynamic system for demonstration of symbolic and algebraic manipulation

The Lagrangian then is partially differentiated with respect to both generalized coordinates and the rate of generalized coordinates. The equations of motion will be obtained after taking the time derivatives to appropriate terms and assembling the necessary terms. Mathematically, the Lagrange equations are expressed in the form of

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = 0 \qquad (4.1)$$

where L is the Lagrangian and is defined as the difference between kinetic energy T and potential energy V.

$$T = \tfrac{1}{2}M_2(\dot{x} + r\dot{\theta})^2 + \tfrac{1}{2}M_1(r\dot{\theta})^2 + \tfrac{1}{2}I\dot{\theta}^2 \qquad (4.2)$$

$$V = -M_1 gr(1 - \cos\theta) - M_2 g(r - r\cos\theta + x\sin\theta) + \tfrac{1}{2}k(r\theta)^2 \qquad (4.3)$$

4.2.2 REDUCE program and solution

1: on time;
Time: 83 ms

```
%declaring theta and x as a function of time.
2: depend theta,time;
Time: 67 ms

3: depend x,time;
Time: 50 ms

%Calculating the total kinetic energy of system.
%The velocity v of body 2 need be evaluated later.
4: te:=(1/2)*i0*(df(theta,time))**2+(1/2)*m1*r**2*(df(theta,time))**2+(1/2)*m2*v**2;

                    2                      2  2      2
       DF(THETA,TIME) *I0 + DF(THETA,TIME) *R *M1 + M2*V
TE := ---------------------------------------------------
                                2
Time: 500 ms

%Calculating the position vector of body 2.
5: p:=(r*cos(theta)-x*sin(theta))*j+(r*sin(theta)+x*cos(theta))*i;
P := COS(THETA)*I*X+COS(THETA)*J*R+SIN(THETA)*I*R-SIN(THETA)*J*X
Time: 333 ms

%The velocity vectors is then obtained by taking the derivative of
%the position vector with respect to time.
6: dp:=df(p,time);
DP := COS(THETA)*DF(THETA,TIME)*I*R-COS(THETA)* DF(THETA,
      TIME)*J*X+COS(THETA)*DF(X,TIME)*I-DF(THETA,TIME)*
      SIN(THETA) *I*X-DF(THETA,TIME)*SIN(THETA)*J*R-DF(X,TIME)
      *SIN(THETA)*J
Time: 234 ms

%Getting the magnitude square of velocity of body 2.
7: v**2:=lcof(dp,i)**2+lcof(dp,j)**2;

  2                    2                    2  2                    2
V :=COS(THETA) *DF(THETA,TIME) *R +COS(THETA) *DF(THETA,TIME)

    2                    2                                          2
  *X +2*COS(THETA) *DF(THETA,TIME)*DF(X,TIME)*R+COS(THETA) *

      2                    2                    2  2                    2
    DF(X,TIME) +DF(THETA,TIME) *SIN(THETA) *R +DF(THETA,TIME) *

      2  2                                                    2
    SIN(THETA) *X +2*DF(THETA,TIME)*DF(X,TIME)*SIN(THETA) *R+
                  2                    2
    DF(X,TIME) *SIN(THETA)
Time: 650 ms

%Teaching REDUCE the trigonometric rule for simplification of %formulae.
8: let (cos(theta))**2+(sin(theta))**2=1;
Time: 150 ms
```

```
%Inputting the potential energy of system.
9: ve:=-m1*g*r*(1-cos(theta))-m2*g*(r*(1-cos(theta))+x*sin(theta))+(1/2)*k*(r*theta)**2;
VE := (2*COS(THETA)*G*R*M1+2*COS(THETA)*G*R*M2-2*SIN(THETA)
                                                   2         2
     *G*M2*X- 2*G*R*M1 - 2*G*R*M2 + K*R *THETA )/2
Time: 483 ms

%calculating the Lagrangian.
10: la:=te-ve;

LA := -(2*COS(THETA)*G*R*M1+2*COS(THETA)*G*R*M2-DF(THETA,
          2                        2  2                         2  2
     TIME) *I0 - DF(THETA,TIME) *R *M1 - DF(THETA,TIME) *R
                                2      2
     *M2 - DF(THETA,TIME) *M2*X -2*DF(THETA,TIME)*DF(X,TIME)
                          2
     *R*M2 - DF(X,TIME) *M2 -2*SIN(THETA)*G*M2*X-2*G*R*M1-
                     2       2
     2*G*R*M2+K*R *THETA )/2
Time: 617 ms

%Deriving the equation of motion for theta coordinate without
%any simplification.

11: e1:=df(df(la,df(theta,time)),time)-df(la,theta);

E1 := -(COS(THETA)*G*M2*X+DF(THETA,THETA,TIME)*DF(THETA,
                                                                2
     TIME) *I0+DF(THETA,THETA,TIME)*DF(THETA,TIME)*R *M1
                                                         2
     +DF(THETA,THETA,TIME)*DF(THETA,TIME)*R *M2 +DF(THETA,
                                                   2
     THETA,TIME)*DF(THETA,TIME)*M2*X +DF(THETA,THETA,TIME)
                                                                2
     *DF(X,TIME)*R*M2-DF(THETA,TIME,2)*I0-DF(THETA,TIME,2)*R
                                 2                            2
     *M1-DF(THETA,TIME,2)*R *M2-DF(THETA,TIME,2)*M2*X -2*
     DF(THETA,TIME)*DF(X,TIME)*M2*X-DF(X,TIME,2)*R*M2 +
                                                         2
     SIN(THETA) *G*R*M1+SIN(THETA)*G*R*M2-K*R *THETA)
Time: 783 ms

%Deriving the equation of motion for x coordinate without simplification.
12: e2:=df(df(la,df(x,time)),time)-df(la,x)+f;
                                                   2
E2 := DF(THETA,TIME,2)*R*M2-DF(THETA,TIME) *M2*X-DF(THETA,
     TIME) *DF(X,TIME,X)*R*M2-DF(X,TIME,X)*DF(X,TIME)*M2 +
     DF(X,TIME,2)*M2 - SIN(THETA)*G*M2+F
Time: 483 ms

%Teaching REDUCE the 2nd derivative of theta w/t theta and time.is null, and so does x.
```

```
13: let df(theta,time,theta)=0,df(x,x,time)=0,df(theta,theta,time)=0,df(x,time,x)=0;
Time: 350 ms

%turning off the automatic expansion switch.
14: off exp;
Time: 50 ms

%==============================================
%Starting polynomial manipulation to simplify the equations of motion.
%==============================================

%Getting the coefficient of angular acceleration term.
15: a1:=lcof(e1,df(theta,time,2));

                        2              2
A1 := (M1 + M2)*R + I0 + M2*X
Time: 617 ms

%Getting the coefficient of sin(theta).
16: a2:=lcof(e1,sin(theta));
A2 :=  - (M1 + M2)*G*R
Time: 550 ms

%Getting the leftover after taking off the above two terms.
17: a3:=e1-a1*df(theta,time,2)-a2*sin(theta);
A3 :=-(COS(THETA)*G*M2*X-2*DF(THETA,TIME)*DF(X,TIME)*M2*X
                                  2
     -DF(X,TIME,2)*R*M2-K*R *THETA)
Time: 533 ms

%Rearranging the equation of motion for theta coordinate.
%The results are simpler than those of in command 11.
18: e1:=a1*df(theta,time,2)+a2*sin(theta)+a3;
                    2              2
E1 := ((M1+M2)*R +I0+M2*X )*DF(THETA,TIME,2)-(COS(THETA)*G*M2*
                                                          2
     X-2*DF(THETA,TIME)*DF(X,TIME)*M2*X-DF(X,TIME,2)*R*M2-K*R
      *THETA)-(M1+M2)*SIN(THETA)*G*R
Time: 400 ms

%Getting the coefficient of x acceleration term.
19: c1:=lcof(e2,df(x,time,2));
C1 := M2
Time: 350 ms

%Getting the coefficient of angular acceleration term.
20: c2:=lcof(e2,df(theta,time,2));
C2 := R*M2
Time: 333 ms

%Getting the leftover terms by removing the above two terms.
21: c3:=e2-c1*df(x,time,2)-c2*df(theta,time,2);
                                2
C3 :=  - (DF(THETA,TIME) *M2*X + SIN(THETA)*G*M2 - F)
```

```
Time: 350 ms

%equation of motion for x coordinate.
22: e2:=c1*df(x,time,2)+c2*df(theta,time,2)+c3;
                                2
E2 := -((DF(THETA,TIME) *M2*X+SIN(THETA)*G*M2-F)-
       DF(THETA,TIME,2)*R*M2 - DF(X,TIME,2)*M2)
Time: 317 ms

23: bye;
```

The examples shown above are the demonstration of obtaining the left hand side formulae of equation (4.1). The equations of motion of system can be simply done by setting the results of command 18 and 22 equal to zero. As the system is complex, the analytical derivations of equations of motion by hand will become tedious and prone to error. For the cases of complex systems, The application of symbolic and algebraic manipulation will be more significant.

4.3 Automatic tensor formulation for shell problem

4.3.1 Preliminary formulation

Tensors are convenient mathematical entities for concisely describing physical situations, that are independent of coordinate transformations. Although the benefits gained by employing tensor notation are quite significant in the related fields, the expressions of tensor formula often rise to errors. Fortunately, this difficulty can be avoided by the use of symbolic and algebraic manipulation. This advantage will be demonstrated by formulating the thin shell problem in tensor form and using SAM to expand the resulting tensor equations. For the sake of convenience, the Latin indices will be referred for the range 1, 2, 3 and the Greek indices are in the range of 1, 2 in the following paragraph without special stress.

On the formulation of the thin shell problem, the only necessary inputs are three parametric equations $f^i(u^1,u^2)$ of the shell middle surface. Based on parametric equations, the covariant metric tensor a_{ab} and its determinant are calculated

$$a_{\alpha\beta} = f^i_{,\alpha} f^i_{,\beta} = \frac{\partial f^i}{\partial u^\alpha} \cdot \frac{\partial f^i}{\partial u^\beta} \tag{4.4}$$

$$a = \det(a_{\alpha\beta}) \tag{4.5}$$

The contravariant metric tensor is therefore given by

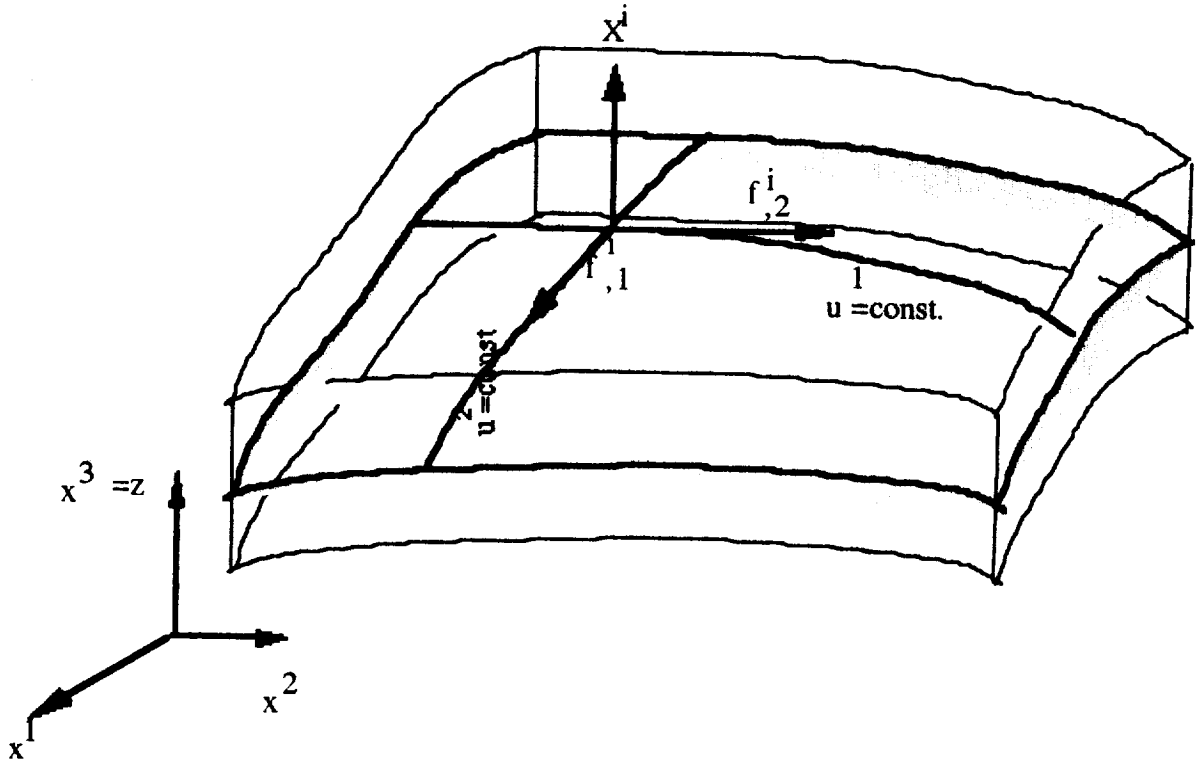

Figure 4.2 : Pictorically vector notations of shell

$$a^{\alpha\beta} = cofactor\ (a_{\alpha\beta})/a \tag{4.6}$$

The first and second kind of Christoffel symbols for the surface are defined as follows respectively :

$$[\alpha\beta,\gamma] = \tfrac{1}{2}(a_{\alpha\gamma,\beta} + a_{\beta\gamma,\alpha} - a_{\alpha\beta,\gamma}) \tag{4.7}$$

$$\left\{ \begin{matrix} \alpha \\ \beta\ \gamma \end{matrix} \right\} = \tfrac{1}{2}a^{\alpha\delta}(a_{\beta\delta,\gamma} + a_{\delta\gamma,\beta} - a_{\beta\gamma,\delta}) \tag{4.8}$$

The covariant differentiations of a covariant vector (the 1st order tensor) and the 2nd order tensor are therefore calculated

$$A_{\beta,\alpha} = D_\alpha A_\beta = A_{\beta,\alpha} - \left\{ \begin{matrix} \gamma \\ \beta\ \alpha \end{matrix} \right\} A_\gamma \tag{4.9}$$

$$A_{\beta\gamma,\alpha} = D_\alpha A_{\beta\gamma} = A_{\beta\gamma,\alpha} - \left\{ \begin{matrix} \delta \\ \beta\ \alpha \end{matrix} \right\} A_{\delta\gamma} - \left\{ \begin{matrix} \delta \\ \gamma\ \alpha \end{matrix} \right\} A_{\beta\delta} \tag{4.10}$$

When the above formulations are done, the curvature tensor and its determinant are then computed by the formulae of

$$d_{\alpha\beta} = X^i f^i_{\alpha\beta} \tag{4.11}$$

$$d = \det(d_{\alpha\beta}) \tag{4.12}$$

Where the X^i are the unit normal vectors of the shell middle surface [see Figure 4.2] and are equal to

$$X^i = a^{-1/2} e_{ijk} f^j_{,1} f^k_{,2} \tag{4.13}$$

Where e_{ijk} is the generalized Kronecker delta.

The strain tensor is defined as half of the difference between the deformed metric tensor and undeformed metric tensor.

$$E_{\alpha\beta} = \tfrac{1}{2}(a^*_{\alpha\beta} - a_{\alpha\beta}) \tag{4.14}$$

Where the asterisk superscript is denoted as deformed state. The $a^*_{\alpha\beta}$ is defined as

$$a^*_{\alpha\beta} = a_{\alpha\beta} + p_{\alpha\beta} + p_{\beta\alpha} + p^\gamma_\alpha p_{\beta\gamma} + q_\alpha q_\beta \tag{4.15}$$

The generalized two-dimensional displacement gradient p, and rotation q are represented as

$$p_{\alpha\beta} = D_\alpha v_\beta - d_{\alpha\beta} w \tag{4.16}$$

$$q_\alpha = d_{\alpha\beta} v^\beta + w_{,\alpha} \tag{4.17}$$

Where the v_α are the in-plane displacements and w is the out of plane displacement of the middle surface of the shell. Similarly the bending tensor is equal to the difference between the deformed curvature tensor and undeformed curvature tensor.

$$K_{\alpha\beta} = d^*_{\alpha\beta} - d_{\alpha\beta} \tag{4.18}$$

The deformed curvature tensor $d^*_{\alpha\beta}$ is expressed as

$$d^*_{\alpha\beta} = \left(\frac{a}{a^*}\right)^{1/2}[(1 + p^\eta_\eta + \vartheta/a)(d_{\alpha\beta} + D_\beta q_\alpha + d^\gamma_\beta p_{\alpha\gamma}) - (q^\rho + \varepsilon^{\rho\eta}\varepsilon^{\gamma\delta} q_\gamma p_{\delta\eta})(D_\beta p_{\alpha\rho} - d_{\beta\rho} q_\alpha)]$$

Where $\vartheta = \det(p_{\alpha\beta})$

The constitutive equations for isotropically elastic and thin shell are

$$N^{\alpha\beta} = \frac{Eh}{1-\nu^2}[(1-\nu)E^{\alpha\beta} + \nu a^{\alpha\beta} E^{\gamma}_{\gamma}] \tag{4.19}$$

$$M_{\alpha\beta} = \frac{Eh^3}{12(1-\nu^2)}[(1-\nu)K^{\alpha\beta} + \nu a^{\alpha\beta} K^{\gamma}_{\gamma}] \tag{4.20}$$

Where the Young's modules E and strain tensor $E^{\alpha\beta}$ shouldn't be confused here.The final equilibrium equations are then

$$D_\alpha N^{\alpha\beta} + 2d^{\beta}_{\gamma} D_\alpha M^{\gamma\alpha} + M^{\gamma\alpha} D_\alpha d^{\beta}_{\gamma} + F^{\beta} = 0 \tag{4.21}$$

$$D_\alpha D_\beta M^{\alpha\beta} - d_{\alpha\gamma} d^{\gamma}_{\beta} M^{\alpha\beta} - d_{\alpha\beta} N^{\alpha\beta} - P = 0 \tag{4.22}$$

Where the F^{β} and P are the external forces applied in the in-plane and out-plane directions, respectively.

4.3.2 REDUCE program and resultant expressions

The REDUCE program shown below is based on the above formulation methodology. The explanations of the program are also included to facilitate an understanding where it is necessary. The resultant expressions are too huge to be included here and are available in reference [17].

```
ARRAY X(3),C1(2,2,2),C2(2,2,2);
OPERATOR U,V,W,F;
MATRIX A(2,2),CONTRA(2,2),D(2,2);

%=======================================
% INPUTTING SURFACE PARAMETRIC FUNCTIONS
%=======================================

X(1):=U(1);
X(2):=U(2);
X(3):=CONSTANT;

%=========================================================
% CALCULATING COVARIANT METRIC TENSOR & ITS DETERMINANT
%=========================================================

FOR M:=1:2 DO FOR N:=1:2 DO
A(M,N):=FOR I:=1:3 SUM DF(X(I),U(M))*DF(X(I),U(N));
DETA:=DET(A);
DEPEND W,U(1),U(2);
FOR I:=1:2 DO DEPEND V(I),U(1),U(2);

%=====================================================
%CALCULATING CONTRAVARIANT METRIC TENSOR COMPONENT
%=====================================================
```

```
FOR L:=1:2 DO FOR M:=1:2 DO
IF L=1 and M=1 THEN CONTRA(L,M):=A(2,2)/DETA
ELSE IF L NEQ M THEN CONTRA(L,M):=-A(M,L)/DETA
ELSE CONTRA(L,M):=A(1,1)/DETA;

%=====================================
%CALCULATING THE 1ST CHRISTOLFFEL SYMBOL
%=====================================

FOR L:=1:2 DO FOR M:=1:2 DO FOR N:=1:2 DO
C1(L,M,N):=(1/2)*(DF(A(L,N),U(M))+DF(A(M,N),U(L))-DF(A(L,M),U(N)));

%=================================================
%CALCULATING THE 2ND CHRISTOLFFEL SYMBOL (SEE EQ. 4.6)
%=================================================

FOR L:=1:2 DO FOR M:=1:2 DO FOR N:=1:2 DO
C2(L,M,N):=FOR I:=1:2 SUM A(L,I)*C1(M,N,I);

%=====================================================
%SUBROUTINE FOR CALCULATING THE COVARIANT DERIVATIVE
%FOR THE 1ST ORDER TENSOR
%=====================================================

PROCEDURE COVD(L,VAR(M));
DF(VAR(M),U(L))-(FOR I:=1:2 SUM C2(I,L,M)*VAR(I));

%=====================================================
%SUBROUTINE FOR CALCULATINE THE COVARIANT DERIVATIVE
%FOR THE 2ND ORDER TENSOR
%=====================================================

PROCEDURE COVD2(L,FUN(M,N));
DF(FUN(M,N),U(L))-(FOR I:=1:2 SUM C2(I,L,M)*FUN(I,N))
-(FOR I:=1:2 SUM C2(I,L,N)*FUN(M,I));
MATRIX E(2,2),K(2,2),P(2,2),FF(2,3),XX(1,3);

%=============================================
%CALCULATING RELATIVE ALTERNATE TENSOR
%ALSO CALLED AS GENERALIZED KRONECKER DELTA
%=============================================

PROCEDURE EE(I,J,K);
IF I=J OR J=K OR K=I THEN 0
ELSE IF
   (I=1 AND J=2 AND K=3)
 OR (I=3 AND J=1 AND K=2)
 OR (I=2 AND J=3 AND K=1)
    THEN 1
ELSE -1;

%=================================================
%CALCULATING THE CURVATURE TENSOR & ITS DETERMINANT
%=================================================
```

```
FOR I:=1:2 DO FOR J:=1:3 DO
FF(I,J):=DF(X(J),U(I));
FOR I:=1:3 DO
 XX(1,I):=(1/SQRT(DETA))*(FOR J:=1:3 SUM
      (FOR K:=1:3 SUM EE(I,J,K)*FF(1,J)*FF(2,K)));
FOR I:=1:2 DO FOR J:=1:2 DO
 D(I,J):=FOR I1:=1:3 SUM XX(1,I1)*DF(FF(I,I1),U(J));
DETD:=DET(D);

%================================================
%CALCULATING THE GENERIZED DISPLACEMENT TENSOR
%================================================

FOR L:=1:2 DO FOR M:=1:2 DO
P(L,M):=DF(V(M),U(L))-(FOR I:=1:2 SUM C2(I,L,M)*V(I))-D(L,M)*W;

%=============================================
%CALCULATING THE ROTATION TENSOR (SEE EQ. 4.17)
%=============================================

ARRAY Q(2);
FOR M:=1:2 DO
Q(M):=DF(W,U(M))+(FOR I:=1:2 SUM D(M,I)*(FOR J:=1:2 SUM CONTRA(I,J)*V(J)));

%===========================
%DERIVING THE STRAIN TENSOR
%===========================

FOR I:=1:2 DO FOR J:=1:2 DO
E(I,J):=(1/2)*(P(I,J)+P(J,I)
 +(FOR I1:=1:2 SUM (FOR J1:=1:2 SUM CONTRA(I1,J1)*P(I,J1))* P(J,I1))+Q(I)*Q(J));
OFF PERIOD;
ON FORT;
OFF PERIOD;
FOR I:=1:2 DO FOR J:=1:2 DO
WRITE "      E(",I,",",J,")=",E(I,J);

%=============================================
%CALCULATING THE ABSOLUTE 2-D ALTERNATE TENSOR
%=============================================

PROCEDURE EPS(L,M);
IF L=1 AND M=2 THEN SQRT(DETA)
ELSE IF L=2 AND M=1 THEN -SQRT(DETA)
ELSE 0;

%=========================================================
%CALCULATE CONTRAVARIANT ABSOLUTE 2-D ALTERNATE TENSOR
%=========================================================

PROCEDURE CEPS(L,M);
IF L=1 AND M=2 THEN 1/SQRT(DETA)
ELSE IF L=2 AND M=1 THEN -1/SQRT(DETA)
```

```
ELSE 0;

%================================
%CALCULATING THE BENDING TENSOR
%================================

FOR I:=1:2 DO FOR J:=1:2 DO
<<K(I,J):=SQRT(DETA/AD)*((1+(FOR L:=1:2 SUM CONTRA(L,L)*P(L,L))
   +DET(P)/DETA)*(D(I,J)+DF(Q(I),U(J))
   -(FOR I1:=1:2 SUM C2(I1,J,I)*Q(I1))
   +(FOR L:=1:2 SUM (FOR M:=1:2 SUM CONTRA(L,M)*D(J,M))*P(I,L)))
   -(FOR L:=1:2 SUM ((FOR M:=1:2 SUM CONTRA(L,M)*Q(M))
   +(FOR N:=1:2 SUM CEPS(L,N)*(FOR S:=1:2 SUM
   (FOR R:=1:2 SUM CEPS(R,S)*Q(R))*P(S,N))))*(DF(P(I,L),U(J))
   -(FOR I2:=1:2 SUM C2(I2,J,I)*P(I2,L))
   -(FOR I2:=1:2 SUM C2(I2,J,L)
   *P(I,I2))-D(J,L)*Q(I))))-D(I,J);
WRITE "      K(",i,",",j,")=",K(I,J)>>;
DEPEND AD,U(1),U(2);
RAD:=DET(A+2*E)$
WRITE "     AD=",RAD;

%================================
%DERIVING THE CONSTITUTIVE EQUATIONS
%================================

MATRIX N(2,2),M(2,2);
FOR I:=1:2 DO FOR J:=1:2 DO

%-----------------------------------------------------
%EFFECTIVE MEMBRANE STRESS TENSOR
%-----------------------------------------------------

<<N(I,J):=(YE/(1-VV**2))*((1-VV)*(FOR L:=1:2 SUM CONTRA(L,I)*
  (FOR I1:=1:2 SUM CONTRA(I1,J)*E(L,I1)))+VV*CONTRA(I,J)*
  (FOR L:=1:2 SUM CONTRA(L,L)*E(L,L)));

%---------------------------------------
%EFFECTIVE MOMENT TENSOR
%---------------------------------------

 M(I,J):=(YE*H**3/(12*(1-VV**2)))*((1-VV)
   *(FOR L:=1:2 SUM CONTRA(L,I)*
   (FOR I1:=1:2 SUM CONTRA(I1,J)*K(L,I1)))
   +VV*CONTRA(I,J)*(FOR L:=1:2
   SUM A(L,L)*K(L,L)))>>;

%=================================================
%WRITING-OUT EXPRESSIONS OF STRESS AND MOMENT TENSOR
%=================================================

 FOR I:=1:2 DO FOR J:=1:2 DO
  WRITE "      N(",I,",",J,")=",N(I,J);
 FOR I:=1:2 DO FOR J:=1:2 DO
```

```
 WRITE "      M(",I,",",J,")=",M(I,J);

 %=================================
%DERIVING THE EQUILIBRIUM EQUATIONS
 %=================================

ARRAY M1(2),N1(2);
FOR I:=1:2 DO
M1(I):=FOR J:=1:2 SUM
  (DF(M(I,J),U(J))+(FOR I1:=1:2 SUM C2(I,I1,J)*M(I1,J))
  +(FOR I1:=1:2 SUM C2(J,I1,J)*M(I,I1)));
MM:=FOR I:=1:2 SUM
  (DF(M1(I),U(I))+(FOR I1:=1:2 SUM C2(I,I1,I)*M1(I1)));
BEQ:=MM-(FOR I1:=1:2 SUM
  (FOR I2:=1:2 SUM D(I1,I2)*(FOR I3:=1:2 SUM
     CONTRA(I2,I3)*(FOR I4:=1:2 SUM D(I4,I3)*M(I1,I4)))))
     -(FOR I1:=1:2 SUM
     (FOR I2:=1:2 SUM D(I1,I2)*N(I1,I2)))-PP;
FOR I:=1:2 DO
N1(I):=FOR J:=1:2 SUM
       DF(N(I,J),U(J))+(FOR I1:=1:2 SUM C2(I,I1,J)*N(I1,J))
       +(FOR I1:=1:2 SUM C2(J,I1,J)*N(I,I1))
       +2*(FOR I1:=1:2 SUM CONTRA(I,I1)*
      (FOR I2:=1:2 SUM D(I2,I1)*(DF(M(I2,J),U(J))
       +(FOR I3:=1:2 SUM C2(I2,I3,J)*M(I3,J))
       +(FOR I3:=1:2 SUM C2(J,I3,J)*M(I2,I3)))))-
      (FOR I1:=1:2 SUM M(I1,J)
      *(FOR I2:=1:2 SUM D(I1,I2)*(DF(CONTRA(I,I2),U(J))
      +(FOR I3:=1:2 SUM C2(I,I3,J)*CONTRA(I3,I2))
      +(FOR I3:=1:2 SUM C2(I2,I3,J)*CONTRA(I,I3))))
      +(FOR I2:=1:2 SUM CONTRA(I1,I2)*(DF(D(I1,I2),U(J))-
      (FOR I3:=1:2 SUM C2(I3,I1,J)*D(I3,I2))-
      (FOR I3:=1:2 SUM C2(I3,I2,J)*D(I1,I3)))))+F(I);
WRITE BEQ,"     =0";
FOR I:=1:2 DO WRITE N1(I),"     =0";
BYE;
```

4.3.3 Remarks

The REDUCE program and solution shown above are just for the case of plates. For the geometry other than plates, the program is easily modified by changing the input parametric equations in the beginning of program. As the results show, the formulae of strain, bending, stress, moment tensors and equilibrium equations are functions of three displacements and their derivatives. Therefore we may assume three displacement fields as polynomial (or power series, Fourier series etc.), neglect the unnecessary terms and solve the problem. Since it is out of the scope of this report, it will be left to the interested researchers.

4.4 Approximation of a function by Fourier series

4.4.1 Introduction

Mathematically, two functions which belong to different spaces are apparently different in properties. It is theoretically impossible to replace one in terms of the other. However, to approximate one by the other is feasible and is actually adopted by many engineers in various fields. The purpose of approximation depends on the problem. It may be for avoiding the mathematical difficulties or for simplifying the function so that it can be solved easily. Some popular methods to approximate a function include Taylor series approximation and Fourier series expansion etc. Since they are the approximation methods, the accuracy is up to the number of terms included. Of course, as more terms are involved in the formulation, more complicated expression will turn out and possibility of making errors is increased. Based on these reasons, even though a function can be approximated theoretically, a high accuracy in evaluation of such approximations is difficult to achieve. By the symbolic and algebraic manipulation, the approximation can be formulated without errors and the desired accuracy can be attained. Here, an example of Fourier series approximation to the hat function will be shown to demonstrate the advantages of application of symbolic and algebraic manipulation.

The hat function is a fundamentals of shape function in finite element method. It is defined as :

$$\begin{cases} f(x) = 1 + x & , \text{for} \quad -1 \le x < 0 \\ f(x) = 1 - x & , \text{for} \quad 0 \le x < -1 \end{cases}$$

Mathematically, it's a piecewise C^1 continuous function which certainly is different from the sinusoidal function which is C^∞ continuous function within the domain they span. The approximation to the hat function by Fourier series is expressed as

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \tag{4.23}$$

where the Fourier coefficients are evaluated as follows

$$a_0 = \frac{1}{L}\int_{-L}^{L} f(x)dx \tag{4.24}$$

$$a_n = \frac{1}{L}\int_{-L}^{L} f(x)\cos(\frac{n\pi x}{L})dx \tag{4.25}$$

$$b_n = \frac{1}{L}\int_{-L}^{L} f(x)\sin(\frac{n\pi x}{L})dx \tag{4.26}$$

4.4.2 REDUCE program for generating Fourier series

The REDUCE program to generate the Fourier series and output a fortran subroutine for hat function is shown in the follow assuming L=1.0. This program can be used to generate Fourier series for an arbitrary function by simply changing the input function.

```
%==================================
% Inputting the given function and informations.
% M    : number of piecewise bounded interval.
% f(M): the function in the Mth interval.
% c(n) : the upper and lower limit of finite integration.
% l     : half length of interval.
% k     : number of Fourier series terms needed.
%==================================

M:=2;
K:=50;
ARRAY F(M),C(M+1);
f(1):=1+x; %c(1) =< x =< c(2)
f(2):=1-x; %c(2) =< x <  c(3)
c(1):=-1;c(2):=0;c(3):=1;
l:=(c(m+1)-c(1))/2;

%============================
% Obtaining the Fourier series coefficients
%============================

a0:=for i:=1:m sum
(sub(x=c(i+1),int(f(i),x))-sub(x=c(i),int(f(i),x)))/l;
an:=for i:=1:m sum
    (sub(x=c(i+1),int(f(i)*cos(n*pi*x/l),x))
    -sub(x=c(i),int(f(i)*cos(n*pi*x/l),x)))/l;
bn:=for i:=1:m sum
    (sub(x=c(i+1),int(f(i)*sin(n*pi*x/l),x))
    -sub(x=c(i),int(f(i)*sin(n*pi*x/l),x)))/l;
on rat;
on div;

%==================
% Generating the Fourier series.
%==================
```

```
fs:=a0/2+for i:=1:k sum
sub(n=i,an)*cos(i*pi*x/l)+sub(n=i,bn)*sin(i*pi*x/l);
off echo;
on fort;
cardno!*:=10;

%=================================
% Outputting the fortran subroutine of Fourier series.
%=================================

out "fourier.ftn";
write "     subroutine fourier(x,fs)";
write "     implicit real*8(a-h,o-z)";
write "     pi=3.141592654";
fs:=fs;
write "     return";
write "     end";
shut "fourier.ftn";
bye;
```

4.4.3 Resultant fortran subroutine from REDUCE

The following results are produced automatically from the above REDUCE program for fifty terms Fourier series case. This subroutine can be directly input into fortran main program without any troubles.

```
      subroutine fourier(x,fs)
      implicit real*8(a-h,o-z)
      pi=3.141592654
       ANS1=4./625.*COS(25.*PI*X)*PI**(-2)+4./529.*COS(23.*
     . PI*X)*PI**(-2)+4./441.*COS(21.*PI*X)*PI**(-2)+4./
     . 361.*COS(19.*PI*X)*PI**(-2)+4./289.*COS(17.*PI*X)*PI
     . **(-2)+4./225.*COS(15.*PI*X)*PI**(-2)+4./169.*COS(
     . 13.*PI*X)*PI**(-2)+4./121.*COS(11.*PI*X)*PI**(-2)+4./
     . 81.*COS(9.*PI*X)*PI**(-2)+4./49.*COS(7.*PI*X)*PI**(-2
     . )+4./25.*COS(5.*PI*X)*PI**(-2)+4./9.*COS(3.*PI*X)*PI
     . **(-2)+1./2.
       FS=4.*COS(PI*X)*PI**(-2)+4./2401.*COS(49.*PI*X)*PI**(
     . -2)+4./2209.*COS(47.*PI*X)*PI**(-2)+4./2025.*COS(45.
     . *PI*X)*PI**(-2)+4./1849.*COS(43.*PI*X)*PI**(-2)+4./
     . 1681.*COS(41.*PI*X)*PI**(-2)+4./1521.*COS(39.*PI*X)*
     . PI**(-2)+4./1369.*COS(37.*PI*X)*PI**(-2)+4./1225.*
     . COS(35.*PI*X)*PI**(-2)+4./1089.*COS(33.*PI*X)*PI**(
     . -2)+4./961.*COS(31.*PI*X)*PI**(-2)+4./841.*COS(29.*
     . PI*X)*PI**(-2)+4./729.*COS(27.*PI*X)*PI**(-2)+ANS1
      return
      end
```

Four curves are plotted in the same figure for three-term, five-term , seven-term and fifty-term cases. As the figure shows, the hat function can be actually simulated by Fourier series. When fifty terms are used, the difference between hat function and its Fourier series is just invisible although they are in different spaces. This is one of the advantages of the application of symbolic and algebraic manipulation.

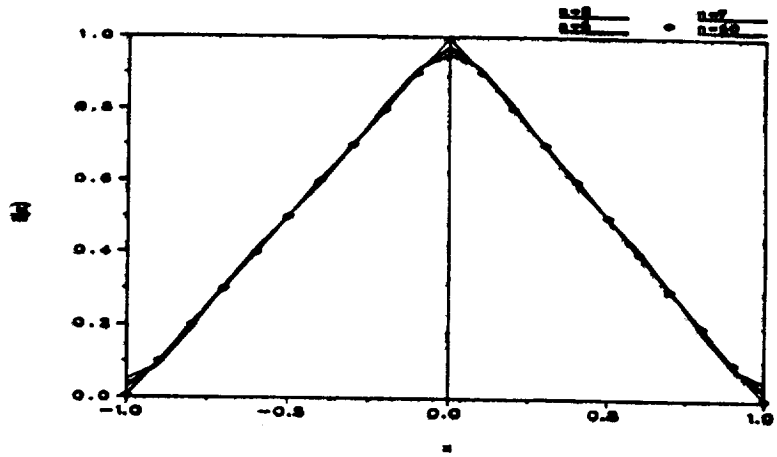

Figure 4.3 : Convergence of Fourier series approximation

4.5 Template for nonlinear numerical analysis

4.5.1 Introduction

In nonlinear numerical analysis, it is necessary to evaluate the Jacobian matrix and to solve the system of equations at each iteration. Symbolically, the Newton-Raphson iteration can be expressed as

$$\{X\}^{(k)} = \{X\}^{(k-1)} - [J]^{-1}\{F\} \tag{4.27}$$

Since the evaluation of the inverse of the Jacobian matrix is quite time-consuming, equation (4.27) is traditionally changed into the following form and then is solved as a linear system of equations at each iteration stage.

$$[J]\{\Delta X\}^{(k)} = -\{F\} \tag{4.28}$$

Although the avoidance in evaluation of the inverse of the Jacobian does expedite the execution, it still takes time to solve a system of equations, especially for the case of a large number of unknowns. With the help of symbolic and algebraic manipulation, the efficiency can be improved further.

Instead of transforming equation (4.27) into equation (4.28), it is rearranged into

$$\{X\}^{(k)} - \{X\}^{(k-1)} = \{\Delta X\}^{(k)} = -[J]^{-1}\{F\} \tag{4.29}$$

The application of symbolic and algebraic manipulation to this problem is to make a template form of the right hand side of equation (4.29). This includes the symbolic evaluation of the Jacobian matrix, its inverse, and the multiplication of the matrix by load vector. The results are then converted into fortran code for numerical analysis. The iterations are done by simply substituting the current solutions into the template formulae to get the residuals. This substitution is much faster than solving the system of equations. This is a new way to improve the efficiency of program execution.

4.5.2 Preliminary formulation

The above idea is demonstrated by solving the Fermat-Weber location problem for the case of a two-dimensional Euclidean space. Given n fixed points ,(x_i, y_i), i=1,..,n, in the plane, the object of the Fermat-Weber problem is to find the best location to minimize the total length of the distances among the optimal location and each point. Mathematically, it is

$$\textit{Minimize} \sum_{i=1}^{n} \sqrt{(x - x_i)^2 + (y - y_i)^2} \tag{4.30}$$

or rewritten in the following form

$$\textit{Minimize} \sum_{i=1}^{n} \| c_i - A_i^T Y \|_2 \tag{4.31}$$

where (x,y) is the coordinate of the optimal location to be found, and

$$c_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{4.32}$$

$$A_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.33}$$

$$Y = \begin{pmatrix} x \\ y \end{pmatrix} \tag{4.34}$$

Intuitively, the minimization will be accomplished by differentiating equation (4.31) with respect to $\boldsymbol{Y}$ and setting the results to zero. But since the objective function in equation (4.31) is not differentiable at the exact (x_i, y_i) points, a smoothing parameter e is introduced into the object function to avoid the difficulty.

$$\Phi_\varepsilon(Y) = \sum_{i=1}^{n} \sqrt{\| c_i - A_i^T Y \|_2^2 + \varepsilon^2} \qquad (4.35)$$

The perturbed objective function now becomes differentiable everywhere and is strictly convex. Then $\frac{\partial \Phi_\varepsilon}{\partial Y} = 0$ will result in a fixed point iteration

$$Y^{(k+1)} = \left[\sum w_i^{(k)} A_i A_i^T\right]^{-1} \left(\sum w_i^{(k)} A_i c_i\right) \qquad , k = 0, 1, 2 \cdots \qquad (4.36)$$

Where the weight $w_i^{(k)}$ is defined as

$$w_i^{(k)} = \frac{1}{\sqrt{\| c_i - A_i^T Y \|_2^2 + \varepsilon^2}} \qquad (4.37)$$

4.5.3 REDUCE program

Consider the specific problem which is to find the optimal location among three points (0,1), (0,-1) and (x,0) where x varies in the range [0, ∞). The REDUCE program is based on equation (4.36).

```
%----------------------------------------
% waa  : W*A*A in equation (4.36)
% wac  : W*A*C in equation (4.36)
%----------------------------------------

matrix waa(2,2),wac(2,1);
waa(1,1):=1/sqrt(r1)+1/sqrt(r2)+1/sqrt(r3);
waa(2,2):=waa(1,1);
wac(1,1):=x1/sqrt(r1)+x2/sqrt(r2)+x3/sqrt(r3);
wac(2,1):=y1/sqrt(r1)+y2/sqrt(r2)+y3/sqrt(r3);
wac:=(1/waa)*wac;
on fort;
off echo;
off period;
cardno!*:=10;
out "cssa.ftn";
write "     subroutine cssa(x1,y1,x2,y2,x3,y3,pu,x0,y0)";
write "    implicit real*8(a-h,o-z)";
write "    r1=(x-x1)**2+(y-y1)**2+pu**2";
write "    r2=(x-x2)**2+(y-y2)**2+pu**2";
write "    r3=(x-x3)**2+(y-y3)**2+pu**2";
write "    x=",wac(1,1);
write "    y=",wac(2,1);
write "    return";
write "    end";
shut "cssa.ftn";
bye;
```

4.5.4 Fortran subroutine from REDUCE

The fortran subroutine produced here is applicable when the number of points is three. The coordinates of three points are arbitrary. The perturbation parameter is an arbitrarily small number except zero. The resultant optimal locations are plotted in Figure 4.4. with respect to variable *x3*. The execution time for this problem on Apollo workstation Domain 4000 is too small to be measured. The difference in execution time will be more significant when the size of the problem is increased.

```
 subroutine cssa(x1,y1,x2,y2,x3,y3,pu,x,y)
implicit real*8(a-h,o-z)
r1=(x-x1)**2+(y-y1)**2+pu**2
r2=(x-x2)**2+(y-y2)**2+pu**2
r3=(x-x3)**2+(y-y3)**2+pu**2
x=(SQRT(R2)*SQRT(R1)*R3*X1+SQRT(R2)*SQRT(R1)*R3*X2+
. SQRT(R3)*SQRT(R1)*R2*X1+SQRT(R3)*SQRT(R1)*R2*X3+SQRT
. (R3)*SQRT(R2)*R1*X2+SQRT(R3)*SQRT(R2)*R1*X3+R1*R2*X3
. +R1*R3*X2+R2*R3*X1)/(2*SQRT(R2)*SQRT(R1)*R3+2*SQRT(
. R3)*SQRT(R1)*R2+2*SQRT(R3)*SQRT(R2)*R1+R1*R2+R1*R3+
. R2*R3)
y=(SQRT(R2)*SQRT(R1)*R3*Y1+SQRT(R2)*SQRT(R1)*R3*Y2+
. SQRT(R3)*SQRT(R1)*R2*Y1+SQRT(R3)*SQRT(R1)*R2*Y3+SQRT
. (R3)*SQRT(R2)*R1*Y2+SQRT(R3)*SQRT(R2)*R1*Y3+R1*R2*Y3
. +R1*R3*Y2+R2*R3*Y1)/(2*SQRT(R2)*SQRT(R1)*R3+2*SQRT(
. R3)*SQRT(R1)*R2+2*SQRT(R3)*SQRT(R2)*R1+R1*R2+R1*R3+
. R2*R3)
return
end
```

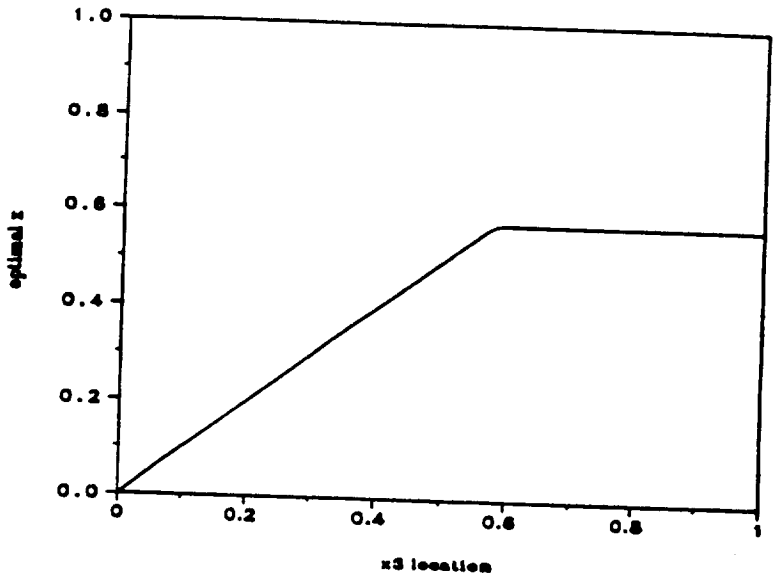

Figure 4.4 : Behavior of optimal location vs. variable x3

4.6 Triangular stiffness and mass matrix construction

4.6.1 Preliminary formulation

It is well known that the finite element method has become a powerful tool in solving general engineering problems. Usually the finite element method is applied in three steps:

1. Domain discretization (pre-processing).

2. Constructing the stiffness, mass matrices and load vectors etc., then prescribing the boundary conditions and solving them.

3. Results treatment (post-processing).

While computing stiffness and mass matrices, partial differentiation, matrix multiplication, matrix inversion and integration are required. For a higher order interpolation, the formulation is always very tedious and prone to introduce errors. With the aid of symbolic and algebraic manipulation, all these troubles can be alleviated. The example shown in this section will demonstrate the application of symbolic and algebraic manipulation to the automatic construction of stiffness and mass matrix of six-node triangular element for a heat transfer problem.

The stiffness **K** and mass **M** matrices are defined as follows assuming unit thickness.

$$K = \int_{\Omega} B^T * D * B dA \tag{4.38}$$

$$M = \int_{\Omega} N^T * N * \rho * c_p dA \tag{4.39}$$

where the shape functions are

$$N_4 = 4 * s_1 * s_2, \quad N_5 = 4 * s_2 * s_3, \quad N_6 = 4 * s_3 * s_1,$$
$$N_1 = s_1 - \tfrac{1}{2}(N_6 + N_4), \quad N_2 = s_2 - \tfrac{1}{2}(N_4 + N_5), \quad N_3 = s_3 - \tfrac{1}{2}(N_5 + N_6) \tag{4.40}$$

The **B** matrix is evaluated by the formulae of

$$B^T = \begin{bmatrix} \frac{\partial N_1}{\partial s_1} & \frac{\partial N_1}{\partial s_2} & \frac{\partial N_1}{\partial s_3} \\ \frac{\partial N_2}{\partial s_1} & \frac{\partial N_2}{\partial s_2} & \frac{\partial N_2}{\partial s_3} \\ \frac{\partial N_3}{\partial s_1} & \frac{\partial N_3}{\partial s_2} & \frac{\partial N_3}{\partial s_3} \\ \frac{\partial N_4}{\partial s_1} & \frac{\partial N_4}{\partial s_2} & \frac{\partial N_4}{\partial s_3} \\ \frac{\partial N_5}{\partial s_1} & \frac{\partial N_5}{\partial s_2} & \frac{\partial N_5}{\partial s_3} \\ \frac{\partial N_6}{\partial s_1} & \frac{\partial N_6}{\partial s_2} & \frac{\partial N_6}{\partial s_3} \end{bmatrix} \begin{bmatrix} b_1 & c_1 \\ b_2 & c_2 \\ b_3 & c_3 \end{bmatrix} \tag{4.41}$$

The material property matrix **D** is assumed to be symmetric

$$D = \begin{bmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{bmatrix} \tag{4.42}$$

The b_i, c_i here are expressed by global nodal coordinates.

$$\begin{aligned} b_1 &= \frac{1}{J}(y_2 - y_3), \quad b_2 = \frac{1}{J}(y_3 - y_1), \quad b_3 = \frac{1}{J}(y_1 - y_2) \\ c_1 &= \frac{1}{J}(x_3 - x_2), \quad c_2 = \frac{1}{J}(x_1 - x_3), \quad c_3 = \frac{1}{J}(x_2 - x_1) \end{aligned} \tag{4.43}$$

The Jacobian J is equal to

$$J = (x_1 - x_3)(y_2 - y_3) - (y_1 - y_3)(x_2 - x_3) = 2 * area \tag{4.44}$$

4.6.2 REDUCE program for stiffness matrix

```
MATRIX NS(6,3),NS1(2,6),NS2(2,6),NS3(2,6),BB(2,6);
ARRAY N(6),B(3),C(3);
%---------------------------------
%Inputting the shape functions
%---------------------------------
N(4):=4*S1*S2;
N(5):=4*S2*S3;
N(6):=4*S3*S1;
N(1):=S1-(1/2)*(N(6)+N(4));
N(2):=S2-(1/2)*(N(4)+N(5));
N(3):=S3-(1/2)*(N(5)+N(6));
```

```
%-----------------------------------------------------------------------------
%Calculate the Jacobian, area and b(i),c(i) defined in Eq.(4.43)& (4.44)
%-----------------------------------------------------------------------------
JAC:=(X1-X3)*(Y2-Y3)-(X2-X3)*(Y1-Y3);
AREA:=JAC/2;
B(1):=(Y2-Y3)/JAC;
B(2):=(Y3-Y1)/JAC;
C(1):=(X3-X2)/JAC;
C(2):=(X1-X3)/JAC;
B(3):=-B(1)-B(2);
C(3):=-C(1)-C(2);
%-----------------------------------------------
%Calculating B matrix defined in Eq. (4.41).
%-----------------------------------------------
FOR M:=1:2 DO <<
FOR I:=1:6 DO <<
NS(I,1):=DF(N(I),S1);NS(I,2):=DF(N(I),S2);NS(I,3):=DF(N(I),S3);
IF M=1 THEN BB(M,I):=FOR J:=1:3 SUM (B(J)*NS(I,J)) ELSE
BB(M,I):=FOR J:=1:3 SUM (C(J)*NS(I,J))>>>>;
FOR I:=1:2 DO <<FOR J:=1:6 DO
<<NS1(I,J):=SUB(S1=2/3,S2=1/6,S3=1/6,BB(I,J));
  NS2(I,J):=SUB(S1=1/6,S2=2/3,S3=1/6,BB(I,J));
  NS3(I,J):=SUB(S1=1/6,S2=1/6,S3=2/3,BB(I,J))>>>>;
%-----------------------------------------
%Given the symmetric material matrix.
%-----------------------------------------
MATRIX D(2,2);
D:=MAT((K11,K12),(K12,K22));
MATRIX LO(6,6),NU(6,6),CC(6,6),SKE(6,6);
%---------------------------------
%Obtaining the stiffness matrix.
%---------------------------------
SKE:=(AREA/3)*(TP(NS1)*D*NS1+TP(NS2)*D*NS2+TP(NS3)*D*NS3)$
%-----------------------------------------------------------------------------
%Making the appropriate substitution to simplify the final expression.
%-----------------------------------------------------------------------------
COB:=-(X3*Y2-X3*Y1-X2*Y3+X2*Y1+X1*Y3-X1*Y2);
FOR I:=1:6 DO FOR J:=1:6 DO
<<LO(I,J):=DEN(SKE(I,J));NU(I,J):=NUM(SKE(I,J));CC(I,J):=LO(I,J)/COB;
SKE(I,J):=NU(I,J)/(CC(I,J)*DJ)>>;
%-----------------------------------------------
%Outputting the resultant fortran subroutine.
%-----------------------------------------------
ON FORT;
OFF ECHO;
OFF PERIOD;
OUT "sym";
WRITE "      subroutine stiff(x1,y1,x2,y2,x3,y3,d11,d12,d22,ske)";
WRITE "     implicit real*8(a-h,o-z)";
WRITE "     dimension ske(6,6)";
WRITE "     DJ=-(X3*Y2-X3*Y1-X2*Y3+X2*Y1+X1*Y3-X1*Y2)";
FOR I:=1:6 DO FOR J:=1:6 DO
IF J>=I THEN WRITE "SKE(",I,",",J,")=",SKE(I,J)
ELSE WRITE "SKE(",I,",",J,")=SKE(",J,",",I,")";
```

```
WRITE "     return";
WRITE "     end";
SHUT "sym";
OFF FORT;
BYE;
```

4.6.3 Resultant stiffness matrix made by REDUCE

```
      subroutine stiff(x1,y1,x2,y2,x3,y3,d11,d12,d22,ske)
      implicit real*8(a-h,o-z)
      dimension ske(6,6)
      DJ=-(X3*Y2-X3*Y1-X2*Y3+X2*Y1+X1*Y3-X1*Y2)
      SKE(1,1)=(D11*Y2**2-2*D11*Y2*Y3+D11*Y3**2-2*D12*X2*Y2
     . +2*D12*X2*Y3+2*D12*X3*Y2-2*D12*X3*Y3+D22*X2**2-2*D22
     . *X2*X3+D22*X3**2)/(2*DJ)
      SKE(1,2)=(D11*Y1*Y2-D11*Y1*Y3-D11*Y2*Y3+D11*Y3**2-D12
     . *X1*Y2+D12*X1*Y3-D12*X2*Y1+D12*X2*Y3+D12*X3*Y1+D12*
     . X3*Y2-2*D12*X3*Y3+D22*X1*X2-D22*X1*X3-D22*X2*X3+D22*
     . X3**2)/(6*DJ)
      SKE(1,3)=-(D11*Y1*Y2-D11*Y1*Y3-D11*Y2**2+D11*Y2*Y3-
     . D12*X1*Y2+D12*X1*Y3-D12*X2*Y1+2*D12*X2*Y2-D12*X2*Y3+
     . D12*X3*Y1-D12*X3*Y2+D22*X1*X2-D22*X1*X3-D22*X2**2+
     . D22*X2*X3)/(6*DJ)
      SKE(1,4)=-(2*(D11*Y1*Y2-D11*Y1*Y3-D11*Y2*Y3+D11*Y3**2
     . -D12*X1*Y2+D12*X1*Y3-D12*X2*Y1+D12*X2*Y3+D12*X3*Y1+
     . D12*X3*Y2-2*D12*X3*Y3+D22*X1*X2-D22*X1*X3-D22*X2*X3+
     . D22*X3**2))/(3*DJ)
      SKE(1,5)=0
      SKE(1,6)=(2*(D11*Y1*Y2-D11*Y1*Y3-D11*Y2**2+D11*Y2*Y3-
     . D12*X1*Y2+D12*X1*Y3-D12*X2*Y1+2*D12*X2*Y2-D12*X2*Y3+
     . D12*X3*Y1-D12*X3*Y2+D22*X1*X2-D22*X1*X3-D22*X2**2+
     . D22*X2*X3))/(3*DJ)
      SKE(2,1)=SKE(1,2)
      SKE(2,2)=(D11*Y1**2-2*D11*Y1*Y3+D11*Y3**2-2*D12*X1*Y1
     . +2*D12*X1*Y3+2*D12*X3*Y1-2*D12*X3*Y3+D22*X1**2-2*D22
     . *X1*X3+D22*X3**2)/(2*DJ)
      SKE(2,3)=(D11*Y1**2-D11*Y1*Y2-D11*Y1*Y3+D11*Y2*Y3-2*
     . D12*X1*Y1+D12*X1*Y2+D12*X1*Y3+D12*X2*Y1-D12*X2*Y3+
     . D12*X3*Y1-D12*X3*Y2+D22*X1**2-D22*X1*X2-D22*X1*X3+
     . D22*X2*X3)/(6*DJ)
      SKE(2,4)=-(2*(D11*Y1*Y2-D11*Y1*Y3-D11*Y2*Y3+D11*Y3**2
     . -D12*X1*Y2+D12*X1*Y3-D12*X2*Y1+D12*X2*Y3+D12*X3*Y1+
     . D12*X3*Y2-2*D12*X3*Y3+D22*X1*X2-D22*X1*X3-D22*X2*X3+
     . D22*X3**2))/(3*DJ)
      SKE(2,5)=-(2*(D11*Y1**2-D11*Y1*Y2-D11*Y1*Y3+D11*Y2*Y3
     . -2*D12*X1*Y1+D12*X1*Y2+D12*X1*Y3+D12*X2*Y1-D12*X2*Y3
     . +D12*X3*Y1-D12*X3*Y2+D22*X1**2-D22*X1*X2-D22*X1*X3+
     . D22*X2*X3))/(3*DJ)
      SKE(2,6)=0
      SKE(3,1)=SKE(1,3)
      SKE(3,2)=SKE(2,3)
      SKE(3,3)=(D11*Y1**2-2*D11*Y1*Y2+D11*Y2**2-2*D12*X1*Y1
     . +2*D12*X1*Y2+2*D12*X2*Y1-2*D12*X2*Y2+D22*X1**2-2*D22
```

```
. *X1*X2+D22*X2**2)/(2*DJ)
 SKE(3,4)=0
 SKE(3,5)=-(2*(D11*Y1**2-D11*Y1*Y2-D11*Y1*Y3+D11*Y2*Y3
. -2*D12*X1*Y1+D12*X1*Y2+D12*X1*Y3+D12*X2*Y1-D12*X2*Y3
. +D12*X3*Y1-D12*X3*Y2+D22*X1**2-D22*X1*X2-D22*X1*X3+
. D22*X2*X3))/(3*DJ)
 SKE(3,6)=(2*(D11*Y1*Y2-D11*Y1*Y3-D11*Y2**2+D11*Y2*Y3-
. D12*X1*Y2+D12*X1*Y3-D12*X2*Y1+2*D12*X2*Y2-D12*X2*Y3+
. D12*X3*Y1-D12*X3*Y2+D22*X1*X2-D22*X1*X3-D22*X2**2+
. D22*X2*X3))/(3*DJ)
 SKE(4,1)=SKE(1,4)
 SKE(4,2)=SKE(2,4)
 SKE(4,3)=SKE(3,4)
 SKE(4,4)=(4*(D11*Y1**2-D11*Y1*Y2-D11*Y1*Y3+D11*Y2**2-
. D11*Y2*Y3+D11*Y3**2-2*D12*X1*Y1+D12*X1*Y2+D12*X1*Y3+
. D12*X2*Y1-2*D12*X2*Y2+D12*X2*Y3+D12*X3*Y1+D12*X3*Y2-
. 2*D12*X3*Y3+D22*X1**2-D22*X1*X2-D22*X1*X3+D22*X2**2-
. D22*X2*X3+D22*X3**2))/(3*DJ)
 SKE(4,5)=(4*(D11*Y1*Y2-D11*Y1*Y3-D11*Y2**2+D11*Y2*Y3-
. D12*X1*Y2+D12*X1*Y3-D12*X2*Y1+2*D12*X2*Y2-D12*X2*Y3+
. D12*X3*Y1-D12*X3*Y2+D22*X1*X2-D22*X1*X3-D22*X2**2+
. D22*X2*X3))/(3*DJ)
 SKE(4,6)=-(4*(D11*Y1**2-D11*Y1*Y2-D11*Y1*Y3+D11*Y2*Y3
. -2*D12*X1*Y1+D12*X1*Y2+D12*X1*Y3+D12*X2*Y1-D12*X2*Y3
. +D12*X3*Y1-D12*X3*Y2+D22*X1**2-D22*X1*X2-D22*X1*X3+
. D22*X2*X3))/(3*DJ)
 SKE(5,1)=SKE(1,5)
 SKE(5,2)=SKE(2,5)
 SKE(5,3)=SKE(3,5)
 SKE(5,4)=SKE(4,5)
 SKE(5,5)=(4*(D11*Y1**2-D11*Y1*Y2-D11*Y1*Y3+D11*Y2**2-
. D11*Y2*Y3+D11*Y3**2-2*D12*X1*Y1+D12*X1*Y2+D12*X1*Y3+
. D12*X2*Y1-2*D12*X2*Y2+D12*X2*Y3+D12*X3*Y1+D12*X3*Y2-
. 2*D12*X3*Y3+D22*X1**2-D22*X1*X2-D22*X1*X3+D22*X2**2-
. D22*X2*X3+D22*X3**2))/(3*DJ)
 SKE(5,6)=-(4*(D11*Y1*Y2-D11*Y1*Y3-D11*Y2*Y3+D11*Y3**2
. -D12*X1*Y2+D12*X1*Y3-D12*X2*Y1+D12*X2*Y3+D12*X3*Y1+
. D12*X3*Y2-2*D12*X3*Y3+D22*X1*X2-D22*X1*X3-D22*X2*X3+
. D22*X3**2))/(3*DJ)
 SKE(6,1)=SKE(1,6)
 SKE(6,2)=SKE(2,6)
 SKE(6,3)=SKE(3,6)
 SKE(6,4)=SKE(4,6)
 SKE(6,5)=SKE(5,6)
 SKE(6,6)=(4*(D11*Y1**2-D11*Y1*Y2-D11*Y1*Y3+D11*Y2**2-
. D11*Y2*Y3+D11*Y3**2-2*D12*X1*Y1+D12*X1*Y2+D12*X1*Y3+
. D12*X2*Y1-2*D12*X2*Y2+D12*X2*Y3+D12*X3*Y1+D12*X3*Y2-
. 2*D12*X3*Y3+D22*X1**2-D22*X1*X2-D22*X1*X3+D22*X2**2-
. D22*X2*X3+D22*X3**2))/(3*DJ)
 return
 end
```

4.6.4 REDUCE program for mass matrix

```
MATRIX N(1,6),SQN(6,6),NS(6,6),DMASS(6,6),SDIF(6,6);
N:=MAT((N1,N2,N3,N4,N5,N6));
%-----------------------------
% Calculating the integrand.
%-----------------------------
SQN:=TP(N)*N;
%-----------------------------
% Inputting shape functions
%-----------------------------
N4:=4*S1*S2;
N5:=4*S2*S3;
N6:=4*S3*S1;
N1:=S1-(1/2)*(N6+N4);
N2:=S2-(1/2)*(N4+N5);
N3:=S3-(1/2)*(N5+N6);
%-----------------------------
% Evaluating the integration.
%-----------------------------
LET S3=1-S1-S2;
FOR I:=1:6 DO FOR J:=1:6 DO IF I<=J THEN <<
A:=INT(SQN(I,J),S2);B:=SUB(S2=1-S1,A)-SUB(S2=0,A);
C:=INT(B,S1);NS(I,J):=SUB(S1=1,C)-SUB(S1=0,C)>>
ELSE NS(I,J):=NS(J,I);
%-------------------------
% Outputting the results.
%-------------------------
ON FORT;
OFF ECHO;
OFF PERIOD;
OUT "mass.ftn";
WRITE "     SUBROUTINE MASS(X1,Y1,X2,Y2,X3,Y3,RO,CP,DMASS)";
WRITE "     IMPLICIT REAL*8(A-H,O-Z)";
WRITE "     DIMENSION DMASS(6,6)";
WRITE "     AREA=-(X3*Y2-X3*Y1-X2*Y3+X2*Y1+X1*Y3-X1*Y2)/2";
DMASS:=RO*CP*2*AREA*NS;
WRITE "     RETURN";
WRITE "     END";
SHUT "mass.ftn";

%---------------------------------------
% checking the correctness of results.
%---------------------------------------
OFF FORT; OUT "CHECK";
R:=FOR I:=1:6 SUM <<FOR J:=1:6 SUM DMASS(I,J)>>;
SDIF:=DMASS-TP(DMASS); SHUT "CHECK";
BYE;
```

4.6.5 Fortran results of mass matrix

```
      SUBROUTINE MASS(X1,Y1,X2,Y2,X3,Y3,RO,CP,DMASS)
      IMPLICIT REAL*8(A-H,O-Z)
```

```
DIMENSION DMASS(6,6)
AREA=-(X3*Y2-X3*Y1-X2*Y3+X2*Y1+X1*Y3-X1*Y2)/2
DMASS(1,1)=(AREA*CP*RO)/30
DMASS(1,2)=-(AREA*CP*RO)/180
DMASS(1,3)=-(AREA*CP*RO)/180
DMASS(1,4)=0
DMASS(1,5)=-(AREA*CP*RO)/45
DMASS(1,6)=0
DMASS(2,1)=-(AREA*CP*RO)/180
DMASS(2,2)=(AREA*CP*RO)/30
DMASS(2,3)=-(AREA*CP*RO)/180
DMASS(2,4)=0
DMASS(2,5)=0
DMASS(2,6)=-(AREA*CP*RO)/45
DMASS(3,1)=-(AREA*CP*RO)/180
DMASS(3,2)=-(AREA*CP*RO)/180
DMASS(3,3)=(AREA*CP*RO)/30
DMASS(3,4)=-(AREA*CP*RO)/45
DMASS(3,5)=0
DMASS(3,6)=0
DMASS(4,1)=0
DMASS(4,2)=0
DMASS(4,3)=-(AREA*CP*RO)/45
DMASS(4,4)=(8*AREA*CP*RO)/45
DMASS(4,5)=(4*AREA*CP*RO)/45
DMASS(4,6)=(4*AREA*CP*RO)/45
DMASS(5,1)=-(AREA*CP*RO)/45
DMASS(5,2)=0
DMASS(5,3)=0
DMASS(5,4)=(4*AREA*CP*RO)/45
DMASS(5,5)=(8*AREA*CP*RO)/45
DMASS(5,6)=(4*AREA*CP*RO)/45
DMASS(6,1)=0
DMASS(6,2)=-(AREA*CP*RO)/45
DMASS(6,3)=0
DMASS(6,4)=(4*AREA*CP*RO)/45
DMASS(6,5)=(4*AREA*CP*RO)/45
DMASS(6,6)=(8*AREA*CP*RO)/45
RETURN
END
```

4.6.6 Checking correctness of results

In some cases, the results from REDUCE are lengthy as the stiffness matrix shown above. It is very important to find a way to check their correctness. In general, a small problem with a known solution is used to test the correctness of a symbolic program before application to actual problems. In addition, some specific checking procedures are also available in each individual field. They require a knowledge of the specific field. Taking the mass matrix problem above as an example, the summation of the entries of the mass matrix should be equal to unit multiplied by the accessory constants. The symmetry of mass matrix is proven by

subtracting the mass matrix from its transpose to get zeros for each entries. The REDUCE program to check correctness is appended in the program shown in subsection 4.6.4. The following results include two parts. The first R is the summation of each entries of mass matrix. The second SDIF(i.j) are the difference of mass matrix and its transpose. These checking results confirm the correctness of REDUCE program.

```
%----------------------------------------------------------------------------
% Checking the correctness of mass matrix by summing each entries
% of mass matrix to make an unit multiplied by accessory constants
%----------------------------------------------------------------------------
R := AREA*CP*RO
%----------------------------------------------------------------------------
% Checking the symmetry of mass matrix by finding the difference
% between mass matrix and its transpose.
%----------------------------------------------------------------------------
SDIF(1,1) := 0
SDIF(1,2) := 0
SDIF(1,3) := 0
SDIF(1,4) := 0
SDIF(1,5) := 0
SDIF(1,6) := 0
SDIF(2,1) := 0
SDIF(2,2) := 0
SDIF(2,3) := 0
SDIF(2,4) := 0
SDIF(2,5) := 0
SDIF(2,6) := 0
SDIF(3,1) := 0
SDIF(3,2) := 0
SDIF(3,3) := 0
SDIF(3,4) := 0
SDIF(3,5) := 0
SDIF(3,6) := 0
SDIF(4,1) := 0
SDIF(4,2) := 0
SDIF(4,3) := 0
SDIF(4,4) := 0
SDIF(4,5) := 0
SDIF(4,6) := 0
SDIF(5,1) := 0
SDIF(5,2) := 0
SDIF(5,3) := 0
SDIF(5,4) := 0
SDIF(5,5) := 0
SDIF(5,6) := 0
SDIF(6,1) := 0
SDIF(6,2) := 0
SDIF(6,3) := 0
SDIF(6,4) := 0
SDIF(6,5) := 0
SDIF(6,6) := 0
```

4.7 Closed form solution of stiffness matrix of four-node element

4.7.1 Introduction

Although the methodology to construct the stiffness matrix of four-node isoparametric quadrilateral element for the plane elasticity problem is the same as that of the triangular element shown in last section, the techniques are quite different from each other. In the triangular element, the Jacobian determinant is constant and therefore the integration is straightforward. However the same advantage can't be gained for the isoparametric quadrilateral element. In general, the determinant of the Jacobian is a function of the natural coordinates. Having the of Jacobian determinant in the denominator of the integrand due to the coordinate transformation produces a tremendous difficulty in performing integration analytically, therefore the numerical Gauss quadrature rule is usually introduced to solve this problem. The discussions of this difficulty and the introduction of Gauss quadrature rule can be found in most relevant literatures, such as Zienkiewicz [10], Becker & Carey & Oden [11], Cook [12], Reddy [13], Huebner [14], Weaver & Johnson [15].

The inability to perform analytic integration introduces the integration error in the finite element results. The following paragraphs will show that this difficulty has been overcome and the exact closed-form solution has been obtained by appropriate application of REDUCE [8].

4.7.2 Preliminary formulation

The local stiffness matrix for a 2-D isoparametric quadrilateral element is formulated by

$$K = \int_{-1}^{1}\int_{-1}^{1} B^T * E * B * t * |J| \, d\xi \, d\eta \qquad (4.45)$$

Where

- K : local stiffness matrix.
- E : material property matrix.
- |JI : determinant of Jacobian matrix.
- ξ, η : natural coordinates.
- t : thickness of element.

• B : strain-displacement matrix.

In general, each entry of strain-displacement matrix B is a function of ξ, η, |J| and can be expressed as :

$$b_{ij}(\xi,\eta,|J|) = \frac{c_0 + c_1\xi + c_2\eta + c_3\xi\eta}{|J|} \tag{4.46}$$

Where

• b_{ij}: are denoted by the entry in ith row and jth column of matrix B

• c_i : are constants.

For simplicity, the material property matrix E and the thickness t are assumed to be independent of natural coordinates. The integrand in equation (4.45) therefore will be function of ξ, η and |J|, too. The entry of integrand can be expressed specifically as :

$$g_{ij}(\xi,\eta,|J|) = \frac{d_0 + d_1\xi + d_2\eta + d_3\xi^2 + d_4\xi\eta + d_5\eta^2 + d_6\xi^2\eta + d_7\xi\eta^2 + d_8\xi^2\eta^2}{|J|} \tag{4.47}$$

Where $d_1, d_2, \ldots, d_8$ are constants, too.

The Jacobian J is

$$J = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{vmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \tag{4.48}$$

And the global coordinate variables x, y can be transformed to local coordinate by the same shape functions as those of field variables. This is an intrinsic property of an isoparametric element.

$$x = \sum_{i=1}^{4} N_i x_i \quad , \qquad y = \sum_{i=1}^{4} N_i y_i \tag{4.49}$$

Where

• x_i, y_i : are global node coordinates.

• N_i : shape functions.

Specifically, the individual shape functions are

$$N_1 = \tfrac{1}{4}(1-\xi)(1-\eta) \tag{4.50}$$

$$N_2 = \tfrac{1}{4}(1+\xi)(1-\eta) \tag{4.51}$$

$$N_3 = \tfrac{1}{4}(1+\xi)(1+\eta) \tag{4.52}$$

$$N_4 = \tfrac{1}{4}(1-\xi)(1+\eta) \tag{4.53}$$

and the entries of Jacobian are derived from equations (4.48) to (4.53).

$$J_{11} = \frac{\partial x}{\partial \xi} = \tfrac{\eta}{4}(x_1 - x_2 + x_3 - x_4) + \tfrac{1}{4}(-x_1 + x_2 + x_3 - x_4) = a_x\eta + b_x \tag{4.54}$$

$$J_{12} = \frac{\partial y}{\partial \xi} = \tfrac{\eta}{4}(y_1 - y_2 + y_3 - y_4) + \tfrac{1}{4}(-y_1 + y_2 + y_3 - y_4) = a_y\eta + b_y \tag{4.55}$$

$$J_{21} = \frac{\partial x}{\partial \eta} = \tfrac{\xi}{4}(x_1 - x_2 + x_3 - x_4) + \tfrac{1}{4}(-x_1 - x_2 + x_3 + x_4) = a_x\xi + c_x \tag{4.56}$$

$$J_{22} = \frac{\partial x}{\partial \xi} = \tfrac{\xi}{4}(y_1 - y_2 + y_3 - y_4) + \tfrac{1}{4}(-y_1 - y_2 + y_3 + y_4) = a_y\xi + c_y \tag{4.57}$$

Therefore, the determinant of the Jacobian will be

$$\begin{aligned} |J| &= J_{11}J_{22} - J_{12}J_{21} \\ &= (b_x a_y - a_x b_y)\xi + (a_x c_y - a_y c_x)\eta + (b_x c_y + b_y c_x) \\ &= U\xi + V\eta + W \end{aligned} \tag{4.58}$$

where U, V, W are independent of natural coordinates.

Obviously, the determinant of the Jacobian is only a linear function of natural coordinates. This linearity allows the exact integration to be performed and the logarithm function is expected in the solution. As the first integration with respect to x is done, there is no longer an integration variables h in the denominator. Therefore the second integration with respect to h can also be performed analytically. However, the algebraic operations to finish these two integrations are too tedious to be handled by hand. Fortunately, with the help of the symbolic and algebraic manipulator REDUCE, these mathematical operations can be done by computer. In addition, the solutions can be organized in a systematic way and converted into a FORTRAN-code subroutine to be called by the main program. All of these procedures and parts of solutions are demonstrated in the next paragraphs by an example of linear elasticity. The explanations of commands and the time consumed in each individual command are also

commented for reference. The total time consumed in this execution by REDUCE is around two and half hours. The resultant fortran expressions for just an element of stiffness matrix occupy almost sixteen pages. These huge expressions are the reason why the closed form solution was not available before.

4.7.3 REDUCE program and explanation

```
%-------------------------------------
% Turning on the CPU elapse time.
%-------------------------------------
1: on time;
Time: 133 ms
%-------------------------------------
% Inputting 4 shape functions.
% p and q are natural coordinates.
% p : xi
% q : eta
%-------------------------------------
2: s1:=(1-p)*(1-q)/4;

       P*Q - P - Q + 1
S1 := ------------------
             4
Time: 600 ms

3: s2:=(1+p)*(1-q)/4;

            P*Q - P + Q - 1
S2 := - ----------------------
                 4
Time: 167 ms

4: s3:=(1+p)*(1+q)/4;

        P*Q + P + Q + 1
S3 := -----------------------
              4
Time: 167 ms

5: s4:=(1-p)*(1+q)/4;

            P*Q + P - Q - 1
S4 := - ----------------------
                 4
Time: 166 ms

%-----------------------------------------
% Expressing x & y by shape function
% and global node coordinates.
%-----------------------------------------
```

```
6: x:=s1*x1+s2*x2+s3*x3+s4*x4;
X := (P*Q*X1-P*Q*X2+P*Q*X3-P*Q*X4-P*X1+P*X2+P*X3-P*X4-
    Q*X1-Q*X2+Q*X3+Q*X4+X1+X2+X3+X4)/4
Time: 400 ms

7: y:=s1*y1+s2*y2+s3*y3+s4*y4;
Y := (P*Q*Y1-P*Q*Y2+P*Q*Y3-P*Q*Y4-P*Y1+P*Y2+P*Y3-P*Y4-
    Q*Y1-Q*Y2+Q*Y3+Q*Y4+Y1+Y2+Y3+Y4)/4
Time: 267 ms

%-------------------------------------------------------------------------------------------------
% Declaring and Inputting matrix elements to calculate the strain-displacement matrix B.
% c     : coefficient matrix
% jac   : combination of Jacobian matrix
% sd    : matrix containing the derivative of shape function.
% b     : strain-displacement matrix.
% detj  : determinant of Jacobian.
% j11,j12,j21,j22       : element of Jacobian matrix.
%-------------------------------------------------------------------------------------------------
8: matrix c(3,4),jac(4,4),sd(4,8),b(3,8)$
Time: 550 ms

9: c:=mat((1,0,0,0),(0,0,0,1),(0,1,1,0))$
Time: 183 ms

10: jac:=mat((j22,-j12,0,0),(-j21,j11,0,0),
        (0,0,j22,-j12),(0,0,-j21,j11))$
Time: 284 ms

11: sd:=mat((df(s1,p),0,df(s2,p),0,df(s3,p),0,df(s4,p),0),
        (df(s1,q),0,df(s2,q),0,df(s3,q),0,df(s4,q),0),
        (0,df(s1,p),0,df(s2,p),0,df(s3,p),0,df(s4,p)),
        (0,df(s1,q),0,df(s2,q),0,df(s3,q),0,df(s4,q)))$
Time: 833 ms

12: b:=c*jac*sd/detj$
Time: 417 ms

%-----------------------------------------------------------
% Inputting material matrix D and calculating integrand.
% D  :  material matrix (assumed symmetric).
% Ga :  integrand.
% th :  thickness of element.
%-----------------------------------------------------------
13: matrix d(3,3),ga(8,8);
Time: 316 ms

14: d:=mat((e11,e12,e13),(e12,e22,e23),(e13,e23,e33))$
Time: 200 ms

15: ga:=tp(b)*d*b*th*detj$
Time: 8067 ms
```

```
%-----------------------------------------
% Evaluating each element of Jacobian.
%-----------------------------------------
16: on factor;
Time: 150 ms

17: on div;
Time: 33 ms

18: on rat;
Time: 50 ms

19: j11:=df(x,p);
          1
J11 := ---*((X1 - X2 + X3 - X4)*Q - X1 + X2 + X3 - X4)
          4
Time: 750 ms

20: j12:=df(y,p);
          1
J12 := ---*((Y1 - Y2 + Y3 - Y4)*Q - Y1 + Y2 + Y3 - Y4)
          4
Time: 550 ms

21: j21:=df(x,q);
            1
J21 := - ---*((X1 + X2 - X3 - X4) - (X1 - X2 + X3 - X4)*P)
            4
Time: 667 ms

22: j22:=df(y,q);
            1
J22 := - ---*((Y1 + Y2 - Y3 - Y4) - (Y1 - Y2 + Y3 - Y4)*P)
            4
Time: 650 ms

%-------------------------------------------------------
%Making substitution for Jacobian matrix.
% ax=(x1-x2+x3-x4)/4,  bx=(-x1+x2+x3-x4)/4
% ay=(y1-y2+y3-y4)/4,  by=(-y1+y2+y3-y4)/4
% cx=(-x1-x2+x3+x4)/4, cy=(-y1-y2+y3+y4)/4
%-------------------------------------------------------
23: j11:=ax*q+bx;
J11 := AX*Q + BX
Time: 333 ms

24: j12:=ay*q+by;
J12 := AY*Q + BY
Time: 117 ms

25: j21:=ax*p+cx;
J21 := AX*P + CX
Time: 117 ms
```

```
26: j22:=ay*p+cy;
J22 := AY*P + CY
Time: 116 ms

%-----------------------------------------
%Calculating the determinant of Jacobian.
%-----------------------------------------
27: matrix j(2,2),ske(8,8);
Time: 200 ms

28: j:=mat((j11,j12),(j21,j22));
J(1,1) := AX*Q + BX
J(1,2) := AY*Q + BY
J(2,1) := AX*P + CX
J(2,2) := AY*P + CY
Time: 367 ms

29: detj:=det(j);
DETJ :=  - ((AX*P + CX)*(AY*Q + BY) - (AX*Q + BX)*(AY*P + CY))
Time: 333 ms

%-------------------------------------------------------------------------
%Making a further substitution and giving the lineality of determinant.
%-------------------------------------------------------------------------
30: let -ax*cy+ay*cx=a1,ax*by-ay*bx=a2,-bx*cy+by*cx=a3;
Time: 450 ms

31: detj:=detj;
DETJ :=  - (A1*Q + A2*P + A3)
Time: 450 ms

32: on exp;
Time: 50 ms

%-----------------------------------------------------------------------------------------------------
%Performing the double integration.
%Due to the complication of the expression in the integrand, it is necessary to make the "pre-
%treatment" to each element of integrand before integration. This is a vital step to avoid the
%limitation of memory space and finish the job.
%-----------------------------------------------------------------------------------------------------
33: for i:=1:8 do for j:=1:8 do
 if j>=i then
       <<cp:=coeff(ga(i,j),p);
         p0:=(first cp);
         p1:=(second cp);
         p2:=(third cp);
         low:=den(ga(i,j));
         ga(i,j):=(d0+d1*p+d2*p**2)/low;
         c1:=int(ga(i,j),p);
         c2:=sub(p=1,c1)-sub(p=-1,c1);
         c3:=sub(d0=p0*low,d1=p1*low,d2=p2*low,c2);
         c4:=sub(log(a1*q-a2+a3)=mlog,log(a1*q+a2+a3)=plog,c3);
         on exp;
```

```
        cq:=coeff(c4,q);
        q0:=(first cq);
        q1:=(second cq);
        q2:=(third cq);
        k01:=lcof(num(q0),mlog)/den(q0);
        q0:=reduct(num(q0),mlog)/den(q0);
        k02:=lcof(num(q0),plog)/den(q0);
        k03:=reduct(num(q0),plog)/den(q0);
        k04:=lcof(num(q1),mlog)/den(q1);
        q1:=reduct(num(q1),mlog)/den(q1);
        k05:=lcof(num(q1),plog)/den(q1);
        k06:=reduct(num(q1),plog)/den(q1);
        k07:=lcof(num(q2),mlog)/den(q2);
        q2:=reduct(num(q2),mlog)/den(q2);
        k08:=lcof(num(q2),plog)/den(q2);
        k09:=reduct(num(q2),plog)/den(q2);
        c4:= d01*log(a1*q-a2+a3)+d02*log(a1*q+a2+a3)+d03
          +d04*q*log(a1*q-a2+a3)+d05*q*log(a1*q+a2+a3)+d06*q
          +d07*q**2*log(a1*q-a2+a3)+d08*q**2*log(a1*q+a2+a3)
          +d09*q**2;
        c5:=int(c4,q);
        c6:=sub(q=1,c5)-sub(q=-1,c5);
        let log(a1**2-a1*a2+a1*a3)=h1,log(-a1**2-a1*a2+a1*a3)=h2,
            log(a1+a2+a3)=h3,log(-a1+a2+a3)=h4,log(a1-a2+a3)=h5,
            log(-a1-a2+a3)=h6;
        factor h1,h2,h3,h4,h5,h6;
        ske(i,j):=sub(d01=k01,d02=k02,d03=k03,d04=k04,d05=k05
                    ,d06=k06,d07=k07,d08=k08,d09=k09,c6)>>
      else ske(i,j):=ske(j,i);
Time: 234766 ms

%-----------------------------------------------------------------------------------------------------------
%Showing the results for the element in 1st row and 1st column of the local stiffness matrix.
%H1=log(a1**2-a1*a2+a1*a3),      H2=log(-a1**2-a1*a2+a1*a3)
%H3=log(a1+a2+a3)       ,       H4=log(-a1+a2+a3)
%H5=log(a1-a2+a3)       ,       H6=log(-a1-a2+a3)
%-----------------------------------------------------------------------------------------------------------
34: ske:=ske;

                          1     -1              1     -1     -1     2
SKE(1,1) := H1*TH*(---*A1  *A2*E13 - ---*A1  *A2  *A3  *E13 +
                          8                8

              1     -1    -1          2          1     -1     -1
            ----*A1  *A2  *A3*BX  *E33 - ---*A1  *A2  *A3*BX*BY*E13
            16                               8

                                    1     -2    2          1
               .....................+ ----*A2  *A3  *E13 + ----*E13)
                                    16                     16

                        1      2    -2           1      2    -3       2
             + H6*TH*( ----*A1  *A2  *E13 + ----*A1  *A2  *AX  *E33 -
                       16                   48
```

```
                                    1                    1        -1          1     -1
.....+ ----*E13) + TH*( - ---*A1*A2  *E13 + ---*A1  *A2*E13 +
                                   16                   4                    4

 1     -2                                1     -2         2
- ---*A2  *A3*BX*BY*E13 + ---*A2  *A3*BY *E11)
 3                                       6
Time: 79317 ms
```

4.7.4 Fortran subroutine from REDUCE

```
%-----------------------------------------------
%Converting the results into FORTRAN code
%-----------------------------------------------
      subroutine(x1,y1,x2,y2,x3,y3,x4,y4,e1,e2,e3,e4,e5,e6,th,ske)
     implicit real*8(a-h,o-z)
     dimension ske(8,8)
     ax=(x1-x2+x3-x4)/4.
     bx=(-x1+x2+x3-x4)/4.
     ay=(y1-y2+y3-y4)/4.
     by=(-y1+y2+y3-y4)/4.
     cx=(-x1-x2+x3+x4)/4.
     cy=(-y1-y2+y3+y4)/4.
     a1=-ax*cy+ay*cx
     a2=ax*by-ay*bx
     a3=-bx*cy+by*cx
     h1=log(a1**2-a1*a2+a1*a3)
     h2=log(-a1**2-a1*a2+a1*a3)
     h3=log(a1+a2+a3)
     h4=log(-a1+a2+a3)
     h5=log(a1-a2+a3)
     h6=log(-a1-a2+a3)
     ANS14=1/4*A1**(-3)*A3**2*AY*CX*E13-1/8*A1**(-3)*A3**2
    . *AY*CY*E11-1/16*A1**(-3)*A3**2*CX**2*E33+1/8*A1**(-3
    . )*A3**2*CX*CY*E13-1/16*A1**(-3)*A3**2*CY**2*E11-1/16
    . *A2**(-2)*A3**2*E13+1/16*E13
          .
          .
     ANS1=H1*TH*ANS2
     ANS28=-1/4*A1**(-3)*A3**2*AY*CX*E13+1/8*A1**(-3)*A3**
    . 2*AY*CY*E11+1/16*A1**(-3)*A3**2*CX**2*E33-1/8*A1**(
    . -3)*A3**2*CX*CY*E13+1/16*A1**(-3)*A3**2*CY**2*E11+1/
    . 16*A2**(-2)*A3**2*E13-1/16*E13
          .
          .
     ANS15=H2*TH*ANS16
     ANS48=1/8*A2**(-3)*A3**2*BX*BY*E13-1/16*A2**(-3)*A3**
    . 2*BY**2*E11
          .
          .
     ANS29=H3*TH*ANS30
     ANS68=-1/16*A2**(-3)*A3**2*BX**2*E33+1/8*A2**(-3)*A3
```

```
. **2*BX*BY*E13-1/16*A2**(-3)*A3**2*BY**2*E11
    .
    .
ANS49=H4*TH*ANS50
ANS75=1/16*A2**(-3)*A3**2*AX**2*E33-1/8*A2**(-3)*A3**
. 2*AX*AY*E13+1/8*A2**(-3)*A3**2*AX*BX*E33+1/16*A2**(
. -3)*A3**2*AY**2*E11-1/4*A2**(-3)*A3**2*AY*BX*E13+1/8
. *A2**(-3)*A3**2*AY*BY*E11+1/16*A2**(-3)*A3**2*BX**2*
. E33-1/8*A2**(-3)*A3**2*BX*BY*E13+1/16*A2**(-3)*A3**2
. *BY**2*E11-1/16*E13
    .
    .
ANS69=H5*TH*ANS70
ANS82=-3/16*A2**(-2)*A3*BY*CY*E11+1/16*A2**(-3)*A3**2
. *AX**2*E33-1/8*A2**(-3)*A3**2*AX*AY*E13+1/8*A2**(-3)
. *A3**2*AX*BX*E33+1/16*A2**(-3)*A3**2*AY**2*E11-1/4*
. A2**(-3)*A3**2*AY*BX*E13+1/8*A2**(-3)*A3**2*AY*BY*
. E11+1/16*A2**(-3)*A3**2*BX**2*E33-1/8*A2**(-3)*A3**2
. *BX*BY*E13+1/16*A2**(-3)*A3**2*BY**2*E11+1/16*E13
    .
    .
ANS83=TH*ANS84
ske(1,1)=ANS1+ANS15+ANS29+ANS49+ANS69+ANS76+ANS83
    .
    .
return
end
```

4.8 Significance and conclusion

Some conclusions are drawn and the significance of automatic problem formulation is discussed as follows :

1. Improving on-line efficiency --- the closed-form solution of the local stiffness matrix allows us to get a numerical value by simply substituting the global nodal coordinates into a FORTRAN-code subroutine. This procedure is done in just one step. Of course, this is faster than the Gauss quadrature rule which usually needs more than one integration point to get a reasonable solution[4] . The symbolic template in the nonlinear numerical analysis also plays the same role. In the case of a large number of elements (or large dimension size in matrix), the significance in improving on-line efficiency will be greater.

2. Increasing the accuracy of solution --- the closed-form solution is an exact solution. There is no integration error introduced into the evaluation of the stiffness matrix. The

[4] Reduce integration is an exception and sometimes results in Hourglass drawback. This special case is ruled out here.

accuracy of the Fourier series approximation can be increased as high as the user requires. Therefore, the results from using symbolic and algebraic manipulation will be more precise than those of pure numerical analysis.

3. Free from hand-calculation and typing error --- As the results in the tensor formulation, derivation of equations of motion and stiffness matrix constructions show, the algebraic expressions are too lengthy to be formulated by hand. Even supposing that they could be done by hand, it would be so tedious that nobody could guarantee that no mistakes would be made when trying to key them into the computer. With the use of symbolic and algebraic manipulation, both difficulties are completely solved. As long as the user inputs the correct commands, there will not be any question about the correctness of the results.

4. Simplifying FORTRAN programming --- the numerical values of the local stiffness matrix can be obtained by simply using the "CALL" command once. This isn't true when Gauss quadrature integration is employed in the finite element method to evaluate integration. It is necessary to have a "DO" loop, "CALL" command and multiplications of various weight coefficients for different integration points. These will complicate the program and therefore will produce more error sources.

5. Further analysis of symbolic results becomes available --- Sometimes the pre-analysis of the expressions produced from symbolic and algebraic manipulation will lead to a dramatic improvement in the incoming numerical analysis. The closed form solution makes this analysis feasible. For example, suppose that the diagonal terms of global stiffness matrix need to be more dominant to improve the ill-condition, this can be achieved by appropriately relocating nodes so that the off-diagonal terms of local stiffness matrix will be smaller or even vanished. This is the unique advantage that the numerical method does not possess.

4.9 References

[1] T. R. Kane and D. A. Levinson,"Dynamics : Theory and Applications", McGraw-Hill, 1985.

[2] L. Meirovitch,"Methods of analytical dynamics",1970.

[3] F. I. Niordson,"shell theory",North-Holland, 1985.

[4] I. S. Sokolnikoff,"Tensor analysis", 2nd edition, John Wiley & Sons Inc, 1964.

[5] W. H. Yang,"AM519 course note", Mechanical engineering ,The University of Michigan, Ann Arbor, 1989.

[6] P. V. O'Neil,"Advanced Engineering Mathematics",Wadsworth, 1983.

[7] N. Kikuchi,"finite element methods in mechanics",Cambridge University press, 1986.

[8] A. R. Korncoff and S. J. Fenves,"Symbolic generation of finite element stiffness matrices", Computers & Structures, vol. 10, pp. 119-124, 1979.

[9] R. W. Luft, Jose M. Roesset and J. J. Connor,"Automatic generation of finite element matrices", J. of the structural division, proc. of ASCE, pp 349-362, Jan. 1971.

[10] O. C. Zienkiewicz,"The finite element method", 3rd edition, McGraw-Hill, pp. 191, 1977.

[11] E. B. Becker, G. F. Carey and J. T. Oden,"Finite elements an introduction", vol. 1, Prentice-Hall, 1981

[12] Robert D. Cook,"Concepts and applications of finite element analysis", 2nd edition, pp. 119, John Wiley & Sons Inc., 1974.

[13] J. N. Reddy,"An introduction to the finite element method", pp.154, McGraw-Hill, 1984.

[14] K. H. Huebner,"The finite element method for engineers", pp.190, John Wiley & Sons, 1975.

[15] W. Weaver, Jr and Paul R. Johnson,"Finite elements for structural analysis, pp. 111, Prentice-Hall Inc. ,1984.

[16] A. C. Hearn," REDUCE user's manual", Version 3.3, July 1987.

[17] W. L. Tsai, "Applications of symbolic and algebraic manipulation software in solving applied mechanics problems", Ph. D. thesis, Department of mechanical engineering and applied mechanics, The University of Michigan, Ann Arbor, Dec. 1989.

CHAPTER V

APPLICATION OF SYMBOLIC AND ALGEBRAIC MANIPULATION TO A MATERIALLY NONLINEAR PROBLEM --- RIGID-PLASTIC RING COMPRESSION

5.1 Introduction

The application of the finite element method to the rigid plastic problem was originally devised by Lee and Kobayashi ([1], [2]), and became popular in the last decade. This method allowed the deformation behavior of metal to be revealed on the computer screen stage by stage during the process of metal forming. As a consequence, the design techniques of die, and the manufacturing process were improved. This contribution to the industrial manufacturing field is recognized to be very significant.

Starting from the principle of virtual work and associating with the normality condition of plasticity, the theoretical analysis of this method leads to an inequality objective function with an equality constraint. This equality constrained problem is then changed into an unconstrained problem by introducing Lagrange multipliers. As the stationary requirement is reached, the total unconstrained problem can be solved incrementally by the finite element method and the upper bound solution will satisfy the equilibrium equations, constitutive equations, compatibility equations, incompressibility constraint, and boundary conditions.

Despite the success of the finite element application to the problem, the formulation of it is very tedious. Especially when the friction boundary condition is considered, the performance of integration along the interface surface and the evaluation of the first and second derivatives with respect to velocity fields is always difficult to obtain by hand. Therefore, unlike the traditional hand derivation, this chapter will utilize the symbolic and algebraic manipulator REDUCE to do the job of formulation. The quantities involved in the formulae then can be made into subroutines symbolically at the the element level to facilitate the global assemblage. As a result, otherwise intractable tasks become possible and free of errors.

5.2 Preliminary formulation

Consider a body of volume V with the essential boundary condition , velocity $\vec{U}$, prescribed on surface S_u and two separated natural boundary conditions , traction $\vec{F}$ and frictional stress $\vec{f}$, prescribed on S_F and S_c, respectively [Figure 5.1]. The actual stress and velocity fields will satisfy the following relationships :

1. Equilibrium conditions :

$$\sigma_{ij,j} = 0 \qquad (neglecting\ \ body\ \ force\ ) \tag{5.1}$$

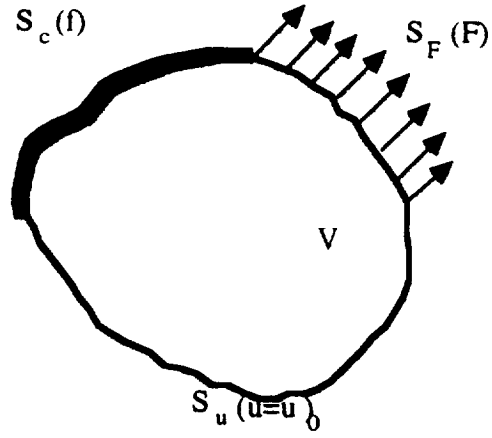

Figure 5.1: Configuration of domain and boundary conditions

2. Compatibility and incompressibility conditions:

$$\dot{\varepsilon}_{ij} = \tfrac{1}{2}(u_{i,j} + u_{j,i}) \tag{5.2}$$

$$\dot{\varepsilon}_{ii} = 0 \tag{5.3}$$

3. Stress-strain rate relationship

$$\sigma'_{ij} = \frac{2}{3}\frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}}\dot{\varepsilon}_{ij} \tag{5.4}$$

Where

- σ'_{ij} : deviatoric stress.
- $\bar{\sigma}$ and $\dot{\bar{\varepsilon}}$: effective stress and strain-rate, respectively.

4. Boundary conditions :

$$\sigma_{ij} n_i = F_j \qquad on \qquad S_F \tag{5.5}$$

$$u_i = U_i \qquad on \qquad S_u \tag{5.6}$$

$$\sigma_{ij} n_i = f_j(\vec{V}_r) \qquad on \qquad S_c \tag{5.7}$$

Where the relative velocity between die and deforming body is defined as follows :

$$\vec{V}_r = \vec{u} - \vec{u}^d = V_r \tilde{t} \tag{5.8}$$

The $\tilde{t}$ here is the unit base vector along die and working piece interface.

With an admissible velocity field $\vec{u}^*$, the virtual work principle gives

$$\int_V \sigma_{ij} \dot{\varepsilon}^*_{ij} dV = \int_{S_F} \vec{F} \cdot \vec{u}^* dS + \int_{S_c} \tilde{f} \cdot (\vec{V}^*_r + \vec{u}^d) dS + \int_{S_u} (\sigma_{ij} n_i) U_j dS \tag{5.9}$$

The frictional stress $\tilde{f}$ is defined as follows and is plotted in Figure 5.2.

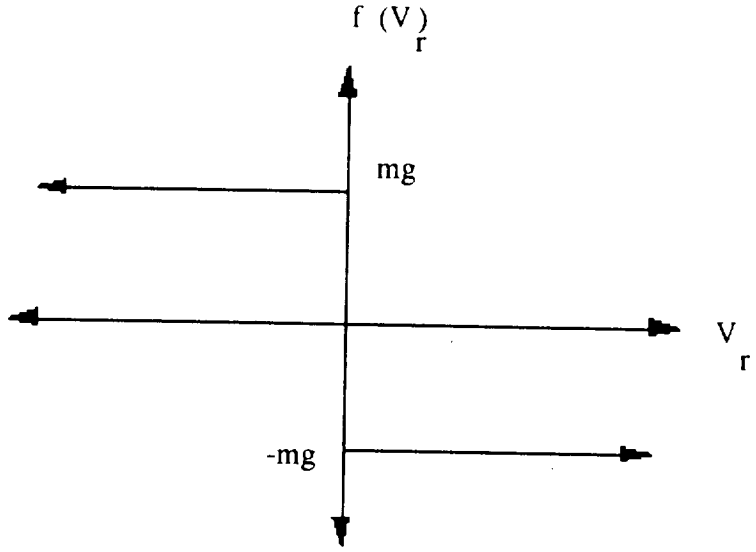

Figure 5.2 : Plot of frictional stress vs. relative velocity

$$\tilde{f}(\vec{V}_r) = -m * g * \frac{V_r}{|\vec{V}_r|} \tilde{t} \tag{5.10}$$

where the friction factor m is in the range of $0 \le m \le 1$ and g is the yield shear stress. By considering the normality condition of yield surface

$$(\sigma^*_{ij} - \sigma_{ij}) \dot{\varepsilon}^*_{ij} \ge 0 \tag{5.11}$$

and inequality equation[5]

$$\tilde{f}(\vec{V}_r) \cdot \vec{V}_r^* - \tilde{f}(\vec{V}_r^*) \cdot \vec{V}_r^* \geq 0 \tag{5.12}$$

The equation (5.9) becomes

$$\int_V \sigma_{ij}^* \dot{\varepsilon}_{ij}^* dV - \int_{S_F} \vec{F} \cdot \vec{u}^* dS - \int_{S_c} \tilde{f}(\vec{V}_r^*) \cdot \vec{V}_r^* dS \geq \int_{S_u} (\sigma_{ij} n_i) U_j dS + \int_{S_c} \tilde{f}(\vec{V}_r) \cdot \vec{u}^d dS \tag{5.13}$$

All conditions will be met when the left hand side of equation (5.13) reaches the minimum values with respect to $\vec{u}^*$ and satisfies the incompressibility condition of equation (5.3). Since there is no ambiguity in omitting the asterisk, for the sake of simplicity the admissible field $\vec{u}^*$ will be denoted as u for the following discussions without any special note. By introducing the Lagrange multiplier l, the equality constraint equation can be included into the objective function. The stationary value problem for finite element formulation is therefore

$$\frac{\partial \Psi}{\partial \vec{u}} = \frac{\partial}{\partial \vec{u}} [\int_V \bar{\sigma} \vec{\varepsilon} dV - \int_{S_F} \vec{F} \cdot \vec{u} dS - \int_{S_c} \tilde{f}(\vec{V}_r) \cdot \vec{V}_r dS + \int_V \lambda \dot{\varepsilon}_v dV] = 0 \tag{5.14}$$

where Ψ represents the functional inside the square bracket. Since the frictional stress is not differentiable at the point $\vec{V}_r = 0$ [see Figure 5.2], $\frac{\partial \Psi}{\partial \vec{u}}$ does not exist and the convergent solution will not be available for equation (5.14). In order to overcome this numerical difficulty, the frictional stress is approximated in terms of arctangent function [see Figure 5.3]

$$\tilde{f}(\vec{V}_r) = -m * g * [(\frac{2}{\pi}) \tan^{-1}(\frac{\vec{V}_r}{a|\vec{u}^d|})] \tilde{t} \tag{5.15}$$

where the arbitrary constant a is several orders less than the die velocity. Its function is to exaggerate the argument of arctangent to reduce the error between equation (5.15) and equation (5.10). The equation (5.15) is absorbed into functional by the following inequality[6]

[5] See appendix A for proof.
[6] See appendix B for proof.

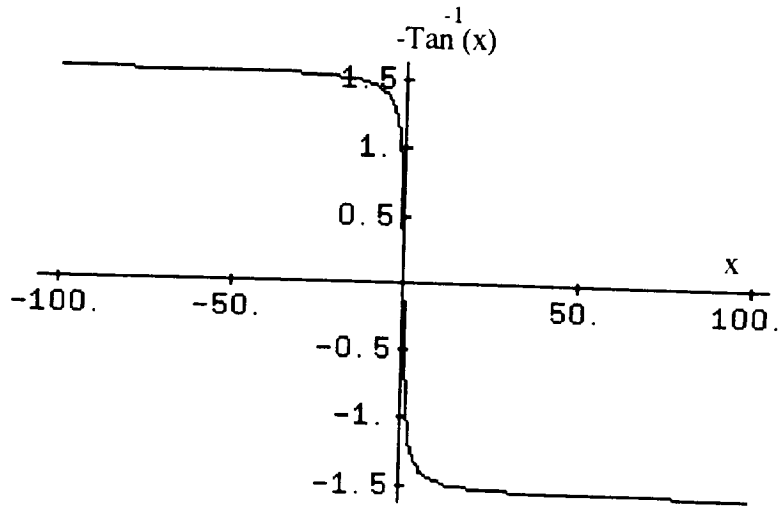

Figure 5.3 : Arctangent approximation of frictional stress

$$\int_0^{|\vec{v}_r^*|} \vec{f} \cdot d\vec{v}_r - \int_0^{|\vec{v}_r|} \vec{f} \cdot d\vec{v}_r \le \vec{f}(\vec{v}_r) \cdot (\vec{v}_r^* - \vec{v}_r) \qquad (5.16)$$

The final form of equation (5.13) is

$$\frac{\partial \Psi}{\partial \vec{u}} = \frac{\partial}{\partial \vec{u}} [\int_v \vec{\sigma}\vec{\dot{\varepsilon}}\, dV - \int_{S_F} \vec{F} \cdot \vec{u}\, dS - \int_{S_c} (\int_0^{|\vec{v}_r|} \vec{f}\, d\vec{v}_r) dS + \int_v \lambda \dot{\varepsilon}_v dV\ ] = 0 \quad (5.17)$$

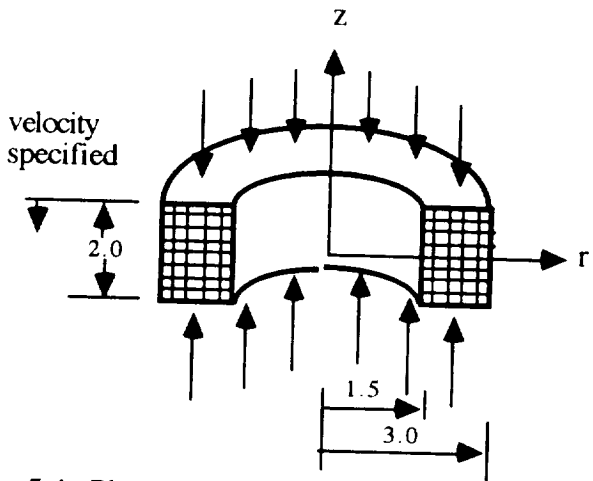

Figure 5.4 : Physical configuration for ring compression problem

5.3 Matrix method for the ring compression problem

If a specific problem of ring compression is considered [see Figure 5.4], the surface traction is due to friction force only. Therefore the term for S_F integration in equation (5.17) can be dropped. By the substitution of the following equations to equation (5.17),

$$\dot{\varepsilon} = \begin{Bmatrix} \dot{\varepsilon}_r \\ \dot{\varepsilon}_z \\ \dot{\varepsilon}_\theta \\ \dot{\gamma}_{rz} \end{Bmatrix} = \begin{bmatrix} \frac{\partial}{\partial r} & 0 \\ 0 & \frac{\partial}{\partial z} \\ \frac{1}{r} & 0 \\ \frac{\partial}{\partial z} & \frac{\partial}{\partial r} \end{bmatrix} \begin{Bmatrix} u \\ v \end{Bmatrix} = B * U \tag{5.18}$$

$$\bar{\dot{\varepsilon}} = \sqrt{\frac{2}{3}\dot{\varepsilon}_{ij}\,\dot{\varepsilon}_{ij}} \tag{5.19}$$

the stationary requirement of the functional results in the following nonlinear equation.

$$\int_v \left(\frac{2 * \bar{\sigma}}{\bar{\dot{\varepsilon}}}\right) K * \tilde{u}\, dV + \int_v U^T * K * U * \frac{\partial\left(\frac{\bar{\sigma}}{\bar{\varepsilon}}\right)}{\partial U} dV + \lambda * Q = -\frac{\partial}{\partial U}\left(\int_S \int_0^{|\nabla_r|} \bar{F} dV_r\right) dS \tag{5.20}$$

together with an incompressibility constraint equation

$$U^T * Q = 0 \tag{5.21}$$

where

- $K = B^T * D * B$
- $Q = \int_v B^T * C\, dV$
- C : proper matrix such that implies the $C^T \dot{\varepsilon}$ incompressibility condition.
- D : flow matrix.
- U: vector of velocity. It is represented as $\tilde{u}$ previously.

The derivative part in the second term of equation (5.20) is equal to

$$\frac{\partial(\frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}})}{\partial U} = \frac{1}{\dot{\bar{\varepsilon}}}\frac{\partial \bar{\sigma}}{\partial U} - \frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}^2}\frac{\partial \dot{\bar{\varepsilon}}}{\partial U} = -\frac{Y}{\dot{\bar{\varepsilon}}^2}\frac{\partial \dot{\bar{\varepsilon}}}{\partial U} \tag{5.22}$$

The first term of (5.22) is dropped for simplicity due to the assumption that $\bar{\sigma}$ is constant for the infinitesimal strain change [see References 2]. Physically, $\bar{\sigma}$ is the current yield strength (denoted as Y) of the material.

The equations (5.20) and (5.21) are then perturbed by introducing a small velocity ΔU into the velocity vector. This gives the following equations :

$$\begin{aligned}
&\int_v [\frac{2\bar{\sigma}}{\dot{\bar{\varepsilon}}} + \frac{\partial}{\partial U}(\frac{2\bar{\sigma}}{\dot{\bar{\varepsilon}}})\Delta U\,] * K * (U + \Delta U)dV + \lambda * Q \\
&+ \int_v (U + \Delta U)^T * K * (U + \Delta U)[\frac{\partial}{\partial U}(\frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}}) + \frac{\partial^2}{\partial U^2}(\frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}})\Delta U\,]dV \\
&= -[\frac{\partial \Phi}{\partial U} + \frac{\partial^2 \Phi}{\partial U^2}\Delta U\,]
\end{aligned} \tag{5.23}$$

and

$$(U + \Delta U)^T * Q = 0 \tag{5.24}$$

where

$$\Phi = -\int_S (\int_0^{\dot{\bar{\varepsilon}}} \tilde{F}\, d\dot{\bar{\varepsilon}})dS$$

is contributed by the friction traction. After neglecting the second and higher order terms, the equations (5.23) and (5.24) can be combined into the following forms :

$$\left(\begin{bmatrix} P & Q \\ Q^T & 0 \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 \Phi}{\partial U^2} & 0 \\ 0 & 0 \end{bmatrix}\right)_{9*9} \begin{Bmatrix} \Delta U \\ \frac{\lambda}{Y} \end{Bmatrix}_{9*1} = -\begin{Bmatrix} H \\ Q^T U \end{Bmatrix}_{9*1} - \frac{1}{Y}\begin{Bmatrix} \frac{\partial \Phi}{\partial U} \\ 0 \end{Bmatrix}_{9*1} \tag{5.25}$$

where

$$P_{8*8} = 2\int_v \left\{\frac{1}{\dot{\bar{\varepsilon}}}K + [M - (\frac{1}{\dot{\bar{\varepsilon}}})^2 N\,]\right\} \tag{5.26}$$

$$H_{8*8} = \int_v \left\{\frac{2}{\dot{\bar{\varepsilon}}}K * U - \frac{k}{\dot{\bar{\varepsilon}}^2}E\right\} \tag{5.27}$$

$$\dot{\bar{\varepsilon}} = \frac{2}{3}\sqrt{\dot{\varepsilon}_r^2 + \dot{\varepsilon}_z^2 + \dot{\varepsilon}_\theta^2 - \dot{\varepsilon}_r\dot{\varepsilon}_z - \dot{\varepsilon}_z\dot{\varepsilon}_\theta - \dot{\varepsilon}_r\dot{\varepsilon}_\theta + \frac{3}{4}\dot{\gamma}_{rz}^2} \tag{5.28}$$

$$E_{8*1} = B^T * A \tag{5.29}$$

$$Q_{8*1} = \int_v B^T C \, dV \tag{5.30}$$

$$A_{4*1} = [\frac{\partial \dot{\bar{\varepsilon}}}{\partial \dot{\varepsilon}}]_{4*1} = \frac{2}{9\dot{\varepsilon}} \begin{Bmatrix} 2\dot{\varepsilon}_r - \dot{\varepsilon}_z - \dot{\varepsilon}_\theta \\ 2\dot{\varepsilon}_z - \dot{\varepsilon}_\theta - \dot{\varepsilon}_r \\ 2\dot{\varepsilon}_\theta - \dot{\varepsilon}_r - \dot{\varepsilon}_z \\ \frac{3}{2}\dot{\gamma}_{rz} \end{Bmatrix} \tag{5.31}$$

$$K_{8*8} = B^T * D * B \tag{5.32}$$

$$M_{8*8} = (\frac{1}{\dot{\varepsilon}})^2[\frac{k}{\dot{\varepsilon}}E - K * U]E^T \tag{5.33}$$

$$\lambda = \frac{1}{3}C^T \dot{\varepsilon} \tag{5.34}$$

$$N_{8*8} = E * U^T * K \tag{5.35}$$

$$k = U^T * K * U \tag{5.36}$$

$$\Phi = -\int_{S_c}[\int_0^{V_r} m * g * (\frac{2}{\pi})\tan^{-1}(\frac{V_r}{a \cdot |\mu^d|})dV_r]dS \tag{5.37}$$

m, a g are constants for a specific material.

5.4 Finite element analysis

The domain discretization for ring cross section is shown in Figure 5.5. The actual domain used by finite element analysis is only the upper half of that given in Figure 5.5 due to the symmetric geometry.

The finite element model contains 96 four-node quadrilateral elements with 117 nodes in total. For an isoparametric element, the shape functions are

$$q_1 = \frac{1}{4}(1-s)(1-t) \quad , \quad q_2 = \frac{1}{4}(1+s)(1-t)$$

$$q_3 = \frac{1}{4}(1+s)(1+t) \quad , \quad q_4 = \frac{1}{4}(1-s)(1+t)$$

and

$$u(s,t) = \sum_{i=1}^{4} q_i * u_i \quad , \quad v(s,t) = \sum_{i=1}^{4} q_i * v_i$$

$$r(s,t) = \sum_{i=1}^{4} q_i * r_i \quad , \quad z(s,t) = \sum_{i=1}^{4} q_i * z_i$$

Where

- s, t are natural coordinates.
- r, z are physical coordinates.
- u, v are the velocity components in the r, z direction, respectively.

The subscripts in r, z, u, v represent the nodal indices.

The strain rate in axisymmetric case can be expressed in a matrix form as follows :

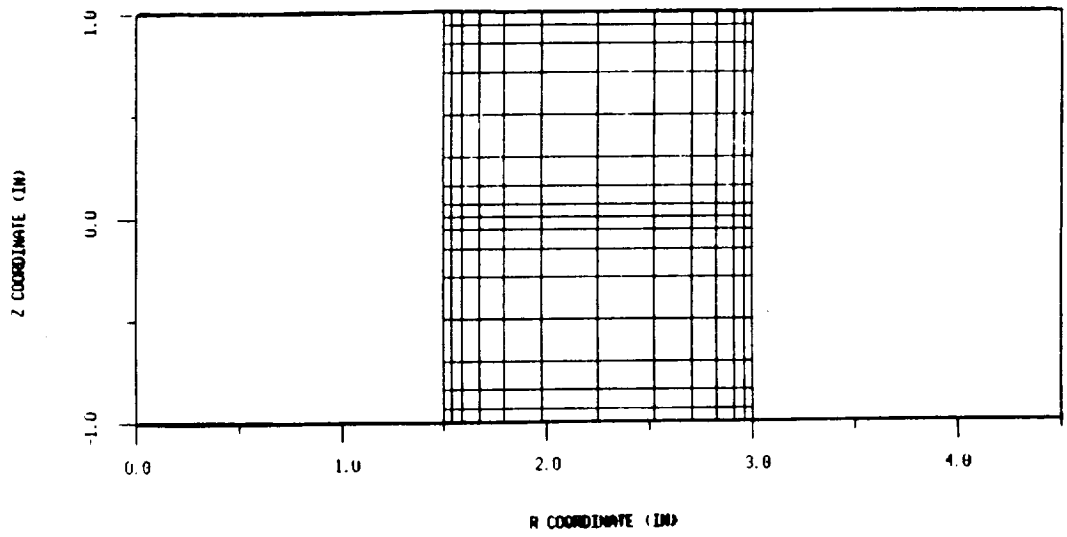

Figure 5.5 : Discretization of ring cross section. Due to the symmetry of geometry, only the upper half of this cross section is used for finite element analysis.

$$\dot{\varepsilon} = \begin{Bmatrix} \dot{\varepsilon}_r \\ \dot{\varepsilon}_z \\ \dot{\varepsilon}_\theta \\ \dot{\gamma}_{rz} \end{Bmatrix} = \begin{bmatrix} \frac{\partial}{\partial r} & 0 \\ 0 & \frac{\partial}{\partial z} \\ \frac{1}{r} & 0 \\ \frac{\partial}{\partial z} & \frac{\partial}{\partial r} \end{bmatrix} \begin{Bmatrix} u \\ v \end{Bmatrix} = B * U \tag{5.38}$$

Where

$$U^T = \{u_1 \quad u_2 \quad u_3 \quad u_4 \quad u_5 \quad u_6 \quad u_7 \quad u_8\} \tag{5.39}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{r} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} q_1 & 0 & q_2 & 0 & q_3 & 0 & q_4 & 0 \\ 0 & q_1 & 0 & q_2 & 0 & q_3 & 0 & q_4 \end{bmatrix} +$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial r}{\partial s} & \frac{\partial z}{\partial s} & 0 & 0 \\ \frac{\partial r}{\partial t} & \frac{\partial z}{\partial t} & 0 & 0 \\ 0 & 0 & \frac{\partial r}{\partial s} & \frac{\partial z}{\partial s} \\ 0 & 0 & \frac{\partial r}{\partial t} & \frac{\partial z}{\partial t} \end{bmatrix}^{-1} \begin{bmatrix} q_{1,s} & 0 & q_{2,s} & 0 & q_{3,s} & 0 & q_{4,s} & 0 \\ q_{1,t} & 0 & q_{2,t} & 0 & q_{3,t} & 0 & q_{4,t} & 0 \\ 0 & q_{1,s} & 0 & q_{2,s} & 0 & q_{3,s} & 0 & q_{4,s} \\ 0 & q_{1,t} & 0 & q_{2,t} & 0 & q_{3,t} & 0 & q_{4,t} \end{bmatrix} \tag{5.40}$$

The Jacobian is :

$$J = \begin{vmatrix} \frac{\partial r}{\partial s} & \frac{\partial z}{\partial s} \\ \frac{\partial r}{\partial t} & \frac{\partial z}{\partial t} \end{vmatrix} \tag{5.41}$$

The flow matrix is ;

$$D = \begin{bmatrix} \frac{2}{3} & 0 & 0 & 0 \\ 0 & \frac{2}{3} & 0 & 0 \\ 0 & 0 & \frac{2}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{3} \end{bmatrix} \tag{5.42}$$

$$C^T = (1 \quad 1 \quad 1 \quad 0) \tag{5.43}$$

Since bilinear velocity distributions are assumed for four-node quadrilateral element, the relative velocities along the interface are interpolated as follows :

$$V_r = \frac{r - r_2}{r_1 - r_2} u_1 + \frac{r_1 - r}{r_1 - r_2} u_2 \tag{5.44}$$

If die velocity is specified as $u^{|dl|}=1$, then the frictional stress will be

$$\tilde{F} = - \{m * g * (\frac{2}{\pi}) \tan^{-1}(\frac{V_r}{a})\} \tilde{t} \tag{5.45}$$

and

$$\Phi = - \int_S [\int_0^{V_r} m * g * (\frac{2}{\pi}) \tan^{-1}(\frac{V_r}{a}) dV_r] dS \tag{5.46}$$

5.5 Application of symbolic manipulation

The difficulty of formulating the equations shown in the last paragraph can be eased by the employment of symbolic and algebraic manipulation. Based on equation (5.25), the original goal is to make a template form in element level for global assemblage. However, due to the limitation of memory capacity in hardware systems, the goal is modified to make a template form for individual entries of equation (5.25) only. There are three kinds of forms produced by REDUCE :

1. Integrable form --- This is the form which results from the fact that integration can be performed analytically by REDUCE. The evaluation of equations (5.30) and (5.37) belongs to this class. There are no natural coordinates in the resultant expressions.

2. Non-integrable form --- The equations (5.26), (5.27) and (5.33), for example, are not integrable due to the existence of $\dot{\bar{\varepsilon}}$, its square and even cubic in the denominator of the integrands. Moreover, since the resultant expression of $\dot{\bar{\varepsilon}}$ occupies more than sixteen pages, the complete integrands of equations (5.26) and (5.27) are not available. Therefore, the individual form for quantities, such as K, M, N, k and E, are obtained symbolically, and the summation as well as their integration are carried out numerically

3. Miscellaneous forms --- Other equations which have nothing to do with integration, such as (5.28), (5.29), (5.31), (5.32), (5.35) and (5.36), are obtained symbolically by matrix operations.

The evaluation of $\frac{\partial \Phi}{\partial U}$ in equation (5.25) requires a number of delicate pre-treatments. It is necessary to discuss this subject independently here so that the fundamental nature of symbolic and algebraic manipulation will be revealed. Intuitively, there are three steps to evaluate $\frac{\partial \Phi}{\partial U}$ They are :

1. Evaluation of the integral with respect to relative velocity V_r in (5.37).

2. Evaluation of the integral with respect to interface domain S in (5.37).

3. The results then are differentiated with respect to velocity fields.

Theoretically, there is nothing wrong with the methodology given above. In fact, REDUCE can only do the first evaluation. The other two are not feasible. Therefore, some pre-treatments are necessary to make REDUCE work to evaluate the $\frac{\partial \Phi}{\partial U}$. This will force us to deviate from the formal methodology given above as follows.

$$\begin{aligned}\frac{\partial \Phi}{\partial u_1} &= \frac{\partial}{\partial u_1}\int_{S_e}[\int_0^{V_r} - m * g * (\tfrac{2}{\pi})\tan^{-1}(\tfrac{V_r}{a})dV_r]dS \\ &= -m * g * (\tfrac{2}{\pi})\int_{S_e}\frac{\partial}{\partial V_r}[\int_0^{V_r}\tan^{-1}(\tfrac{V_r}{a})dV_r]\frac{\partial V_r}{\partial u_1}dS \\ &= -m * g * (\tfrac{2}{\pi})\int_{S_e}[\tan^{-1}(\tfrac{V_r}{a})\frac{\partial V_r}{\partial u_1}]dS \end{aligned} \tag{5.47}$$

The term $\frac{\partial \Phi}{\partial U}$ can be evaluated from equation (5.44). Equation (5.47) becomes

$$\begin{aligned}\frac{\partial \Phi}{\partial u_1} &= -m * g * (\tfrac{2}{\pi})\int_0^{2\pi}\int_{r_1}^{r_2}\tan^{-1}(\tfrac{V_r}{a})\frac{r-r_1}{r_1-r_2}r\,dr\,d\theta \\ &= \frac{-4*m*g}{r_1-r_2}\int_{r_1}^{r_2}[r^2\tan^{-1}(\tfrac{V_r}{a}) - r\,r_2\tan^{-1}(\tfrac{V_r}{a})]dr \end{aligned} \tag{5.48}$$

Rewrite equation (5.44) into

$$V_r = \frac{u_1-u_2}{r_1-r_2}r + \frac{r_1u_2-r_2u_1}{r_1-r_2} = Yr + Z \tag{5.49}$$

and substitute it in equation (5.48).

$$\frac{\partial \Phi}{\partial u_1} = \frac{-4*m*g}{r_1-r_2}\int_{r_1}^{r_2}[r^2\tan^{-1}(\tfrac{Y}{a}r + \tfrac{Z}{a}) - r\,r_2\tan^{-1}(\tfrac{Y}{a}r + \tfrac{Z}{a})]dr \tag{5.50}$$

Since REDUCE is still unable to handle equation (5.50), further treatments are required. Let

$$W = \frac{Y}{a}r + \frac{Z}{a} = Y'r + Z' \tag{5.51}$$

then

$$dr = \frac{dW}{Y'} \tag{5.52}$$

and equation (5.50) becomes

$$\frac{\partial \Phi}{\partial u_1} = \frac{-4*m*g}{r_1 - r_2}\int_{W_1}^{W_2}\left[\frac{(W-Z')^2}{Y'^3}\tan^{-1}W - \frac{(W-Z')}{Y'^2}r_2\tan^{-1}W\right]dW \tag{5.53}$$

Note that the upper and lower limits of integration are changed. Starting from this point, REDUCE can proceed by itself. The tasks it performs in this specific problem include the integration and back substitution of the variables. The other quantity, such as $\frac{\partial \Phi}{\partial u_2}$ etc., can be calculated in the similar manner. The second derivatives are simply computed by differentiation of the results of the first derivatives. The REDUCE program and a part of fortran solution are presented as follows.

```
%----------------------------------------------------------------------
%REDUCE program to calculate friction part and its derivatives
%----------------------------------------------------------------------
OFF EXP;
A1:=INT(ATAN(W)*(W-ZP)**2/YP-ATAN(W)*(W-ZP)*R2,W)*4*TM*TK/((R1-
R2)*YP**2);
A2:=INT(ATAN(W)*R1*(W-ZP)-ATAN(W)*(W-ZP)**2/YP,W)*4*TM*TK/((R1-
R2)*YP**2);
LET YP=(U1-U2)/(A*(R1-R2)),ZP=(R1*U2-R2*U1)/(A*(R1-R2));
LET LOG((A**2+U2**2)/A**2)-LOG((A**2+U1**2)/A**2)
=LOG((A**2+U2**2)/(A**2+U1**2));
OFF EXP;
ON FORT;
OFF ECHO;
CARDNO!*:=10;
OUT "look.ftn";
WRITE
"      SUBROUTINE FRIC(U1,U2,A,TK,TM,R1,R2,B1,B2,B11,B12,B22)";
WRITE "     IMPLICIT REAL*8(A-H,O-Z)";
B1:=SUB(W=U2/A,A1)-SUB(W=U1/A,A1);
B2:=SUB(W=U2/A,A2)-SUB(W=U1/A,A2);
B11:=DF(B1,U1);
B22:=DF(B2,U2);
B12:=DF(B1,U2);
WRITE "     RETURN";
WRITE "     END";
OFF FORT;
SHUT "look.ftn";
BYE;
```

```
C-----------------------------------------------------------------------
C The fortran subroutine made by REDUCE for frictional part.
C-----------------------------------------------------------------------

      SUBROUTINE FRIC(U1,U2,A,TK,TM,R1,R2,B1,B2,B11,B12,B22)
     IMPLICIT REAL*8(A-H,O-Z)
     ANS4=LOG((A**2+U2**2)/(A**2+U1**2))*A**3*R2**2-3.*LOG
    . ((A**2+U2**2)/(A**2+U1**2))*A*R1**2*U2**2+3.*LOG((A
    . **2+U2**2)/(A**2+U1**2))*A*R1*R2*U1*U2+3.*LOG((A**2+
    . U2**2)/(A**2+U1**2))*A*R1*R2*U2**2-3.*LOG((A**2+U2**
    . 2)/(A**2+U1**2))*A*R2**2*U1*U2+A*R1**2*U1**2-6.*A*R1
    . **2*U1*U2+5.*A*R1**2*U2**2+A*R1*R2*U1**2+6.*A*R1*R2*
    . U1*U2-7.*A*R1*R2*U2**2-2.*A*R2**2*U1**2+2.*A*R2**2*
    . U2**2
     ANS3=-6.*ATAN(U2/A)*A**2*R1**2*U2+3.*ATAN(U2/A)*A**2*
    . R1*R2*U1+9.*ATAN(U2/A)*A**2*R1*R2*U2-3.*ATAN(U2/A)*A
    . **2*R2**2*U1-3.*ATAN(U2/A)*A**2*R2**2*U2+2.*ATAN(U2/
    . A)*R1**2*U2**3-3.*ATAN(U2/A)*R1*R2*U1*U2**2-ATAN(U2/
    . A)*R1*R2*U2**3+3.*ATAN(U2/A)*R2**2*U1*U2**2-ATAN(U2/
    . A)*R2**2*U2**3+LOG((A**2+U2**2)/(A**2+U1**2))*A**3*
    . R1**2-2.*LOG((A**2+U2**2)/(A**2+U1**2))*A**3*R1*R2+
    . ANS4
     ANS2=6.*ATAN(U1/A)*A**2*R1**2*U2-3.*ATAN(U1/A)*A**2*
    . R1*R2*U1-9.*ATAN(U1/A)*A**2*R1*R2*U2+3.*ATAN(U1/A)*A
    . **2*R2**2*U1+3.*ATAN(U1/A)*A**2*R2**2*U2-2.*ATAN(U1/
    . A)*R1**2*U1**3+6.*ATAN(U1/A)*R1**2*U1**2*U2-6.*ATAN(
    . U1/A)*R1**2*U1*U2**2+ATAN(U1/A)*R1*R2*U1**3-3.*ATAN(
    . U1/A)*R1*R2*U1**2*U2+6.*ATAN(U1/A)*R1*R2*U1*U2**2+
    . ATAN(U1/A)*R2**2*U1**3-3.*ATAN(U1/A)*R2**2*U1**2*U2+
    . ANS3
     ANS1=2.*ANS2*TK*TM
     B1=ANS1/(3.*(U1-U2)**3)
          .
          .
          .
          .
     ANS1=2.*ANS2*TK*TM
     B12=ANS1/(U1-U2)**4
     RETURN
     END
```

5.6 Numerical evaluation and result treatment

The numerical scheme employed is the Newton-Raphson iteration with a displacement increment of 0.01 in each stage. The initial guesses are slightly modified from the solution of the elastic ring compression problem. Since the existence of zero velocities in r-direction at interface nodes will overflow the subroutine FRIC, the problem is modified by assigning a different small number to the r-component of every relevant node. The question that arises is how small they should be. According to the experiments, only the numbers which are smaller than 10^{-12} will achieve convergence with this scheme. The convergence criterion used here is $\frac{\|\Delta U\|}{\|U\|} \leq 0.00005$.

The boundary conditions are specified at two parts of the boundary :

1. Symmetric boundary condition --- The velocities in z direction [see Figure 5.5] are specified to be zeros along the z=0 boundary.

2. Die velocity boundary condition --- The velocities at the interface surface between the die and the working piece are specified as unit per second in the negative z direction.

Numerical integration of non-integrable terms is performed by the 4-point Guass quadrature rule. The assembly of a global matrix is also done numerically. The equation solver is the Gauss elimination method, from the IMSL subroutine library.

The deformed configurations for friction factor for m=0.5 and m=0.0 are shown in Figures 5.6 and 5.7, respectively. As the figures show, the deformed shapes are completely different for low and high frictional factors. The velocity distributions in the deformed states are also plotted in Figures 5.8 and 5.9, respectively. The neutral lines in both cases are visible from pictures. Figures 5.10 and 5.11 also show the effective stress[7] distributions for two cases. As the shapes of elements are distorted, the error increases and the convergence of the scheme becomes harder. In order to continue the execution, the technique of adaptive mesh needs to be introduced.

---

[7] The effective stress is defined as

$$\sigma_e = (\sigma_x^2 + \sigma_y^2 + \sigma_z^2 - \sigma_x\sigma_y - \sigma_y\sigma_z - \sigma_x\sigma_z + 3\tau_{xy}^2 + 3\tau_{yz}^2 + 3\tau_{xz}^2)^{\frac{1}{2}}$$

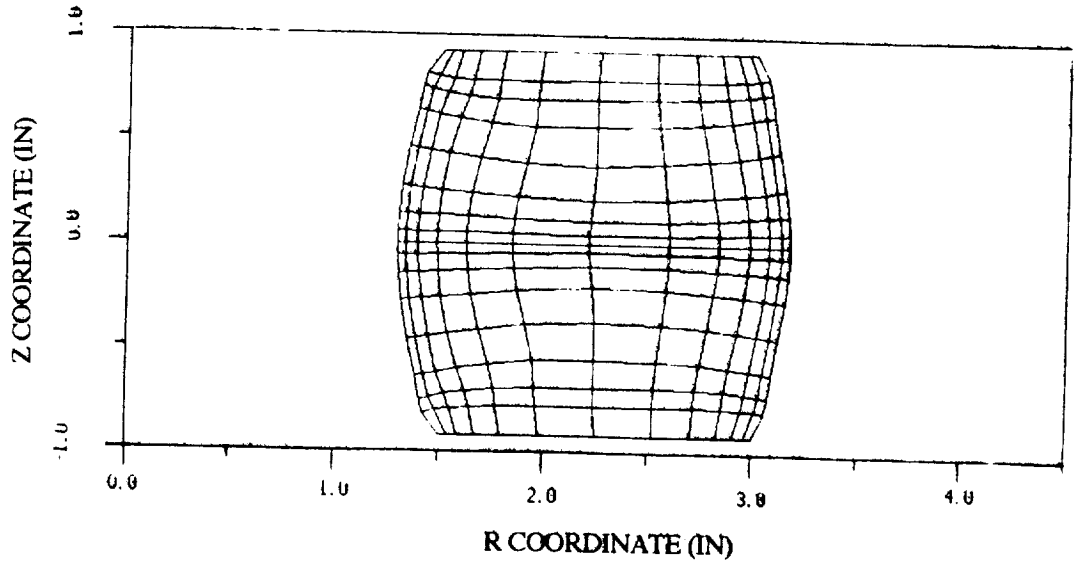

Figure 5.6 : Deformed configuration after the 8th stage for m=0.5

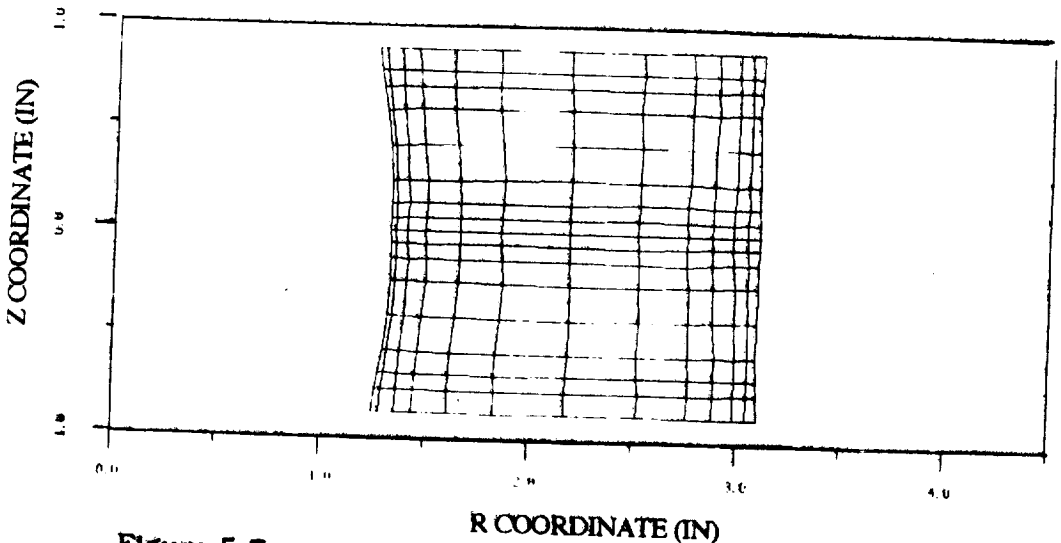

Figure 5.7 : Deformed configuration after the 7th stage for m=0.0

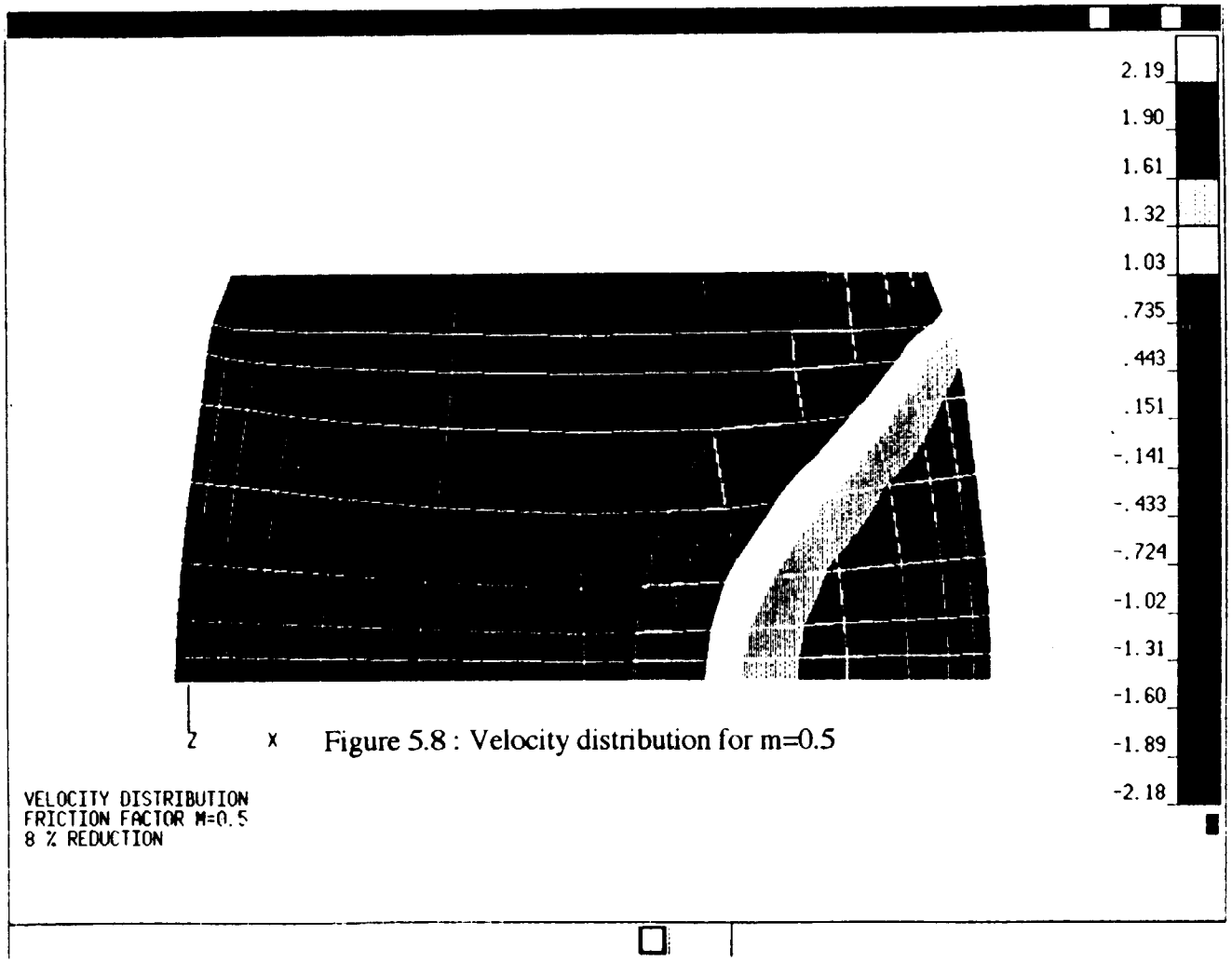

Figure 5.8 : Velocity distribution for m=0.5

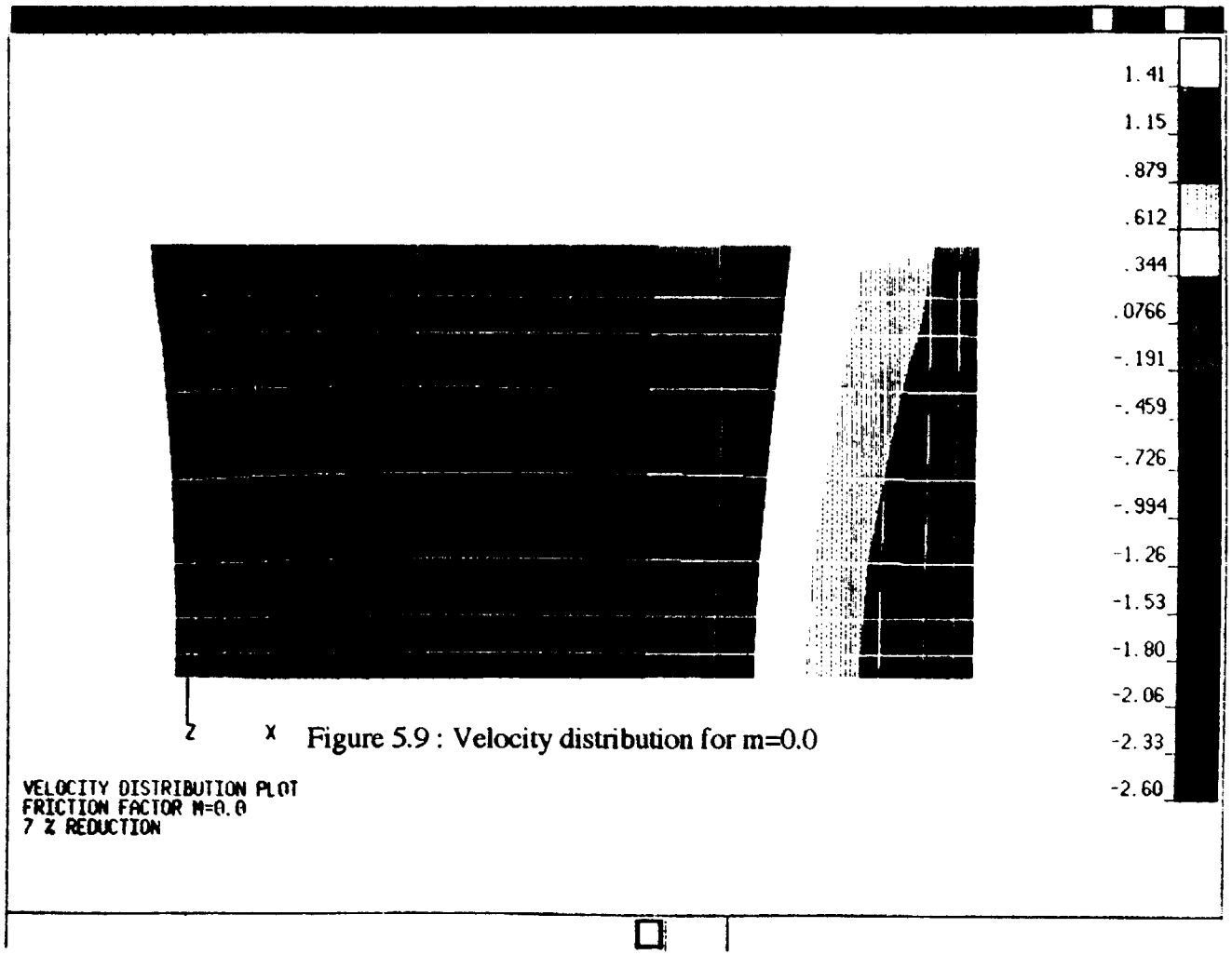

Figure 5.9 : Velocity distribution for m=0.0

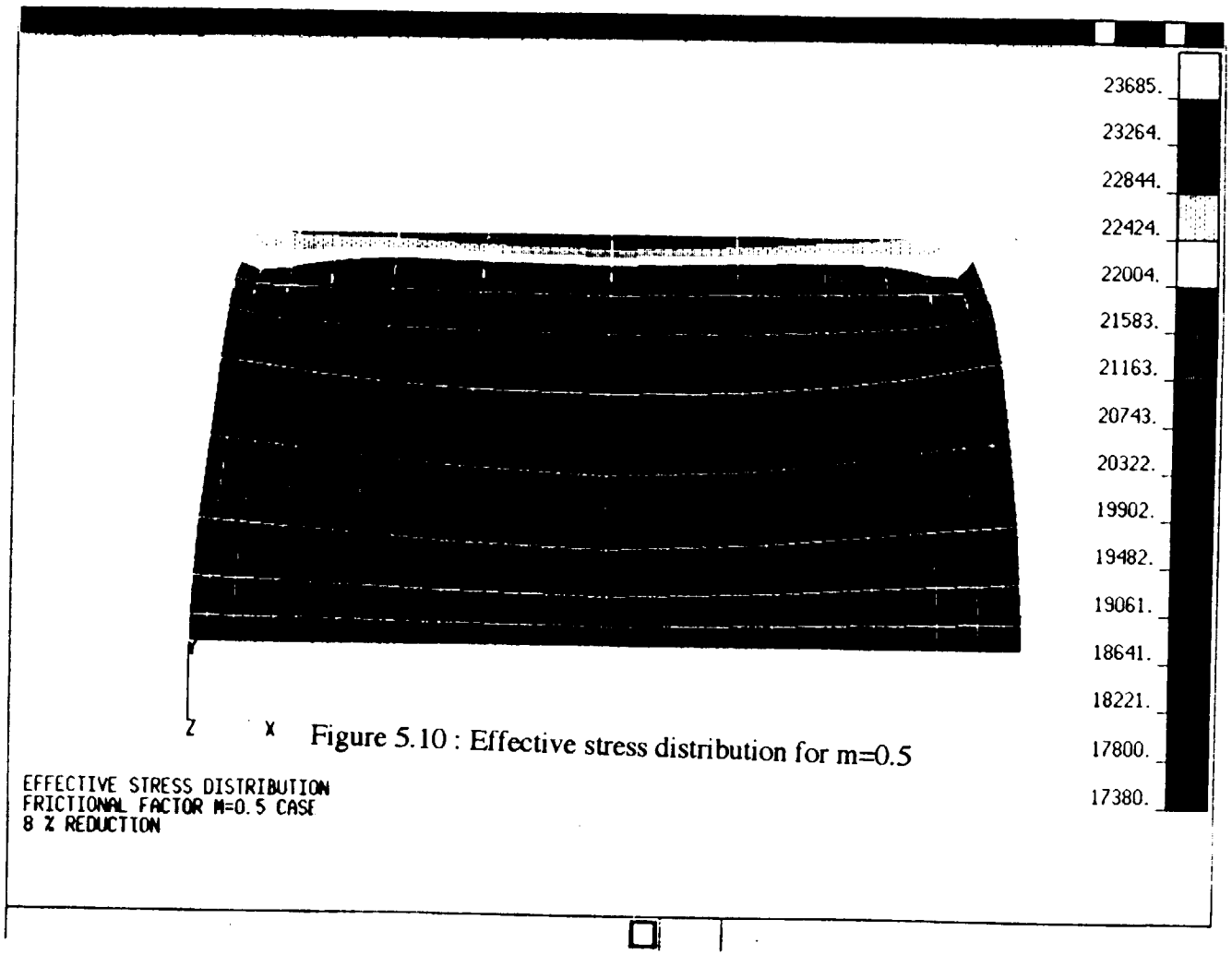

Figure 5.10 : Effective stress distribution for m=0.5

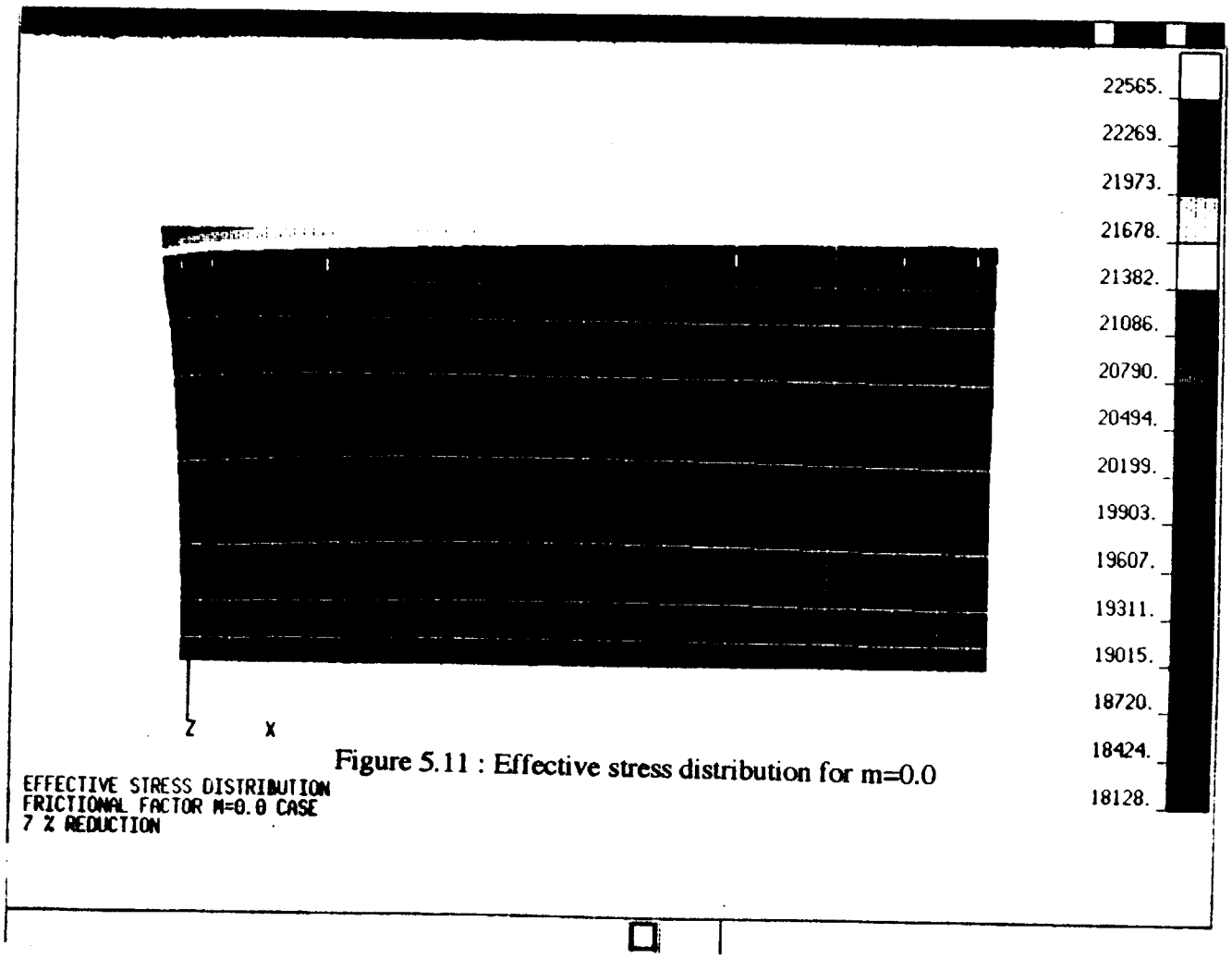

Figure 5.11 : Effective stress distribution for m=0.0

5.7 References

[1] C. C. Chen and S. Kobayashi,"Rigid plastic finite element analysis of ring compression", ASME AMD-vol. 28, pp 163--174,1978.

[2] C. H. Lee and S. Kobayashi,"New solutions to rigid-plastic deformations using a matrix method", Trans. ASME, J. of Engrg. for Ind., vol. 95, pp 865-873, 1973.

[3] C. H. Lee and S. Kobayashi,"Analysis of axisymmetric upsetting and plane-strain side pressing of solid cylinders by the finite element method, J. of engineering for industry, pp 445-454, May 1971.

[4] S. N. Shah and S. Kobayashi,"Rigid-plastic analysis of cold heading by the matrix method", proceeding. of the 15th int. machine tool design & research conf., Birmingham, England, pp 603-610, 1974.

[5] R. Hill,"The mathematical theory of plasticity", Oxford, pp 40, 1983.

[6] A. C. Hearn,"REDUCE user's manual", The Rand corp., Santa Monica, CA 90406, version 3.2, April 1985.

[7] K. Washizu,"Variational methods in elasticity and plasticity", 3rd edition, Pergamon press, 1982.

[8] L. M. Kachanov,"Fundamentals of the theory of plasticity", Mir publisher, Moscow, 1974.

[9] D. R. J. Owen, E. Hinton,"Finite elements in plasticity : theory and practice", Pineridge Press Limited, Swansea, U. K., 1980.

[10] Edited by J. F. T. Pittman, R. D. Wood, J. M. Alexander, O. C. Zienkiewicz,"Numerical methods in industrial forming progresses", Pineridge Press, Swansea, U. K., 1982.

[11] W. Szczepinski,"Introduction to the mechanics of plastic forming of metals", Sijthoff & Noordhoff, 1979.

[12] P. polukhin, S. Gorelik and V. Vorontsov,"Physical Principles, of plastic Deformation", Mir Publishers, Moscow, 1982.

[13] L. E. Malvern,"Introduction to the mechanics of a continuous medium", Prentice-Hall, Inc., 1969.

[14] G. E. Mase,"Theory and problem of continuum mechanics", Schaum's outline series in engineering, MacGraw-Hill Book company, 1970.

[15] D. G. Luenberger,"Linear and nonlinear programming", 2nd edition, Addison-Wesley, 1984.

CHAPTER VI

APPLICATION OF SYMBOLIC AND ALGEBRAIC MANIPULATION TO THE PLATE PROBLEM

6.1 Introduction

Although many shell theories exist in literature, there are only two distinct concepts from which these theories are derived. One takes the three-dimensional body as a starting point and tries by various means to reduce the problem to the form that can be expressed in a two-dimensional manifold. This class of theories are called *derived theories*. The works by W. T. Koiter, and E. Reissner are in this class. The theory adopted by this study belongs to this class. The other class of theories consider a shell as a bounded region of some deformable two-dimensional manifold, and are supplemented by one or more fields of vectors over this manifold. This class of theories are called *direct theories*. A. E. Green, P. M. Naghdi, and W. L. Wainwright worked on this class of shell theories. Since the real shell is a three-dimensional body, the direct approach has to rely upon some *a priori* statements.

Despite a large amount of publications using the finite element method to solve plate and shell problems, none deals with the problems by employing the tool of symbolic and algebraic manipulation. This is not only due to the late availability of software, but also due to the capacity limitations existing in the symbolic and algebraic software[8]. This chapter outlines the simple and universal methodology to solve the plate and shell problems, then presents examples which apply symbolic and algebraic manipulation to them, and finally switches to a numerical method at the point where the symbolic manipulation is stuck by its limitations. As a consequence of this work, the analytic study of plate and shell problems by computer are pushed a step further.

[8] See next chapter for a detail discussion on the capacity limitation of symbolic and algebraic manipulation.

6.2 Preliminary formulation

Stating from the equations of equilibrium in three-dimensional state,

$$D_j \sigma^{ij}(u^\alpha, z) + P^i(u^\alpha, z) = 0 \tag{6.1}$$

Where

- D_j : operator of covariant derivative in 3-D space.
- σ^{ij} : 3-D state of stress (i,j=1, 2, 3).
- P^i : external volume force.
- u^α : Gaussian coordinates of surface (α=1, 2).
- z : normal coordinate of surface.

and applying the following Kirchhoff-Love hypothesis to virtual displacements

$$\delta v_\alpha(u^\alpha, z) = (a_\alpha^\gamma - z d_\alpha^\gamma) * (\delta v_\gamma - z\,\delta q_\gamma) \tag{6.2}$$

$$\delta v_3(u^\alpha, z) = \delta w \tag{6.3}$$

where

- a_α^γ : component of mixed metric tensor.
- d_α^γ : component of mixed curvature tensor.

Here the rotation δq_γ is defined as

$$\delta q_\alpha = d_{\alpha\beta}\,\delta v^\beta + \delta w_{,\alpha} \tag{6.4}$$

Then, following the lengthy derivations by F. I. Niordson in his *Shell Theory* [1], the principle of virtual work gives

$$\iint_D [N^{\alpha\beta}\,\delta E_{\alpha\beta} + M^{\alpha\beta}\,\delta K_{\alpha\beta}]\,dA = \iint_D [F^\alpha \delta v_\alpha + p\delta w]\,dA + \oint_c [T^\alpha \delta v_\alpha + M^\alpha \delta r_\alpha + Q\delta w]\,ds \tag{6.5}$$

Where

- $N^{\alpha\beta}$: effective membrane stress tensor.
- $M^{\alpha\beta}$: effective moment tensor.
- $dE_{\alpha\beta}$: virtual strain tensor.
- $dK_{\alpha\beta}$: virtual bending tensor.
- F^{α}, p : effective load components in surface and normal directions, respectively.
- T^{α} : membrane force vector acting on the boundary.
- M^{α} : moment vector acting on the boundary.
- Q : supplemented shear force on the boundary.
- dv_{α}, dw : virtual displacements in plane and transverse directions, respectively.
- $dr_{\alpha}=\varepsilon^{\alpha\beta}dq_{\beta}$
- $\varepsilon^{\alpha\beta}$: alternate tensor.

Mathematically, the strain tensor $E_{\alpha\beta}$ and bending tensor $K_{\alpha\beta}$ are defined as

$$E_{\alpha\beta} = \tfrac{1}{2}(a^{*}_{\alpha\beta} - a_{\alpha\beta}) \tag{6.6}$$

$$K_{\alpha\beta} = d^{*}_{\alpha\beta} - d_{\alpha\beta} \tag{6.7}$$

where the superscript asterisks indicate the deformed state and can be derived as follows :

$$a^{*}_{\alpha\beta} = a_{\alpha\beta} + p_{\alpha\beta} + p_{\beta\alpha} + p^{\gamma}_{\alpha} p_{\beta\gamma} + q_{\alpha} q_{\beta} \tag{6.8}$$

$$d^{*}_{\alpha\beta} = (\tfrac{a}{a^{*}})^{\frac{1}{2}}[(1 + p^{\gamma}_{\gamma} + P/a)(d_{\alpha\beta} + D_{\beta} q_{\alpha} + d^{\gamma}_{\beta} p_{\alpha\gamma}) - (q^{\rho} + \varepsilon^{\rho\eta}\varepsilon^{\gamma\delta} q_{\gamma} p_{\delta\eta})(D_{\beta} p_{\alpha\rho} - d_{\beta\rho} q_{\alpha})] \tag{6.9}$$

The generalized two-dimensional displacement gradient p and its determinant in equation (6.9) are expressed as :

$$p_{\alpha\beta} = D_{\alpha} v_{\beta} - d_{\alpha\beta} w \tag{6.10}$$

$$\vartheta = \det(p_{\alpha\beta}) \tag{6.11}$$

After linearization, the strain and bending tensors can be expressed in terms of displacements as follows :

$$E_{\alpha\beta} \approx \tfrac{1}{2}(D_\alpha v_\beta + D_\beta v_\alpha) - d_{\alpha\beta} w \tag{6.12}$$

$$K_{\alpha\beta} \approx D_\alpha D_\beta w + d_{\alpha\gamma} D_\beta v^\gamma + d_{\beta\gamma} D_\alpha v^\gamma + v^\gamma D_\beta d_{\gamma\alpha} - d_{\beta\gamma} d^\gamma_\alpha w \tag{6.13}$$

Considering the isotropic thin elastic shell for simplicity. The constitutive equations will be :

$$N^{\alpha\beta} = \frac{Eh}{1-\nu^2}[(1-\nu)E^{\alpha\beta} + \nu a^{\alpha\beta} E^\gamma_\gamma] \tag{6.14}$$

$$M^{\alpha\beta} = \frac{Eh}{12(1-\nu^2)}[(1-\nu)K^{\alpha\beta} + \nu a^{\alpha\beta} K^\gamma_\gamma] \tag{6.15}$$

The substitution of the last four equations into equation (6.5) will result in the weak form which is expressed in terms of displacement vectors. Before applying the finite element method to solve equation (6.5), a universal methodology is outlined in the next paragraph.

6.3 Methodology for solving shell problem by FEM

After the preparations of mathematical formulation, it is necessary to discretize equation (6.5) to solve shell problems by FEM. However, as equations (6.12) to (6.15) show, the calculations of constitutive equations and strain-displacement relations involve the evaluations of metric, curvature tensors, and covariant derivatives. Moreover, in general, the calculations of covariant derivatives require the computations of Christoffel symbols. If a given geometric domain is complex, these calculations will be tedious. With the help of symbolic and algebraic manipulation, these tough tasks can be performed by simply giving the parametric equations of the surface. Here the outline of methodology to solve shell problems is presented as follows :

1 Finding the parametric equations of the middle surface for a given shell.

2 Calculating the Christoffel symbols, the metric and curvature tensors based on the parametric equations. If the parametric equations are chosen correctly, the resulting metric and curvature tensor should obey the integrability condition. In other words, they should not violate the Coddazzi-Mainardi equations, Gaussian equations and the regular condition.

3. Substituting the metric, curvature tensors, and Christoffel symbols into constitutive equations and strain-displacement equations.

4 Discretizing the variational form of equation (6.5) and solving it by finite element method.

The above methodology is universal for any shell problems. Different shell geometries can be solved in the same way by simply feeding appropriate parametric equations into the symbolic and algebraic manipulator REDUCE. Based on this methodology, an example of plate problem is shown in the following paragraph.

6.4 Symbolic and algebraic manipulation application to plate problems

6.4.1 Methodology

1 Three parametric equations are chosen as follows :

$f^1=u^1$, $f^2=u^2$, f^3=constant

2 Calculate surface metric, curvature tensors, and Christoffel symbols symbolically. These calculations are based on their fundamental definitions. given by:

• metric tensor :

$$a_{\alpha\beta} = f^i_{,\alpha} f^i_{,\beta} \tag{6.16}$$

• curvature tensor :

$$d_{\alpha\beta} = X^i f^i_{,\alpha\beta} \tag{6.17}$$

where $f^i_{,1}$, $f^i_{,2}$ and X^i are surface Gaussian and normal coordinates. They are shown in Figure 5.2 pictorially, and X^i is defined by :

$$X^i = a^{-\frac{1}{2}} e_{ijk} f^j_{,1} f^k_{,2} \tag{6.18}$$

• The 2nd kind of Christoffel symbols is defined as:

$$\left\{ \begin{matrix} \alpha \\ \beta \quad \gamma \end{matrix} \right\} = a^{\alpha\beta} [\beta\gamma, \rho] = \frac{1}{2} a^{\alpha\beta} \left[\frac{\partial a_{\beta\rho}}{\partial u^\gamma} + \frac{\partial a_{\gamma\rho}}{\partial u^\beta} - \frac{\partial a_{\beta\gamma}}{\partial u^\rho} \right] \tag{6.19}$$

The REDUCE program for calculating equations (6.16), (6.17), (6.18) and (6.19) is presented as follows. The results follow the program.

```
%*************************************************************
% REDUCE PROGRAM FOR CALCULATING METRIC, CURVATURE
% TENSORS & CHRISTOFFEL SYMBOLS
%*************************************************************

%-------------------------------------------------------
%INPUTTING THE PARAMETRIC EQUATIONS
%-------------------------------------------------------
MATRIX A(2,2),CA(2,2),F(2,3),D(2,2);
ARRAY X(3),C1(2,2,2),C2(2,2,2),N(3),E(3,3,3),U(2);
U(1):=S;
U(2):=P;
OFF PERIOD;
X(1):=U(1);
X(2):=U(2);
X(3):=CONSTANT;
FOR I:=1:2 DO FOR J:=1:3 DO
F(I,J):=DF(X(J),U(I));
FOR ALL T1 LET COS(T1)**2+SIN(T1)**2=1;

%-------------------------------------------------------
%CALCULATING COVARIANT METRIC TENSOR
%-------------------------------------------------------
FOR M:=1:2 DO FOR N:=1:2 DO
A(M,N):=FOR I:=1:3 SUM DF(X(I),U(M))*DF(X(I),U(N));
A:=A;
DETA:=DET(A);

%-------------------------------------------------------------------
%CALCULATING CONTRAVARIANT METRIC TENSOR
%-------------------------------------------------------------------
FOR L:=1:2 DO FOR M:=1:2 DO
IF L=1 AND M=1 THEN CA(L,M):=A(2,2)/DETA
ELSE IF L NEQ M THEN CA(L,M):=-A(M,L)/DETA
ELSE CA(L,M):=A(1,1)/DETA;

%-------------------------------------------------------------------
%CALCULATING THE 1ST & 2ND CHRISTOFFEL SYMBOL
%-------------------------------------------------------------------
WRITE "THE 2ND CHRISTOFFEL SYMBOL";
FOR L:=1:2 DO FOR M:=1:2 DO FOR N:=1:2 DO
C1(L,M,N):=(1/2)*(DF(A(L,N),U(M))+DF(A(M,N),U(L))
      -DF(A(L,M),U(N)));
FOR L:=1:2 DO FOR M:=1:2 DO FOR N:=1:2 DO
<<C2(L,M,N):=FOR I:=1:2 SUM CA(L,I)*C1(M,N,I);
    WRITE "C2(",L,",",M,",",N,")=",C2(L,M,N)>>;

%-------------------------------------------------------------
%CALCULATING ALTERNATING TENSOR E(I,J,K)
%-------------------------------------------------------------
FOR I:=1:3 DO FOR J:=1:3 DO FOR K:=1:3 DO
<<IF (I=J OR J=K OR I=K) THEN E(I,J,K):=0
ELSE IF (I=1 AND J=2 AND K=3) OR (I=2 AND J=3 AND K=1) OR
```

```
    (I=3 AND J=1 AND K=2) THEN E(I,J,K):=1
ELSE E(I,J,K):=-1; WRITE E(I,J,K)>>;

%-----------------------------------------------
%CALCULATING NORMAL VECTOR
%-----------------------------------------------
FOR I:=1:3 DO
N(I):=(FOR J:=1:3 SUM FOR K:=1:3 SUM
E(I,J,K)*F(1,J)*F(2,K))/SQRT(DETA);

%-------------------------------------------------
%CALCULATING CURVATURE TENSOR
%-------------------------------------------------
FOR J:=1:2 DO FOR K:=1:2 DO
D(J,K):=FOR I:=1:3 SUM N(I)*DF(F(J,I),U(K));
FOR ALL T1 CLEAR COS(T1)**2+SIN(T1)**2;
FOR ALL T1 LET COS(T1)**2-1=-SIN(T1)**2;

%---------------------------------
%OUTPUTTING RESULTS
%---------------------------------
OFF PERIOD;
OFF ECHO;
OUT "SURFACE";
WRITE "      ------------------------------------------------------";
WRITE "      THE COMPONENTS OF METRIC TENSOR ";
WRITE "      ------------------------------------------------------";
A:=A;
WRITE "      ------------------------------------------------------------";
WRITE "      THE COMPONENTS OF CURVATURE TENSOR";
WRITE "      ------------------------------------------------------------";
D:=D;
WRITE "      ----------------------------------------";
WRITE "      THE CHRISTOFFEL SYMBOLS";
WRITE "      ----------------------------------------";
FOR I:=1:2 DO FOR J:=1:2 DO FOR K:=1:2 DO
WRITE  "CHRIS(",I,",",J,",",K,")=",C2(I,J,K);
SHUT "SURFACE";
BYE;
```

The outputs of REDUCE are presented as follows :

```
     ------------------------------------------------------
     THE COMPONENTS OF METRIC TENSOR
     ------------------------------------------------------
A(1,1) := 1
A(1,2) := 0
A(2,1) := 0
A(2,2) := 1

     ------------------------------------------------------------
     THE COMPONENTS OF CURVATURE TENSOR
     ------------------------------------------------------------
D(1,1) := 0
```

```
D(1,2) := 0
D(2,1) := 0
D(2,2) := 0

---------------------------------
THE CHRISTOFFEL SYMBOLS
---------------------------------
CHRIS(1,1,1)=0
CHRIS(1,1,2)=0
CHRIS(1,2,1)=0
CHRIS(1,2,2)=0
CHRIS(2,1,1)=0
CHRIS(2,1,2)=0
CHRIS(2,2,1)=0
CHRIS(2,2,2)=0
```

As the output from REDUCE shows, all of the Christoffel symbols vanish in the plate case. According to the above solutions, the metric and curvature tensors in matrix form are

$$[a_{\alpha\beta}] = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad [d_{\alpha\beta}] = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \tag{6.20}$$

3. After substituting the calculated metric and curvature tensors, the constitutive equations are as follows

• For membrane :

$$\begin{Bmatrix} N^{11} \\ N^{12} \\ N^{22} \end{Bmatrix} = \frac{Eh}{1-\nu^2} \begin{bmatrix} 1 & 0 & \nu \\ 0 & 1-\nu & 0 \\ \nu & 0 & 1 \end{bmatrix} \begin{Bmatrix} E^{11} \\ E^{12} \\ E^{22} \end{Bmatrix} \tag{6.21}$$

•For bending :

$$\begin{Bmatrix} M^{11} \\ M^{12} \\ M^{22} \end{Bmatrix} = \frac{Eh^3}{12(1-\nu^2)} \begin{bmatrix} 1 & 0 & \nu \\ 0 & 1-\nu & 0 \\ \nu & 0 & 1 \end{bmatrix} \begin{Bmatrix} K^{11} \\ K^{12} \\ K^{22} \end{Bmatrix} \tag{6.22}$$

The relationships between linearized strain-displacement and bending curvature tensors are from equations (6.12) and (6.13) :

$$E_{\alpha\beta} = \tfrac{1}{2}(D_\alpha v_\beta + D_\beta v_\alpha) = \tfrac{1}{2}(v_{\beta,\alpha} + v_{\alpha,\beta}) \tag{6.23}$$

$$K_{\alpha\beta} = D_\alpha D_\beta w = w_{,\alpha\beta} \tag{6.24}$$

where the disappearance of Christoffel symbols in this case eliminates the distinction between covariant differentiation and ordinary partial differentiation. In addition, the unity of the metric tensors remove the distinction between contravariant and covariant tensors.

4. Finite element method starts (see the following paragraph).

6.4.2 Finite element discretization

The element chosen for this topic is the combination of the constant strain triangle (CST) element with the Cheung, King, Zienkiewicz (CKZ) triangle element. The considerations of selecting this element will be discussed later. The shape functions for this element are

$$
\begin{aligned}
L_\alpha &= \xi_\alpha \\
N_\alpha &= \xi_\alpha + \xi_\alpha\xi_\beta(\xi_\alpha - \xi_\beta) + \xi_\alpha\xi_\gamma(\xi_\alpha - \xi_\gamma) \\
N_{\alpha x} &= 2\Delta[c_\gamma(\xi_\alpha^2\xi_\beta + \tfrac{1}{2}\xi_1\xi_2\xi_3) - c_\beta(\xi_\alpha^2\xi_\gamma + \tfrac{1}{2}\xi_1\xi_2\xi_3)] \\
N_{\alpha y} &= 2\Delta[b_\beta(\xi_\alpha^2\xi_\gamma + \tfrac{1}{2}\xi_1\xi_2\xi_3) - b_\gamma(\xi_\alpha^2\xi_\beta + \tfrac{1}{2}\xi_1\xi_2\xi_3)]
\end{aligned}
\tag{6.25}
$$

Where (α,β,γ) is the permutation of (1,2,3) and no summation convention is applied in equation (6.25). The constants b_i, c_i, and Δ are defined as follows :

$$
\begin{aligned}
&b_1 = \frac{y_2 - y_3}{2\Delta}, \qquad b_2 = \frac{y_3 - y_1}{2\Delta}, \qquad b_3 = \frac{y_1 - y_2}{2\Delta} \\
&c_1 = \frac{x_3 - x_2}{2\Delta}, \qquad c_2 = \frac{x_1 - x_3}{2\Delta}, \qquad c_3 = \frac{x_2 - x_1}{2\Delta} \\
&2\Delta = x_2y_3 - x_3y_2 + x_3y_1 - x_1y_3 + x_1y_2 - x_2y_1
\end{aligned}
\tag{6.26}
$$

Then the deflections u, v, w in local coordinates x, y, z direction can be interpolated by

$$
\begin{aligned}
u &= \sum_{i=1}^{3} u_i L_i \\
v &= \sum_{i=1}^{3} v_i L_i \\
w &= w_\alpha N_\alpha + \frac{\partial w}{\partial x}\Big|_\alpha N_{\alpha x} + \frac{\partial w}{\partial y}\Big|_\alpha N_{\alpha y}
\end{aligned}
\tag{6.27}
$$

The substitution of equations (6.21) to (6.27) into (6.5) gives the discretization form

$$
\iint_D [B_1^T D_1 B_1 + B_2^T D_2 B_2]dA \cdot d = \iint_D f^T dA + \int_c S^T ds + \int_c M^T ds \tag{6.28}
$$

Where

$$D_1 = \frac{Eh}{1-\nu}\begin{bmatrix} 1 & 0 & \nu \\ 0 & \frac{1-\nu}{2} & 0 \\ \nu & 0 & 1 \end{bmatrix} \tag{6.29}$$

$$D_2 = \frac{Eh^3}{12(1-\nu)}\begin{bmatrix} 1 & 0 & \nu \\ 0 & \frac{1-\nu}{2} & 0 \\ \nu & 0 & 1 \end{bmatrix} \tag{6.30}$$

$$B_1 = \begin{bmatrix} b_1 & b_2 & b_3 & 0 & 0 & 0 \\ c_1 & c_2 & c_3 & b_1 & b_2 & b_3 \\ 0 & 0 & 0 & c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial \xi_1} & 0 \\ \frac{\partial}{\partial \xi_2} & 0 \\ \frac{\partial}{\partial \xi_3} & 0 \\ 0 & \frac{\partial}{\partial \xi_1} \\ 0 & \frac{\partial}{\partial \xi_2} \\ 0 & \frac{\partial}{\partial \xi_3} \end{bmatrix} \cdot$$

$$\begin{bmatrix} L_1 & 0 & 0 & 0 & 0 & L_2 & 0 & 0 & 0 & 0 & L_3 & 0 & 0 & 0 & 0 \\ 0 & L_1 & 0 & 0 & 0 & 0 & L_2 & 0 & 0 & 0 & 0 & L_3 & 0 & 0 & 0 \end{bmatrix} \tag{6.31}$$

$$B_2 = \begin{bmatrix} b_1 & b_2 & b_3 & 0 & 0 & 0 \\ c_1 & c_2 & c_3 & b_1 & b_2 & b_3 \\ 0 & 0 & 0 & c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial \xi_1} & 0 \\ \frac{\partial}{\partial \xi_2} & 0 \\ \frac{\partial}{\partial \xi_3} & 0 \\ 0 & \frac{\partial}{\partial \xi_1} \\ 0 & \frac{\partial}{\partial \xi_2} \\ 0 & \frac{\partial}{\partial \xi_3} \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{Bmatrix} \frac{\partial}{\partial \xi_1} \\ \frac{\partial}{\partial \xi_2} \\ \frac{\partial}{\partial \xi_3} \end{Bmatrix} \tilde{N} \tag{6.32}$$

$$M = [M^1 \quad M^2] \begin{bmatrix} c_1 & c_2 & c_3 \\ b_1 & b_2 & b_3 \end{bmatrix} \begin{Bmatrix} \frac{\partial}{\partial \xi_1} \\ \frac{\partial}{\partial \xi_2} \\ \frac{\partial}{\partial \xi_3} \end{Bmatrix} \tilde{N} \tag{6.33}$$

$$\tilde{N} = [0 \quad 0 \quad N_1 \quad N_{1x} \quad N_{1y} \quad 0 \quad 0 \quad N_2 \quad N_{2x} \quad N_{2y} \quad 0 \quad 0 \quad N_3 \quad N_{3x} \quad N_{3y}]$$

$$d^T = [u_1 \quad v_1 \quad w_1 \quad w_{1,x} \quad w_{1,y} \quad u_2 \quad v_2 \quad w_2 \quad w_{2,x} \quad w_{2,y} \quad u_3 \quad v_3 \quad w_3 \quad w_{3,x} \quad w_{3,y}] \tag{6.34}$$

$$r = [F^1 \quad F^2 \quad p]\tilde{N} \tag{6.35}$$

$$S = [T^1 \quad T^2 \quad q]\tilde{N} \tag{6.36}$$

$$\tilde{N} = \begin{bmatrix} L_1 & 0 & 0 & 0 & 0 & L_2 & 0 & 0 & 0 & 0 & L_3 & 0 & 0 & 0 & 0 \\ 0 & L_1 & 0 & 0 & 0 & 0 & L_2 & 0 & 0 & 0 & 0 & L_3 & 0 & 0 & 0 \\ 0 & 0 & N_1 & N_{1x} & N_{1y} & 0 & 0 & N_2 & N_{2x} & N_{2y} & 0 & 0 & N_3 & N_{3x} & N_{3y} \end{bmatrix} \tag{6.37}$$

6.4.3 Numerical results and post-process

Three boundary conditions are applied to the test problem. They are :

1. In-plane uniaxial tension [see Figure 6.1] --- In this case, only the membrane component contribute to the stiffness matrix. The consistent load vector is due to the boundary traction S only. The REDUCE programs to the stiffness and the consistent load vector are presented as follows. In addition, the stress distribution can also be calculated symbolically.

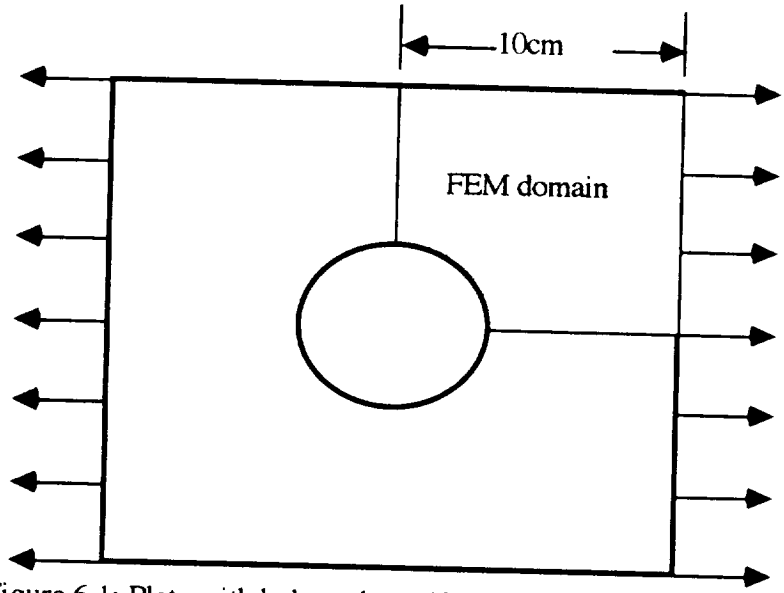

Figure 6.1: Plate with hole under uniform uniaxial tension load
100 N/cm**2

```
%************************************************************
% Symbolical program for making stiffness matrix for plate tension problem
%************************************************************
matrix bc(3,6),ln(6,15),d(3,3),b(3,15),s(15,15);
array n(3);
n(1):=s1;n(2):=s2;n(3):=s3;
ln(1,1):=1;ln(2,6):=1;ln(3,11):=1;ln(4,2):=1;
ln(5,7):=1;ln(6,12):=1;
bc:=mat((b1,b2,b3,0,0,0),(c1,c2,c3,b1,b2,b3),(0,0,0,c1,c2,c3));
d:=mat((1,0,v),(0,(1-v)/2,0),(v,0,1));
b:=bc*ln;
s:=tp(b)*d*b*pj*ye*h/(2*(1-v**2));
b1:=(y2-y3)/pj;b2:=(y3-y1)/pj;b3:=(y1-y2)/pj;
c1:=(x3-x2)/pj;c2:=(x1-x3)/pj;c3:=(x2-x1)/pj;
for i:=1:15 do <<a:=for j:=1:15 sum s(i,j);write a>>;
for i:=1:15 do <<c:=for j:=1:15 sum s(j,i);write c>>;
on fort;
off period;
off echo;
out "plane.ftn";
write "      subroutine ske(x1,y1,x2,y2,x3,y3,s)";
write "     dimension s(15,15)";
write "     implicit real*8(a-h,o-z)";
write "     pj=(x1-x3)*(y2-y3)-(y1-y3)*(x2-x3)";
write "     ye=2.1*1.0e07";
write "     v=0.29";
write "     h=0.2";
for i:=1:15 do for j:=1:15 do
if j>=i then write "      s(",i,",",j,")=",s(i,j)
else write "      s(",i,",",j,")=s(",j,",",i,")";
write "     return";
write "     end";
shut "plane.ftn";
bye;

%************************************************************
% Symbolic program to calculate the load vector for plate tension problem.
%************************************************************
matrix f(1,2),sn(2,15),fn(15,1);
array n(3),ff(15);
n(1):=s1;n(2):=s2;n(3):=s3;
f:=mat((f1,f2));
for i:=1 step 5 until 11 do <<sn(1,i):=n((i-1)/5+1);
sn(2,i+1):=n((i-1)/5+1)>>;
fn:=tp(f*sn);
s2:=0;
s3:=1-s1;
for i:=1:15 do <<a1:=int(fn(i,1),s1);a2:=sub(s1=1,a1)-sub(s1=0,a1);
fn(i,1):=a2*r1*h>>;
for i:=1;15 do ff(i):=fn(i,1);
on fort;
off echo;
off period;
```

```
out "15load.ftn";
write "      subroutine load(x1,y1,x2,y2,x3,y3,f1,f2,f3,fe)";
write "     implicit real*8(a-h,o-z)";
write "     dimension fe(15)";
write "     rl=sqrt((x1-x3)**2+(y1-y3)**2)";
write "     h=0.2";
for i:=1:15 do write "     fe(",i,")=",ff(i);
write "     return";
write "     end";
shut "15load.ftn";
bye;

%*****************************************
% REDUCE program to construct subroutine to compute
% stress distribution for plate tension problem.
%*****************************************
matrix d(3,3),bc(3,6),fn(6,1),stre(3,1);
array n(3);
n(1):=s1;n(2):=s2;n(3):=s3;
bc:=mat((b1,b2,b3,0,0,0),(c1,c2,c3,b1,b2,b3),(0,0,0,c1,c2,c3));
d:=mat((1,0,v),(0,(1-v)/2,0),(v,0,1));
operator u;
for i:=1:6 do fn(i,1):=u(i);
stre:=d*bc*fn*ye/(1-v**2);
b1:=(y2-y3)/pj;b2:=(y3-y1)/pj;b3:=(y1-y2)/pj;
c1:=(x3-x2)/pj;c2:=(x1-x3)/pj;c3:=(x2-x1)/pj;
on fort;
off echo;
off period;
out "st.ftn";
write "      subroutine stress(x1,y1,x2,y2,x3,y3,u,st)";
write "     implicit real*8(a-h,o-z)";
write "     dimension u(6),st(3)";
write "     v=0.29";
write "     pj=(x1-x3)*(y2-y3)-(y1-y3)*(x2-x3)";
write "     ye=2.1*1.0e07";
for i:=1:3 do write "     st(",i,")=",stre(i,1);
write "     return";
write "     end";
shut "st.ftn";
bye;
```

The resultant fortran subroutines obtained from the above programs to compute stiffness matrix, load vector and stress distribution are presented as follows. Since the expression of stiffness matrix is quite lengthy, only a part of it is showed. The interested researchers may refer to the PH. D. thesis of Wen-Lang Tsai [51] for details.

```
      SUBROUTINE SKE(X1,Y1,X2,Y2,X3,Y3,S)
     IMPLICIT REAL*8(a-h,o-z)
     DIMENSION S(15,15)
     pj=(x1-x3)*(y2-y3)-(y1-y3)*(x2-x3)
     YE=2.1*1.0E07
```

```
 V=0.29
 H=0.2
 s(1,1)=(H*YE*(V*X2**2-2*V*X2*X3+V*X3**2-X2**2+2*X2*
. X3-X3**2-2*Y2**2+4*Y2*Y3-2*Y3**2))/(4*PJ*(V**2-1))
 s(1,2)=(H*YE*(V*X2*Y2-V*X2*Y3-V*X3*Y2+V*X3*Y3+X2*Y2-
. X2*Y3-X3*Y2+X3*Y3))/(4*PJ*(V**2-1))
 s(1,3)=0
 s(1,4)=0
 s(1,5)=0
 s(1,6)=-(H*YE*(V*X1*X2-V*X1*X3-V*X2*X3+V*X3**2-X1*X2
. +X1*X3+X2*X3-X3**2+2*Y2*Y3-2*Y2*Y1-2*Y3**2+2*Y3*Y1)
. )/(4*PJ*(V**2-1))
 s(1,7)=-(H*YE*(2*V*X1*Y2-2*V*X1*Y3+V*X2*Y3-V*X2*Y1-2
. *V*X3*Y2+V*X3*Y3+V*X3*Y1-X2*Y3+X2*Y1+X3*Y3-X3*Y1))
. /(4*PJ*(V**2-1))
 s(1,8)=0
 s(1,9)=0
 s(1,10)=0
 s(1,11)=(H*YE*(V*X1*X2-V*X1*X3-V*X2**2+V*X2*X3-X1*X2
. +X1*X3+X2**2-X2*X3+2*Y2**2-2*Y2*Y3-2*Y2*Y1+2*Y3*Y1)
. )/(4*PJ*(V**2-1))
 s(1,12)=(H*YE*(2*V*X1*Y2-2*V*X1*Y3-V*X2*Y2+2*V*X2*Y3
. -V*X2*Y1-V*X3*Y2+V*X3*Y1-X2*Y2+X2*Y1+X3*Y2-X3*Y1))
. /(4*PJ*(V**2-1))
 s(1,13)=0
 s(1,14)=0
 s(1,15)=0
 s(2,1)=s(1,2)
 s(2,2)=(H*YE*(V*Y2**2-2*V*Y2*Y3+V*Y3**2-2*X2**2+4*X2
. *X3-2*X3**2-Y2**2+2*Y2*Y3-Y3**2))/(4*PJ*(V**2-1))
 s(2,3)=0
 s(2,4)=0
 s(2,5)=0
 s(2,6)=(H*YE*(V*X1*Y2-V*X1*Y3+2*V*X2*Y3-2*V*X2*Y1-V*
. X3*Y2-V*X3*Y3+2*V*X3*Y1-X1*Y2+X1*Y3+X3*Y2-X3*Y3))
. /(4*PJ*(V**2-1))
 s(2,7)=(H*YE*(V*Y2*Y3-V*Y2*Y1-V*Y3**2+V*Y3*Y1+2*X1*
. X2-2*X1*X3-2*X2*X3+2*X3**2-Y2*Y3+Y2*Y1+Y3**2-Y3*Y1))
. /(4*PJ*(V**2-1))
 s(2,8)=0
 s(2,9)=0
 s(2,10)=0
 s(2,11)=-(H*YE*(V*X1*Y2-V*X1*Y3+V*X2*Y2+V*X2*Y3-2*V*
. X2*Y1-2*V*X3*Y2+2*V*X3*Y1-X1*Y2+X1*Y3+X2*Y2-X2*Y3))
. /(4*PJ*(V**2-1))
 s(2,12)=-(H*YE*(V*Y2**2-V*Y2*Y3-V*Y2*Y1+V*Y3*Y1+2*X1
. *X2-2*X1*X3-2*X2**2+2*X2*X3-Y2**2+Y2*Y3+Y2*Y1-Y3*Y1
. ))/(4*PJ*(V**2-1))
   .
   .
 s(15,15)=0
 return
 end
```

```
 subroutine load(x1,y1,x2,y2,x3,y3,f1,f2,fe)
IMPLICIT REAL*8(a-h,o-z)
dimension fe(15)
rl=sqrt((x3-x2)**2+(y3-y2)**2)
 h=0.2
fe(1)=0
fe(2)=0
fe(3)=0
fe(4)=0
fe(5)=0
fe(6)=(H*F1*RL)/2
fe(7)=(H*F2*RL)/2
fe(8)=0
fe(9)=0
fe(10)=0
fe(11)=(H*F1*RL)/2
fe(12)=(H*F2*RL)/2
fe(13)=0
fe(14)=0
fe(15)=0
return
end

 subroutine stress(x1,y1,x2,y2,x3,y3,u,st)
implicit real*8(a-h,o-z)
dimension u(6),st(3)
 v=0.29
pj=(x1-x3)*(y2-y3)-(y1-y3)*(x2-x3)
YE=2.1*1.0E07
st(1)=(YE*(U(6)*V*X1-U(6)*V*X2-U(5)*V*X1+U(5)*V*X3+U
. (4)*V*X2-U(4)*V*X3+U(3)*Y2-U(3)*Y1-U(2)*Y3+U(2)*Y1-
. U(1)*Y2+U(1)*Y3))/(PJ*(V**2-1))
st(2)=-(YE*(U(6)*V*Y2-U(6)*V*Y1-U(6)*Y2+U(6)*Y1-U(5)
. *V*Y3+U(5)*V*Y1+U(5)*Y3-U(5)*Y1-U(4)*V*Y2+U(4)*V*Y3
. +U(4)*Y2-U(4)*Y3+U(3)*V*X1-U(3)*V*X2-U(3)*X1+U(3)*
. X2-U(2)*V*X1+U(2)*V*X3+U(2)*X1-U(2)*X3+U(1)*V*X2-U(
. 1)*V*X3-U(1)*X2+U(1)*X3))/(2*PJ*(V**2-1))
 st(3)=(YE*(U(6)*X1-U(6)*X2-U(5)*X1+U(5)*X3+U(4)*X2-U
. (4)*X3+U(3)*V*Y2-U(3)*V*Y1-U(2)*V*Y3+U(2)*V*Y1-U(1)
. *V*Y2+U(1)*V*Y3))/(PJ*(V**2-1))
return
end
```

The stress distribution of plate under tension is plotted by PATRAN in Figure 6.2. The stress concentration is visible in the top of the hole. One interesting phenomenon that should be mentioned here is that the stress at the middle top of the plate is less than the applied load. This is contributed by the bending effect which produces the compression in the top fiber of the plate.

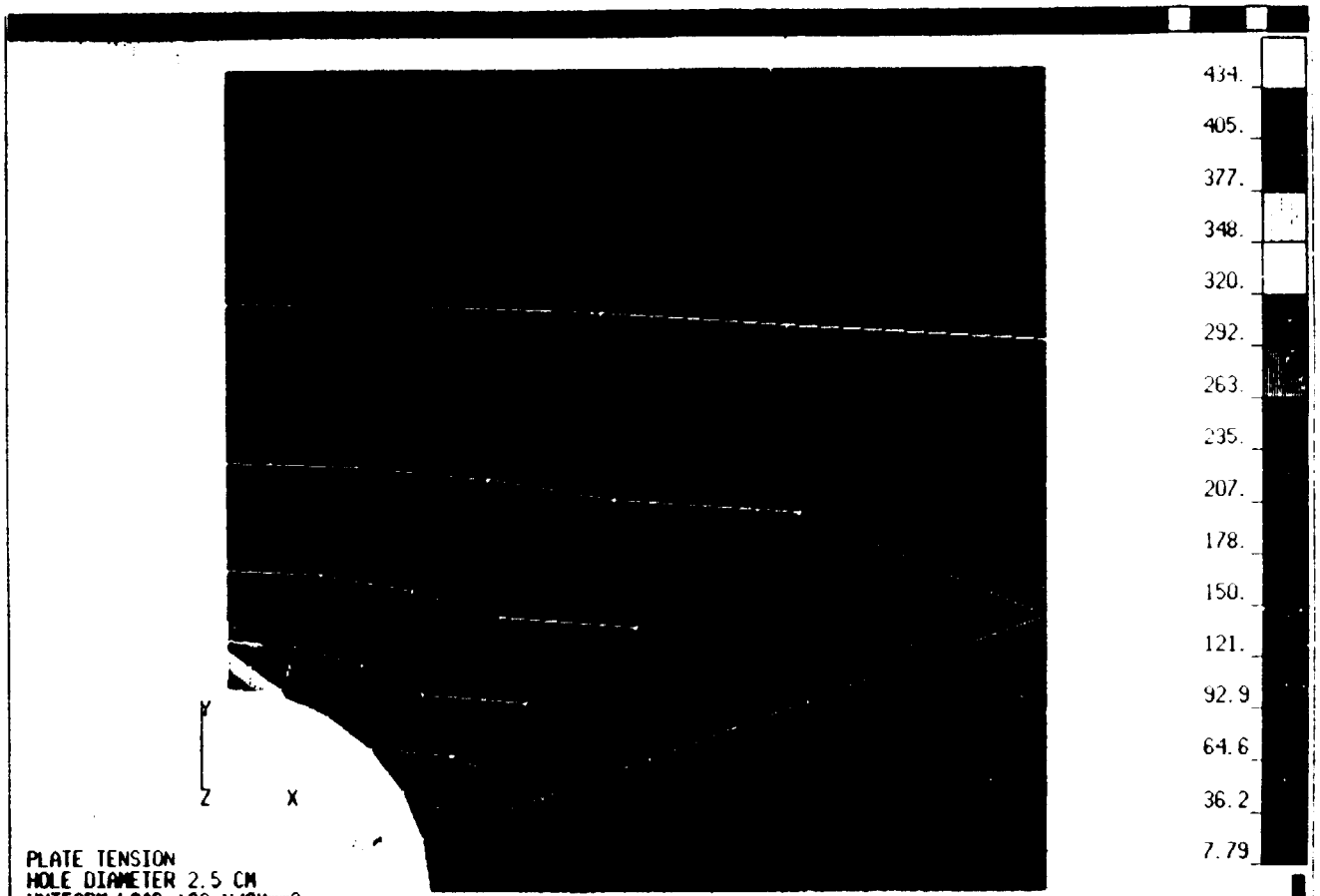

Figure 6.2 : Stress distribution of plate under uniaxial tension

2. Four-edge simply supported plate bending with uniform transverse load [see Figure 6.3] --- In this case, the membrane component of stiffness matrix is neglected. The REDUCE program to make the bending component of stiffness matrix is shown as follows :

```
%*********************************************************************
% REDUCE program for constructing plate bending stiffness matrix
%*********************************************************************
array n(9);
matrix sn(3,9),bc(2,3),bn(2,9),ssn(6,9),bcbc(3,6),bssn(3,9),d(3,3) ,ske(9,9),s(9,9);
n(1):=s1+s1*s2*(s1-s2)+s1*s3*(s1-s3);
n(2):=pj*(c3*(s1**2*s2+s1*s2*s3/2)-c2*(s1**2*s3+s1*s2*s3/2));
n(3):=pj*(b2*(s1**2*s3+s1*s2*s3/2)-b3*(s1**2*s2+s1*s2*s3/2));
n(4):=s2+s2*s3*(s2-s3)+s2*s1*(s2-s1);
n(5):=pj*(c1*(s2**2*s3+s1*s2*s3/2)-c3*(s2**2*s1+s1*s2*s3/2));
n(6):=pj*(b3*(s2**2*s1+s1*s2*s3/2)-b1*(s2**2*s3+s1*s2*s3/2));
n(7):=s3+s3*s1*(s3-s1)+s3*s2*(s3-s2);
n(8):=pj*(c2*(s3**2*s1+s1*s2*s3/2)-c1*(s3**2*s2+s1*s2*s3/2));
n(9):=pj*(b1*(s3**2*s2+s1*s2*s3/2)-b2*(s3**2*s1+s1*s2*s3/2));
for i:=1:9 do <<sn(1,i):=df(n(i),s1);sn(2,i):=df(n(i),s2);
          sn(3,i):=df(n(i),s3)>>;
bc:=mat((b1,b2,b3),(c1,c2,c3));
bn:=bc*sn;
for i:=1:9 do <<ssn(1,i):=df(bn(1,i),s1);ssn(2,i):=df(bn(1,i),s2);
   ssn(3,i):=df(bn(1,i),s3);ssn(4,i):=df(bn(2,i),s1);
   ssn(5,i):=df(bn(2,i),s2);ssn(6,i):=df(bn(2,i),s3)>>;
s3:=1-s1-s2;
bcbc:=mat((b1,b2,b3,0,0,0),(c1,c2,c3,b1,b2,b3),(0,0,0,c1,c2,c3));
bssn:=bcbc*ssn;
D:=MAT((1,0,V),(0,(1-V)/2,0),(V,0,1));
ske:=tp(bssn)*d*bssn*pj$
for i:=1:9 do for j:=1:9 do
if j>=i then <<tem:=int(ske(i,j),s2);
tem1:=sub(s2=1-s1,tem)-sub(s2=0,tem);
tem3:=int(tem1,s1);
s(i,j):=(sub(s1=1,tem3)-sub(s1=0,tem3))*ye*h**3/(12*(1-  v**2))>>
else s(i,j)=s(j,i);
b1:=(y2-y3)/pj;b2:=(y3-y1)/pj;b3:=(y1-y2)/pj;
c1:=(x3-x2)/pj;c2:=(x1-x3)/pj;c3:=(x2-x1)/pj;
on fort;
off echo;
off period;
CARDNO!*:=10;
out "z.ftn";
WRITE "     SUBROUTINE SKE1(X1,Y1,X2,Y2,X3,Y3,S)";
WRITE "    IMPLICIT REAL*8(A-H,O-Z)";
WRITE "    DIMENSION S(2,9)";
WRITE "    pj=(x1-x3)*(y2-y3)-(y1-y3)*(x2-x3)";
WRITE "    YE=2.1*1.0E07";
WRITE "    V=0.29";
WRITE "    H=0.2";
FOR I:=1:2 DO FOR J:=1:9 DO
```

```
IF j>=i THEN WRITE "      S(",I,",",J,")=",S(I,J)
  ELSE WRITE "      S(",I,",",J,")=S(",J,",",I,")";
WRITE "     RETURN";
WRITE "     END";
WRITE "      SUBROUTINE SKE2(X1,Y1,X2,Y2,X3,Y3,S)";
WRITE "     IMPLICIT REAL*8(A-H,O-Z)";
WRITE "      DIMENSION S(4,9),s1(2,9)";
WRITE "      pj=(x1-x3)*(y2-y3)-(y1-y3)*(x2-x3)";
WRITE "      YE=2.1*1.0E07";
WRITE "      V=0.29";
WRITE "      H=0.2";
WRITE "      CALL SKE1(X1,Y1,X2,Y2,X3,Y3,S1)";
WRITE "      DO 20 I=1,2";
WRITE "      DO 20 J=1,9";
WRITE "   20 S(I,J)=S1(I,J)";
FOR I:=3:4 DO FOR J:=1:9 DO
IF j>=i THEN WRITE "      S(",I,",",J,")=",S(I,J)
  ELSE WRITE "      S(",I,",",J,")=S(",J,",",I,")";
WRITE "     RETURN";
WRITE "     END";
WRITE "      SUBROUTINE SKE3(X1,Y1,X2,Y2,X3,Y3,S)";
WRITE "     IMPLICIT REAL*8(A-H,O-Z)";
WRITE "      DIMENSION S(6,9),s2(4,9)";
WRITE "      pj=(x1-x3)*(y2-y3)-(y1-y3)*(x2-x3)";
WRITE "      YE=2.1*1.0E07";
WRITE "      V=0.29";
WRITE "      H=0.2";
WRITE "      CALL SKE2(X1,Y1,X2,Y2,X3,Y3,S2)";
WRITE "      DO 20 I=1,4";
WRITE "      DO 20 J=1,9";
WRITE "   20 S(I,J)=S2(I,J)";
FOR I:=5:6 DO FOR J:=1:9 DO
IF j>=i THEN WRITE "      S(",I,",",J,")=",S(I,J)
  ELSE WRITE "      S(",I,",",J,")=S(",J,",",I,")";
WRITE "     RETURN";
WRITE "     END";
WRITE "      SUBROUTINE SKE(X1,Y1,X2,Y2,X3,Y3,S)";
WRITE "     IMPLICIT REAL*8(A-H,O-Z)";
WRITE "      DIMENSION S(9,9),s3(6,9)";
WRITE "      pj=(x1-x3)*(y2-y3)-(y1-y3)*(x2-x3)";
WRITE "      YE=2.1*1.0E07";
WRITE "      V=0.29";
WRITE "      H=0.2";
WRITE "      CALL SKE3(X1,Y1,X2,Y2,X3,Y3,S3)";
WRITE "      DO 20 I=1,6";
WRITE "      DO 20 J=1,9";
WRITE "   20 S(I,J)=S3(I,J)";
FOR I:=7:9 DO FOR J:=1:9 DO
IF j>=i THEN WRITE "      S(",I,",",J,")=",S(I,J)
  ELSE WRITE "      S(",I,",",J,")=S(",J,",",I,")";
WRITE "     RETURN";
WRITE "     END";
SHUT "z.ftn";
bye;
```

The resultant fortran subroutine is too large (around 135 pages) to be presented completely here. The following is only a small portion of it.

```
      SUBROUTINE SKE1(X1,Y1,X2,Y2,X3,Y3,S)
     IMPLICIT REAL*8(A-H,O-Z)
     DIMENSION S(1,9)
     pj=(x1-x3)*(y2-y3)-(y1-y3)*(x2-x3)
     YE=2.1*1.0E07
     V=0.29
     H=0.2
     ANS5=-4*Y3*Y1**3+2*Y1**4
     ANS4=-4*X2**2*Y2*Y1+8*X2**2*Y3**2+2*X2**2*Y1**2-16*
    . X2*X3**3-16*X2*X3*Y2**2+32*X2*X3*Y2*Y3-16*X2*X3*Y3
    . **2+5*X3**4+8*X3**2*Y2**2-16*X3**2*Y2*Y3+10*X3**2*
    . Y3**2-4*X3**2*Y3*Y1+2*X3**2*Y1**2+5*Y2**4-16*Y2**3*
    . Y3-4*Y2**3*Y1+24*Y2**2*Y3**2+6*Y2**2*Y1**2-16*Y2*Y3
    . **3-4*Y2*Y1**3+5*Y3**4-4*Y3**3*Y1+6*Y3**2*Y1**2+
    . ANS5
     ANS3=2*X1**4-4*X1**3*X2-4*X1**3*X3+6*X1**2*X2**2+6*
    . X1**2*X3**2+2*X1**2*Y2**2-4*X1**2*Y2*Y1+2*X1**2*Y3
    . **2-4*X1**2*Y3*Y1+4*X1**2*Y1**2-4*X1*X2**3-4*X1*X2*
    . Y2**2+8*X1*X2*Y2*Y1-4*X1*X2*Y1**2-4*X1*X3**3-4*X1*
    . X3*Y3**2+8*X1*X3*Y3*Y1-4*X1*X3*Y1**2+5*X2**4-16*X2
    . **3*X3+24*X2**2*X3**2+10*X2**2*Y2**2-16*X2**2*Y2*Y3
    . +ANS4
     ANS2=H**3*YE*ANS3
     ANS1=ANS2/(18*PJ**3*(V**2-1))
     S(1,1)= -ANS1
      .
```

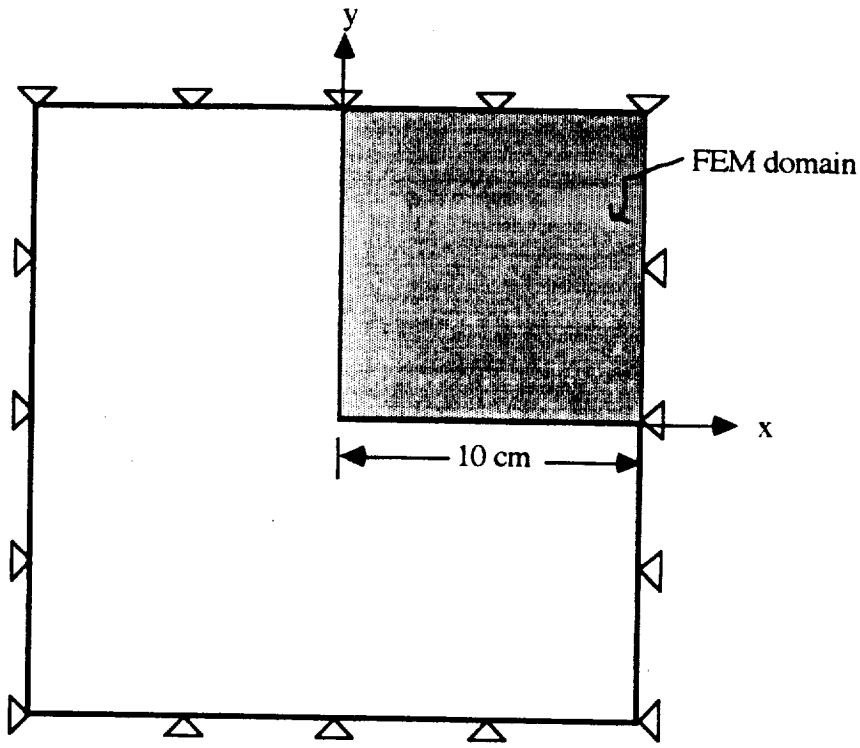

Figure 6.3 : Physical configuration of plate bending problem

The load vector in this case is for a uniform transverse pressure only. The REDUCE program and its output are shown as follows :

```
%***********************************************
% REDUCE program to construct the load vector
% for plate bending problem.
%***********************************************
array n(9),fe(9);
n(1):=s1+s1*s2*(s1-s2)+s1*s3*(s1-s3);
n(2):=pj*(c3*(s1**2*s2+s1*s2*s3/2)-c2*(s1**2*s3+s1*s2*s3/2));
n(3):=pj*(b2*(s1**2*s3+s1*s2*s3/2)-b3*(s1**2*s2+s1*s2*s3/2));
n(4):=s2+s2*s3*(s2-s3)+s2*s1*(s2-s1);
n(5):=pj*(c1*(s2**2*s3+s1*s2*s3/2)-c3*(s2**2*s1+s1*s2*s3/2));
n(6):=pj*(b3*(s2**2*s1+s1*s2*s3/2)-b1*(s2**2*s3+s1*s2*s3/2));
n(7):=s3+s3*s1*(s3-s1)+s3*s2*(s3-s2);
n(8):=pj*(c2*(s3**2*s1+s1*s2*s3/2)-c1*(s3**2*s2+s1*s2*s3/2));
n(9):=pj*(b1*(s3**2*s2+s1*s2*s3/2)-b2*(s3**2*s1+s1*s2*s3/2));
s3:=1-s1-s2;
for i:=1:9 do <<tem1:=int(n(i),s2);
            tem2:=sub(s2=1-s1,tem1)-sub(s2=0,tem1);
            tem3:=int(tem2,s1);
            fe(i):=(sub(s1=1,tem3)-sub(s1=0,tem3))*pj*f3;
            write i,fe(i)>>;
b1:=(y2-y3)/pj;b2:=(y3-y1)/pj;b3:=(y1-y2)/pj;
c1:=(x3-x2)/pj;c2:=(x1-x3)/pj;c3:=(x2-x1)/pj;
off period;
```

```
off echo;
ON FORT;
out "zload.ftn";
write "     SUBROUTINE LOAD(X1,Y1,X2,Y2,X3,Y3,F3,FE)";
write "     IMPLICIT REAL*8(A-H,O-Z)";
WRITE "     DIMENSION FE(9)";
write "     pj=(x1-x3)*(y2-y3)-(y1-y3)*(x2-x3)";
FOR I:=1:9 DO WRITE "     FE(",I,")=",FE(I);
WRITE "     RETURN";
write "     end";
shut "zload.ftn";
bye;
```

```
 SUBROUTINE LOAD(X1,Y1,X2,Y2,X3,Y3,F3,FE)
 IMPLICIT REAL*8(A-H,O-Z)
 DIMENSION FE(9)
 pj=(x1-x3)*(y2-y3)-(y1-y3)*(x2-x3)
 FE(1)=(PJ*F3)/6
 FE(2)=-(PJ*F3*(2*X1-X2-X3))/48
 FE(3)=(PJ*F3*(Y2+Y3-2*Y1))/48
 FE(4)=(PJ*F3)/6
 FE(5)=(PJ*F3*(X1-2*X2+X3))/48
 FE(6)=-(PJ*F3*(2*Y2-Y3-Y1))/48
 FE(7)=(PJ*F3)/6
 FE(8)=(PJ*F3*(X1+X2-2*X3))/48
 FE(9)=(PJ*F3*(Y2-2*Y3+Y1))/48
 RETURN
 end
```

The deformed shape is shown in Figure 6.4 and the stress distribution pattern is in Figure 6.5. In addition, three different sizes of mesh are tested to investigate the convergence of solution. They are shown in Figure 6.6. The convergence trend is presented in Figure 6.7 which is the plot of error vs. element mesh size.

3. Four-edge clamped plate bending under uniform load --- the only difference between this case and the last case is the boundary constraints. The extra slope constraints are enforced in the edges in this case. Therefore it is expected that the solution will be stiffer than that of the simply-supported case. The deformed shape is shown in Figure 6.8. The stiffer phenomenon is visible by comparing Figures 6.4 and 6.8.

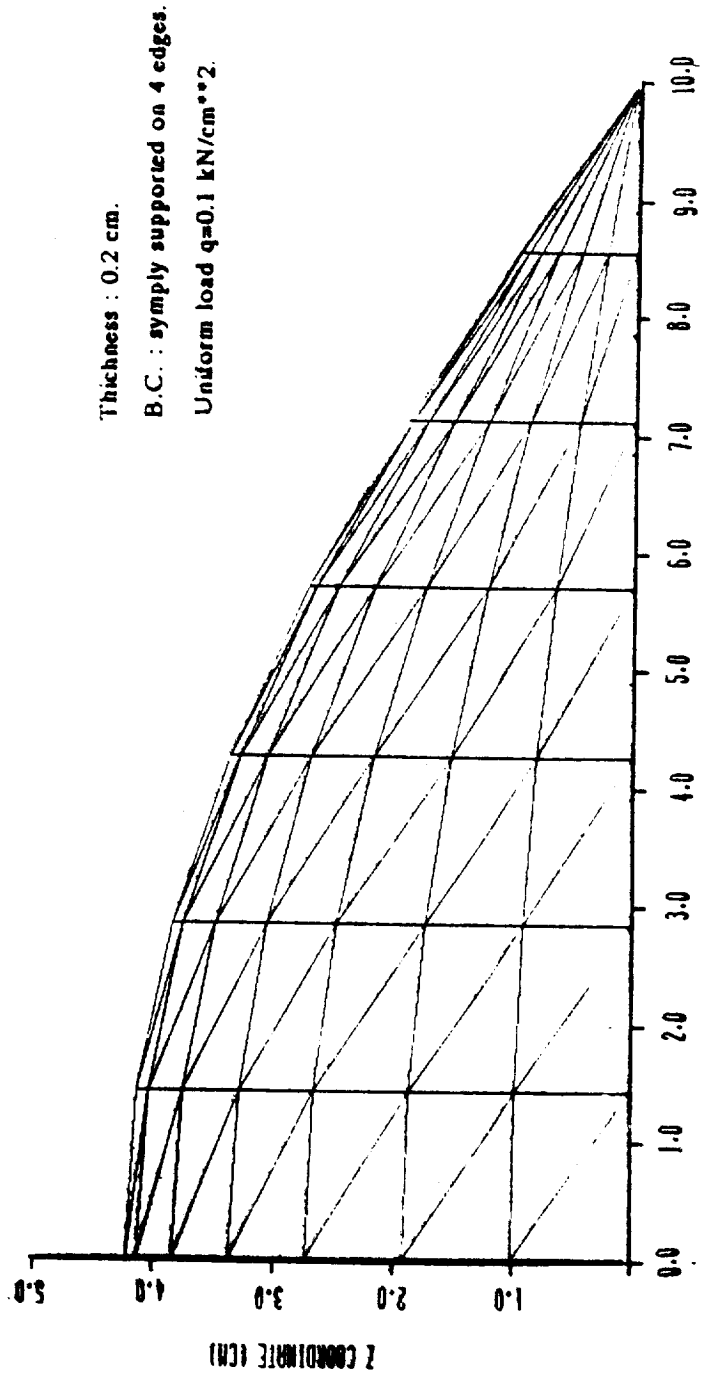

Figure 6.4 : Deformed shape of plate under bending load.

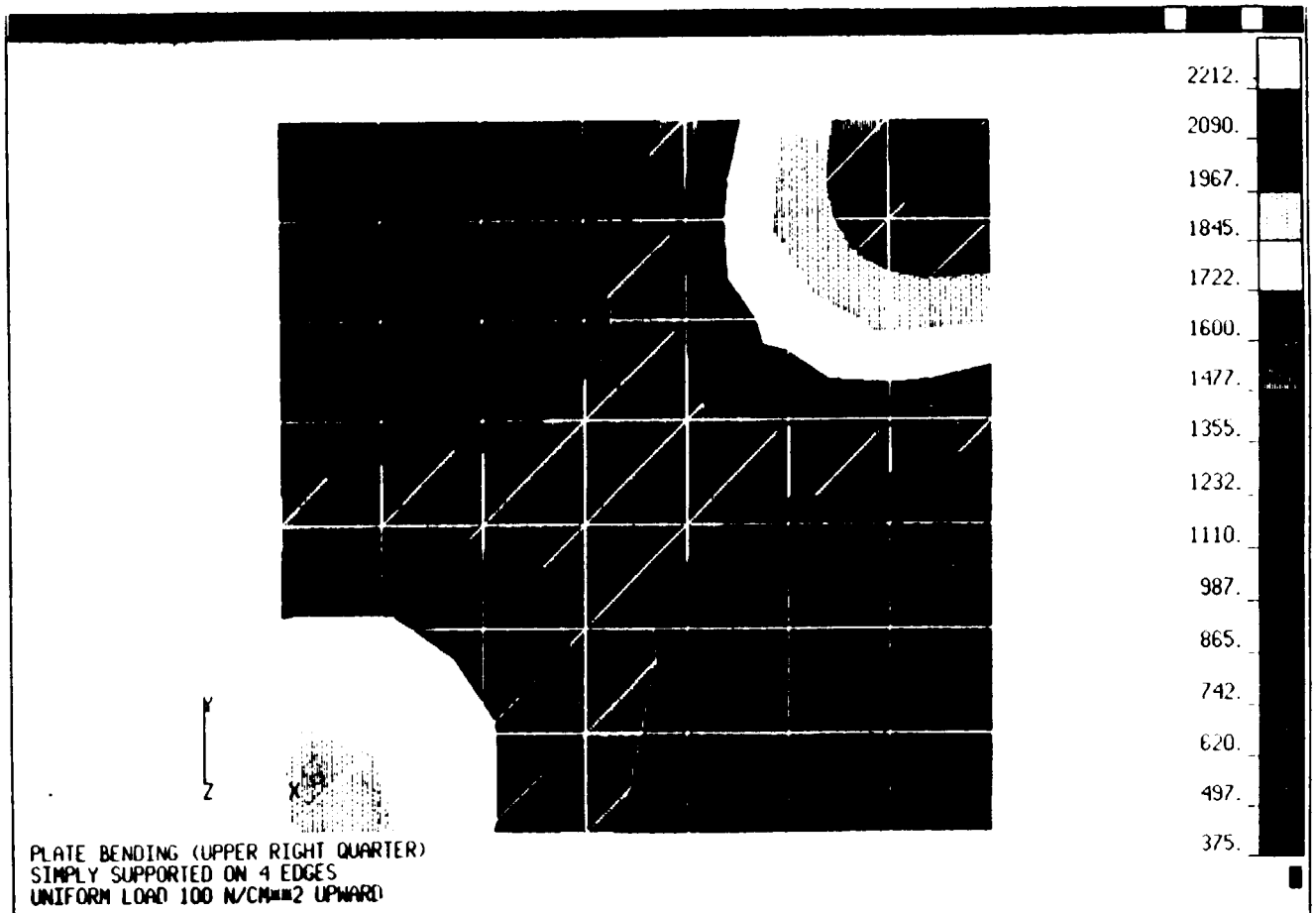

Figure 6.5 : Stress distribution of outer fiber of plate.

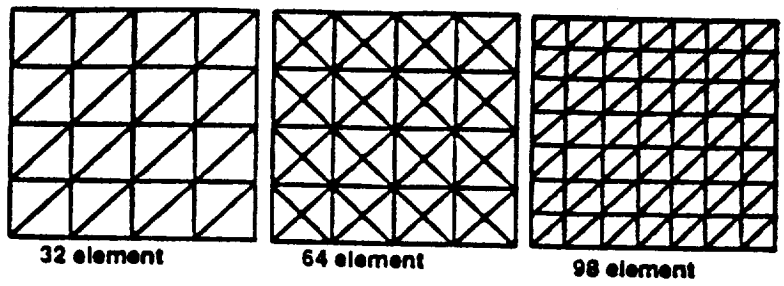

Figure 6.6 : Three different sizes of mesh for testing the convergence of solution of plate bending

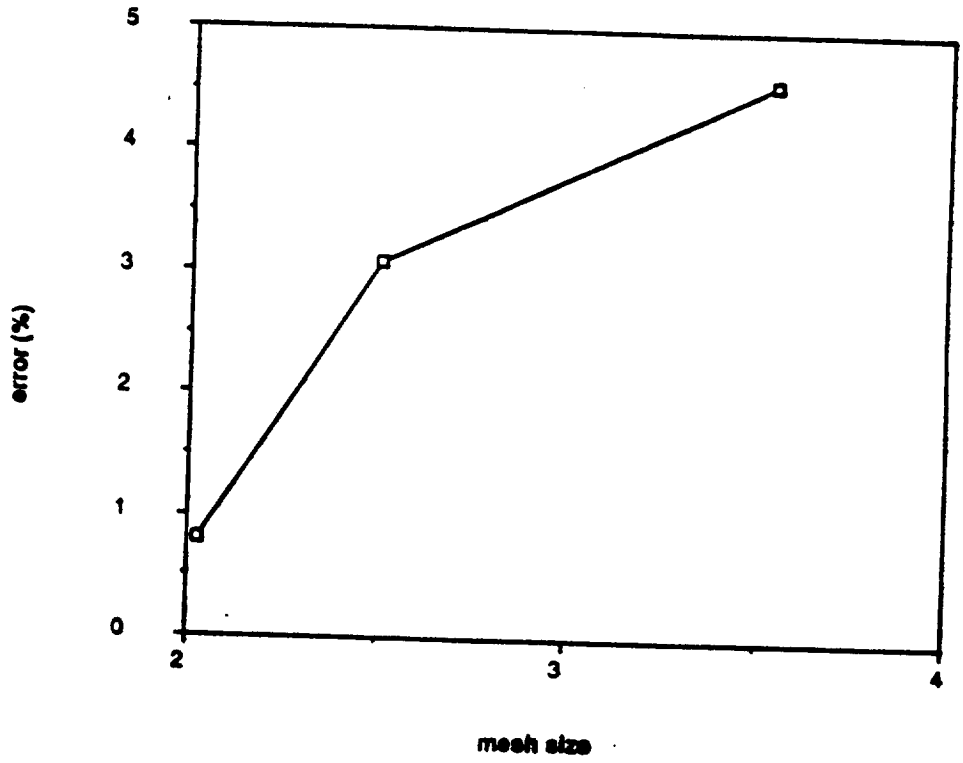

Figure 6.7 : Convergence of plate bending solution

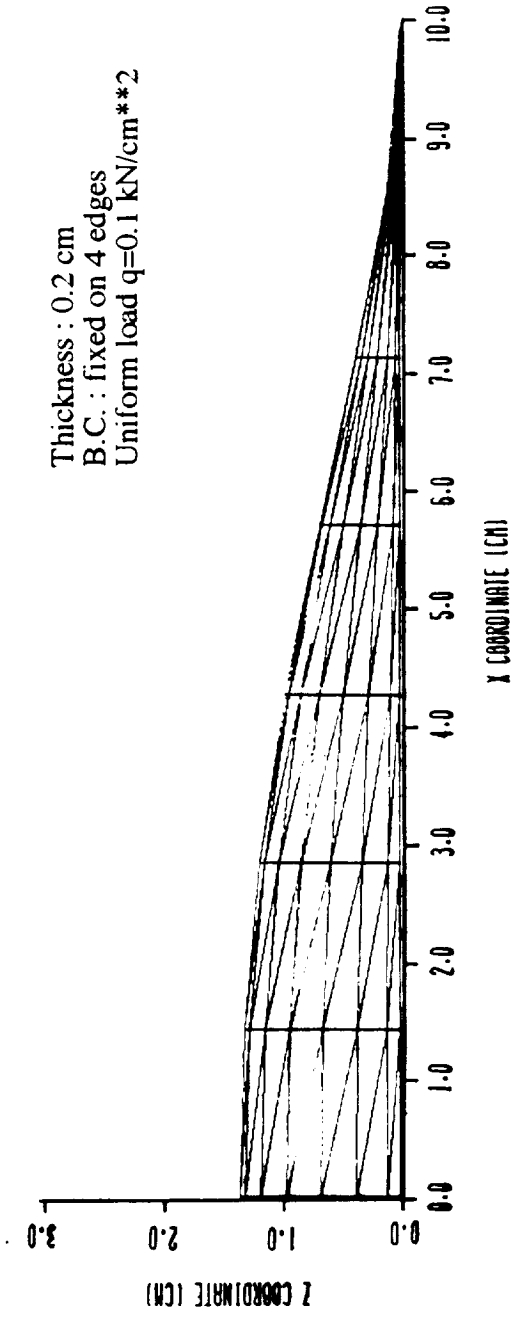

Figure 6.8 : Deformed shape of plate under uniform transverse load

6.5 References

[1] F. I. Niordson, "Shell theory", North-Holland, 1985.

[2] I. S. Sokolnikoff, "Tensor analysis", John Wiley & Sons,1964.

[3] M. Dikmen, "Theory of thin elastic shells", Pitman,1982.

[4] O. C. Zienkiewicz, "The finite element method", McGraw-Hill,1977.

[5] Eric Reissner, "The effect of transverse shear deformation on the bending of elastic plates", J. of applied mechanics, pp. A69-A77, June, 1945.

[6] R. D. Mindlin, "Influence of rotatory inertia and shear on flexural motions of isotropic, elastic plates", J. of applied mechanics, pp 31-38, March 1951.

[7] B. Budiansky, "Notes on nonlinear shell theory", J. of applied mechanics, pp 393-401, June 1968.

[8] O. C. Zienkiewicz, R. L. Taylor and J. M. Too, "Reduced integration technique in general analysis of plates and shells", Int. J. for nume. methods in engi., Vol.3, pp 275-290, 1971.

[9] O. C. Zienkiewicz, C. Parekh, I. P. King, "arch dams analysed by a linear finite element shell solution program", proc. of symposium, pp 19-22, London, March 1968.

[10] R. W. Clough and C. P. Johnson, "A finite element approximation for the analysis of thin shells", Int. J. solids structures, vol. 4, pp 43-60, 1967.

[11] Jean-Louis Batoz and M. B. Tahar, "Evaluation of a new quadrilateral thin plate bending element", Int. J. for numerical methods in engineering, vol. 18, pp 1655-1677, 1982.

[12] J. L. Batoz, "An explicit formulation for an efficient triangular plate-bending element", Int. J. for numerical methods in Engi., vol. 18, pp 1077-1089, 1982.

[13] J. L. Batoz, K. J. Bathe, Lee-Wing Ho, "A study of three-node triangular plate bending elements", Int. J. for numerical methods in Engi., Vol. 15, pp 1771-1812, 1980.

[14] B. M. Irons and A. Razzaque, "Shape function formulations for elements other than displacement models", proc. of conference at Uni. of Southampton, pp 4/59-4/72, sep. 1972.

[15] A. B. sabir and A. C. Lock, "A curved, cylindrical shell, finite element", Int. J. mech. Sci., vol. 14, pp 125-135, 1972.

[16] G. P. Bazeley, Y. K. Cheung, B. M. Irons, O. C. Zienkiewicz, "Triangular elements in plate bending - conforming and non-conforming solutions", pp 547-576, AFFDL-TR-66-80.

[17] D. G. Ashwell and A. B. Sabir, "A new cylindrical shell finite element based on simple independent strain functions", Int. J. mech. Sci., Vol. 14, pp 171-183, 1972.

[18] J. D. Chieslar and A. Ghali, "Solid to shell element geometric transformation", Computers & Structures, Vol. 25, No. 3, pp 451-455, 1987.

[19] J. D. Chieslar and A. Ghali, " A hybrid strain technique for finite element analysis of plates and shells", Computers & Structures, Vol. 24, No. 5, pp. 749-765, 1986.

[20] W. J. Sutcliffe and J. Mistry, "Shell segmentation requirements for numerical integration solutions", Computer methods in applied mechanics and engineering, Vol. 7, pp 179-190, 1976.

[21] Y. Yokoo and H. Matsunaga, "A general nonlinear theory of elastic shells", Int. J. solids Structures, vol. 10, pp 261-274, 1974.

[22] F. Par i's and S. De Le o'n, "Boundary element method applied to the analysis of thin plates", Computer & Structures, Vol. 25, No. 2, pp 225-233, 1987.

[23] P. C. Shen and J. G. Wan, "Vibration analysis of flat shells by using B spline functions", Computer & Structures, vol. 25, No. 1, pp 1-10, 1987.

[24] H. C. Huang, "Implementation of assumed strain degenerated shell elements", Computers & Structures, vol. 25, No. 1, pp 147-155, 1987.

[25] Kamal A. Meroueh, "On a formulation of a nonlinear theory of plates and shells with applications", Computers & Structures, Vol. 24, No. 5, pp 691-705, 1986.

[26] Kolbein Bell, "A refined triangular plate bending finite element", Int. J. for numerical methods in engi. Vol. 1, pp 101-122, 1969.

[27] N. Katz, A. G. Peano, M. P. Rossow, "Nodal variables for complete conforming finite elements of arbitrary polynomial order", Comp. and Maths with Appls., Vol. 4, pp 85-112, 1978.

[28] A. Peano, "Hierarchies of conforming finite elements for plane elasticity and plate bending", Comp. & Maths with Appls., Vol. 2, pp 211-224, 1976.

[29] A. Peano, "Conforming approximations for Kirchhoff plates and shells", Int. J. for nume. methods in engi., vol. 14, pp 1273-1291, 1979.

[30] I. M. Smith,"A finite element analysis for 'moderated-thick' rectangular plates in bending", Int. J. Mech. Sci., Vol. 10, pp 563-570, 1968.

[31] I. M. Smith and W. Duncan, 'The effectiveness of excessive nodal continuities in the finite element analysis of thin rectangular and skew plates in bending", Int. J. for numerical methods in engi., Vol. 2, pp 253-257, 1970.

[32] B. F. De Veubeke, "A conforming finite element for plate bending", Int. J. solids Structures, Vol. 4, pp 95-108, 1968.

[33] G. A. Butlin and R. Ford, "A compatible triangular plate bending finite element", Int. J. Solids Structures, Vol. 6, pp 323-332, 1970.

[34] T. J. Hughes, "A simple and efficient finite element for plate bending", Int. J. for nume. methods. in engi., Vol. 11, pp 1529-1543, 1977.

[35] Isaac Fried, "Shear in C^0 and C^1 bending finite elements", Int. J. Solids Structures, Vol. 9, pp 449-460, 1973.

[36] B. R. Somashekar, G. Prathap, C. R. Baru, "A field- consistent, four-noded, laminated, Anisotropic plate/shell element", Computer & Structures, Vol. 25, No. 3, pp 345-353, 1987.

[37] A. Razzaque, "Program for triangular bending elements with derivative smoothing", Int. J. for nume. methods in engi., Vol. 6, pp 333-343, 1973.

[38] F. K. Bogner, R. L. Fox, L. A. Schmit, Jr., "The generation of inter-element-compatible stiffness and mass matrices by the use of interpolation formulas", pp 397-423, AFFDL-TR-66-80.

[39] W. Weaver & P. R. Johnston, "Finite elements for structural analysis", prentice-Hall, 1984.

[40] S. Klein, "A study of the matrix displacement method as applied to shells of revolution", pp 275-298, AFFDL-TR-66-80.

[41] P. K. Mishra and S. S. Dey, "discrete energy method for the analysis of cylindrical shells", Computer & Structures, vol 27, No.6, pp 753-762,1987.

[42] J. H. Argyris, De, Fraes, "Matrix displacement analysis of anisotropic shells by triangular elements", J. of the royal aeronautical society, Vol. 68, pp 801-805, Nov. 1965.

[43] N. Kikuchi, "Finite element methods in mechanics", Cambridge,1986.

[44] A. E. Green, P. M. Nagdi & W. L. Wainwright, "A general theory of a cosserat surface", Archives of rational mechanics and analysis,p.287,vol. 20, 1965.

[45] S. Timoshenko & S.Woinowsky-Krieger, "Theory of plates and shells", 2nd edition,McGraw-Hill,1959.

[46] D. J. Dawe, "Rigid-body motions and strain-displacement equations of curved shell finite elements", Int. J. mech. sci., Vol. 14, pp 569-578, 1972.

[47] G. R. Cowper, G. M. Lindberg, M. D. Olson, "A shallow shell finite element of triangular shape", Int. J. Solids Structures, Vol. 6, pp 1133-1156, 1970.

[48] G. Cantin, "Rigid body motions in curved finite elements", AIAA Journal, Vol. 8, No. 7, pp 1252-1255, 1970.

[49] W. L. Tsai, "The investigations and experimentations on symbolic and algebraic manipulation software--REDUCE", unpublished, Uni. of Michigan, Aug.,1987

[50] R. S. Millman, G. D. Parker, "Element of differential geometry, Prentice-Hall,1977.

[51] W.L.Tsai, " Applications of symbolic and algebraic manipulation software in solving applied mechanics problems", Ph. D. thesis, Dept. of mechanical and applied mechanics, The University of Michigan, Ann Arbor, Dec. 1989.

CHAPTER VII

CONCLUSIONS

7.1 Introduction

The topics discussed in this chapter include the advantages of using symbolic and algebraic manipulation, the difficulties existing in running symbolic and algebraic software, the SAM in education, contributions, and the prospect of future development and application. Simple examples will be presented to illustrate the points where necessary.

7.2 Advantages of symbolic and algebraic manipulation

There are many advantages of application of symbolic and algebraic manipulation. They can be classified as follows :

1. Tireless power--- Together with human intelligence, the tireless capability in manipulating symbols and numbers has made SAM an indispensable tool in modern computational community. It has created the potential to challenge both previously intractable problems and new sophisticated formulae. Due to this advantage, the analytical work is pushed forward.

2. Accuracy --- A solution obtained by symbolic and algebraic manipulation is always exact. There is no round-off error accumulation.

3. Reliability --- The resultant expressions obtained by symbolic and algebraic manipulation will be correct if the input information is right. In addition, the capability of automatic code generation eliminates any typographic errors and substantially reduces the time in debugging the programs.

4. Efficiency --- This is a new advantage found in this research. The symbolic template in nonlinear numerical analysis can significantly improve the efficiency of program execution. This advantage is believed to be a crucial solution in the future for fields in

which the development time plays an important role. For example, the real time control will be one of them.

7.3 Internal swelling and mathematical limitations

As mentioned in the last chapters, there are some difficulties existing in the symbolic and algebraic manipulation. The followings are the detail discussions :

1. Memory capacity limitation --- A large amount of memory space is required for symbolic and algebraic manipulation. This is one of its fundamental limitations. The amount of memory space needed for running a symbolic and algebraic manipulator varies a lot from one system (hardware and software) to another. Different software systems need different sizes of memory space, and different hardware systems may provide different amounts of memory space for the same software. Even the same software running in the same hardware system sometimes may need different memory spaces depending on whether external packages are connected or not. For example, 834560 bytes RAM (about 815 K bytes) are currently provided (Fall,1989) for running REDUCE in the Michigan Terminal System (MTS) when it is invoked. If integration performance is involved in the computation, the external integration package should be manually included and the memory space will be extended to 1048560 bytes (about 1 mega bytes). The total memory space that MTS can provide during the computation is up to seven mega bytes. On the other hand, three mega bytes are provided to run REDUCE in an Apollo Domain workstation at the Computer Aided Engineering Network (CAEN) of The University of Michigan. Unlike MTS, this space can be automatically extended up to six mega bytes during execution if necessary. When the space requirement is beyond the provisions of hardware, execution will be aborted automatically. Therefore it is recommended that the symbolic and algebraic manipulator be implemented on a machine with at least one mega bytes RAM capacity to guarantee a safe execution.

To demonstrate the mechanism of internal swell in symbolic and algebraic manipulation, an example is given to calculate the factorial of a number. Mathematically, a factorial is defined as :

$$n! = \begin{cases} 1 & \text{if } n = 0 ; \\ n(n-1)! & \textit{otherwise} . \end{cases} \qquad (7.1)$$

The LISP function for this problem is as follows :

```
(defun factorial (n)
 (if (= n 0)
  1
  (* n (factorial (1- n)))))
```

When the function is called to calculate the factorial of four, the building process will occur first and then the collapsing process follows[9].

```
(factorial 4) -> (* 4 (factorial 3))
              -> (* 4 (* 3 (factorial 2)))
              -> (* 4 (* 3 (* 2 (factorial 1))))
              -> (* 4 (* 3 (* 2 (* 1 (factorial 0)))))
              -> (* 4 (* 3 (* 2 (* 1 1))))
              -> (* 4 (* 3 (* 2 1)))
              -> (* 4 (* 3 2))
              -> (* 4 6)
              -> 24
```

The internal swelling phenomenon occurs during the process of building. It is unquestionable that this phenomenon will become more serious if a larger number is given. Moreover if input number is negative, the recursive process will theoretically continue infinitely. Of course, the execution will be aborted when the provided space is used up.

2. Mathematical limitation --- Strictly speaking, a mathematical limitation should not be completely classified as limitation of symbolic and algebraic manipulation. For example, the analytical solution for the general 5th polynomial is proven to be non-existent.

---

[9] In some cases, the collapsing process may be impossible and the swelling phenomenon will last to the end of execution if it is not beyond the capacity of the hardware system.

Therefore it is also impossible for symbolic and algebraic manipulation to solve this problem. However, in addition to the above mentioned example, difficulties in solving mathematical equations whose analytical solutions exist are sometimes encountered. The occurrence of this phenomenon is quite system dependent. In general, most of such occurrences are encountered during integration and equation solving (algebraic and differential equation).

7.4 Symbolic and algebraic manipulation and education

The impact of SAM to science and engineering is significant. There are many publications of its application on celestial mechanics, relativity theory, and fluid mechanics. Compared to such successful applications, the response from educational community is far behind. So far, schools which officially include symbolic and algebraic manipulation in the content of courses include Cornell University, The University of Pennsylvania, and The University of Michigan. At Cornell University, MACSYMA was the first system introduced into the graduate-level course, in the winter of 1983. It was not until the fall of 1984 that Professor Richard Rand introduced the muMATH system into the sophomore engineering mathematics course. Unlike the MACSYMA system which ran on the mainframe, the muMATH system was implemented on IBM-XT and AT. At The University of Pennsylvania, Professor H. H. Bau employed MACSYMA in the instruction of approximate analyses. At The University of Michigan, Professor Noboru Kikuchi has used REDUCE to facilitate courses of finite element methods and applied mathematics since 1985. The other system, MATHEMATICA, was also implemented into the Macintosh II in the computational laboratory by Professor Kikuchi around 1988. Others such as Professor P. Papalambros and Professor R. Scott also used REDUCE in the courses on optimal design and finite element method.

The introduction of symbolic and algebraic manipulation into the education field should certainly be encouraged. So far, some critic views have been reported by students at The University of Michigan. They are :

1. Since there is no introductory course in symbolic and algebraic manipulation, students always struggle in learning the symbolic and algebraic manipulator itself rather than its application to the subject.

2. Most of manuals of symbolic and algebraic manipulators, such as REDUCE and MACSYMA, are unfriendly to the users. It is difficult for new users to understand the new terminologies in such a short time.

3. There is no appropriate textbook to facilitate to teach symbolic and algebraic manipulation and its application[10]. Unlike numerical analysis, the amounts of symbolic and algebraic manipulation results are not predictable. Therefore if assignment is not carefully designed, it could turn out just as simple as a symbol (say '0') or several hundred pages of outputs or nothing at all due to the internal swelling problem.

4. Qualified instructors are not easily found.

5. The software may not be fully operational. For instance, the MACSYMA system at The University of Michigan has just a half of its full capabilities. It is not easy to use because some functions cannot be found even when they are listed in the manual. The MATHEMATICA system is only implemented in some specific offices and is not yet available for public use.

In order to overcome these problems, some proposals are suggested as follows :

1. The education of symbolic and algebraic manipulation should start from the early undergraduate period. It is recommended that the existing "Numerical analysis" course be revised into "Numerical analysis and symbolic manipulation". The concept of symbolic manipulation, the use of available software, and the complementarity between symbolic manipulation and numerical analysis should be taught in the course.

2. It is urgent to design a textbook for such a new course. The existing manuals need be revised for easy accessibility.

3. The software systems should be rechecked and made available to the public.

7.5 Contributions of this study

The study presented in this report is believed to have made three original contributions to applied mechanics and symbolic manipulation. They are :

A). To applied mechanics :

1. Before this study, all of the advantages from the applications of symbolic and algebraic manipulation were either in handling lengthy formulae, or in increasing the accuracy of solution. In addition, this report points out for the first time a new advantage in

[10] The one written by Gerhand Rayna is rather an experimental book of REDUCE than an application of REDUCE in applied mechanics.

improving the efficiency of the execution of a numerical program. It is believed that this advantage will be crucial in such applications that the development time plays an important role. For instance, if the template is prepared in symbolic form beforehand and implemented in a chip, the data received by a heat-seeking missile can simply be substituted into the template. The response time will be substantially shortened.

2. The closed-form solution of a stiffness matrix of a 4-node quadrilateral isoparametric element was not available before. This dissertation presents the first analytical solutions of it. The contributions to the finite element analysis by this breakthrough are multifolds. First, the integration error is eliminated and the solutions are more accurate. Secondly, the closed-form solution can be automatically coded into a fortran subroutine. This allows the element stiffness matrix to be obtained by simple substitution of nodal coordinates. Of course, the fortran programming is simplified and the assemblage of the global stiffness matrix is expedited.

B). To Symbolic and algebraic manipulation :

3. Although the difficulties of the internal swelling problem and mathematical limitation were well known in the SAM field, no one has given the remedy for it. This report proposes a sysmatic pre-treatment method to avoid these difficulties and then successfully applies it to solve the problems.

7.6 Prospects and continuations of this research

As the criterion of the quality of results (in both industry and academia) becomes more and more strict, it is expected that more and more sophisticated formulations will be produced. Some expectations of future trends are as follows :

1. The design trend of symbolic and algebraic systems will continuously go towards smaller, more convenient packages for personal computers or even calculators. However, the mainframe SAM system will still co-exist to process large-expressions.

2. Applications of SAM in industry are scare at this time. However, this situation will change gradually after the teaching of symbolic and algebraic manipulation is actually implemented in schools.

3. The relationship between numerical analysis and symbolic manipulation will be smoother in the future. The switch from symbolic manipulation to numerical analysis (or vice versa) is expected to be automatic eventually

4. The gap between theoretical analysis and computational experiments will be smaller and smaller.

5. The reconsideration of every problem, equation, and formulation will become necessary. Regardless of whether they have already been solved or not. The solved problem can be used to check the correctness of solutions by SAM. The unsolved problem might then become solvable with the employment of SAM.

6. The inclusion of higher order terms for applied mechanics problems will become popular due to the availability of SAM systems.

7. To debug the symbolic program and to check the correctness of results are the important associated tasks of symbolic and algebraic manipulation. It is expected that the self debugging function of symbolic manipulators will be developed soon. A technique for the sysmatic checking of results from symbolic and algebraic manipulation should be available in the future.

The following three topics are closely relative to this study, and should be continued in future research. They are :

1. Extension of methodology in constructing a stiffness matrix for 2-D isoparametrical quadrilateral element to a 3-D problem.

2. Inclusion of the higher derivative term of $\frac{\partial^2}{\partial U^2}(\frac{\sigma}{F})$ in Equation (6.23) to Equation (6.25). This was neglected in the original formulation by Lee and Kobayashi and in this thesis.

3. Extension of methodology presented in section 7.3 to solve the general shell problem.

7.7 References

[1] P. H. Winston, B. K. P. Horn, "LISP", 2nd edition, Addison Wesley, 1984.

[2] Robert R Kessler, "LISP, objects and symbolic programming", Scott, Foresman and Company, 1988.

[3] Editor H. H. Bau, T. Herbert, "Symbolic computation in fluid mechanics and heat transfer", ASME HTD-Vol. 105, AMD-Vol. 97, 1988.

[4] P. H. Winston, "Artificial intelligence", 2nd edition, Addison Wesley, 1984.

APPENDIX A

Proof of Equation (5.12)

Given : $\tilde{f}(\tilde{V}_r) = -m * g * \frac{V_r}{|\tilde{V}_r|}\tilde{t}$

Prove : $\tilde{f}(\tilde{V}_r) \cdot \tilde{V}_r^* - \tilde{f}(\tilde{V}_r^*) \cdot \tilde{V}_r^* \geq 0$

Proof : Let m*g =1 for simplicity,

$$\begin{aligned}\Gamma &= \tilde{f}(\tilde{V}_r) \cdot \tilde{V}_r^* - \tilde{f}(\tilde{V}_r^*) \cdot \tilde{V}_r^* \\ &= -\left(\frac{v_r}{|\tilde{v}_r|}\tilde{t} \cdot V_r^*\tilde{t} - \frac{v_r^*}{|\tilde{v}_r^*|}\tilde{t} \cdot V_r^*\tilde{t}\right) \\ &= -\frac{V_r V_r^*}{|\tilde{V}_r|} + \frac{V_r^{*2}}{|\tilde{V}_r^*|}\end{aligned} \qquad (A.1)$$

There are three cases for discussions :

Case 1 : when $V_r^* > 0$,equation (A.1) will be

$$\Gamma = V_r^*\left(1 - \frac{V_r}{|\tilde{V}_r|}\right), \qquad \begin{cases} > 0, & \text{when } V_r \leq 0 \\ = 0, & \text{when } V_r > 0 \end{cases} \qquad (A.2)$$

Case 2 : when $V_r^* = 0$, equality is hold.

Case 3 : when $V_r^* < 0$, equation (A.1) will be

$$\Gamma = V_r^*\left(-1 - \frac{V_r}{|\tilde{V}_r|}\right), \qquad \begin{cases} > 0, & \text{when } V_r \geq 0 \\ = 0, & \text{when } V_r < 0 \end{cases} \qquad (A.3)$$

Therefore, $\tilde{f}(\tilde{V}_r) \cdot \tilde{V}_r^* - \tilde{f}(\tilde{V}_r^*) \cdot \tilde{V}_r^* \geq 0$ is proven

APPENDIX B

Proof of Equation (5.16)

Given : $\tilde{f}(\vec{V}_r) = -m * g * \frac{V_r}{|\vec{V}_r|}\tilde{i}$

Prove : $\int_0^{|\vec{V}_r^*|} \tilde{f} \cdot d\vec{V}_r - \int_0^{|\vec{V}_r|} \tilde{f} \cdot d\vec{V}_r \le \tilde{f}(\vec{V}_r) \cdot (\vec{V}_r^* - \vec{V}_r)$ (B.1)

Proof : Let m*g =1 for simplicity,

$$\int_0^{|\vec{V}_r^*|} \tilde{f} \cdot d\vec{V} = \int_0^{|\vec{V}_r^*|} -\frac{V_r}{|\vec{V}_r|} dV_r \tilde{i} \cdot \tilde{i} = \int_0^{|\vec{V}_r^*|} -sign(V_r) dV_r \qquad (B.2)$$

$$If\ V_r > 0 \Rightarrow \int_0^{|\vec{V}_r^*|} \tilde{f} \cdot d\vec{V}_r = -|V_r^*|, \quad \int_0^{|\vec{V}_r|} \tilde{f} \cdot d\vec{V}_r = -|V_r| \qquad (B.3)$$

$$If\ V_r < 0 \Rightarrow \int_0^{|\vec{V}_r^*|} \tilde{f} \cdot d\vec{V}_r = |V_r^*| \quad \int_0^{|\vec{V}_r|} \tilde{f} \cdot d\vec{V}_r = |V_r| \qquad (B.4)$$

There are four cases for discussions :

1. $V_r^* > 0, V_r > 0$ case :

Left side of (B.1)= $-|V_r^*| + |V_r| = -V_r^* + V$

$= -\frac{V_r}{|\vec{V}_r|}(V_r^* - V_r)$ =right side of (B.1)

2. $V_r^* > 0, V_r < 0$ case :

Left side of (B.1)= $|V_r^*| - |V_r| = V_r^* + V_r$

$\le -\frac{V_r}{|\vec{V}_r|}(V_r^* - V_r)$ =right side of (B.1)

3. $V_r^* < 0, V_r > 0$ case :

Left side of (B.1)= $-|V_r^*| + |V_r| = V_r^* + V$

$$\leq -V_r^* + V_r = -\frac{V_r}{|V_r|}(V_r^* - V_r) = \text{right side of (B.1)}$$

4. $V_r^* < 0, V_r < 0$ case :

Left side of (B.1)$= |V_r^*| - |V_r| = -V_r^* + V$

$$= -(V_r^* - V_r) = -(-\frac{V_r}{|V_r|})(V_r^* - V_r)$$

=-1*(right side of (B.1)) (B.5)

Equation (B.5) implies that equality is hold and both sides of (B.1) are zero for case 4.

Therefore equation (B.1) is proven.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1993	Contractor Report

4. TITLE AND SUBTITLE

Application of Symbolic and Algebraic Manipulation Software in Solving Applied Mechanics Problems

5. FUNDING NUMBERS

NCA2-136

6. AUTHOR(S)

Wen-Lang Tsai and Noboru Kikuchi

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Michigan
Department of Mechanical Engineering
Ann Arbor, MI 48109

8. PERFORMING ORGANIZATION REPORT NUMBER

A-93112

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR-4544

11. SUPPLEMENTARY NOTES

Point of Contact: Hiro Miura, Ames Research Center, MS 237-11, Moffett Field, CA 94035-1000; (415) 604-5888

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified — Unlimited
Subject Category 31

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

As its name implies, symbolic and algebraic manipulation is an operational tool which not only can retain symbols throughout computations but also can express results in terms of symbols.

This report starts with a history of symbolic and algebraic manipulators and a review of the literatures. With the help of selected examples, the capabilities of symbolic and algebraic manipulators are demonstrated. These applications to problems of applied mechanics are then presented. They are the application of automatic formulation to applied mechanics problems, application to a materially nonlinear problem (rigid-plastic ring compression) by finite element method (FEM) and application to plate problems by FEM. At the end of the report, the advantages and difficulties, contributions, education and perspectives of symbolic and algebraic manipulation are discussed.

It is well known that there exist some fundamental difficulties in symbolic and algebraic manipulation, such as internal swelling and mathematical limitation. This report proposes a remedy for these difficulties, and successfully solves the three applications mentioned above. For example, the closed from solution of stiffness matrix of four-node isoparametrical quadrilateral element for 2-D elasticity problem was not available before. Due to the work presented here, the automatic construction of it becomes feasible. In addition, a new advantage of the application of symbolic and algebraic manipulation found in this study is believed to be crucial in improving the efficiency of program execution in the future. This will substantially shorten the response time of a system. It is very significant for certain systems, such as missile and high speed aircraft systems, in which time plays an important role.

14. SUBJECT TERMS

Symbolic manipulation, Applied nonlinear mechanics, Finite element analysis

15. NUMBER OF PAGES

164

16. PRICE CODE

A08

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000