Development of a user-friendly interface for the creation of user elements

Kinshuk
German National Research Center for Information Technology
GMD-FIT, Schloss Birlinghoven
D-53754 Sankt Augustin, Germany
Phone : (49) 2241 14 2144 \
Fax: (49) 2241 14 2065
EMail : Kinshuk@gmd.de

International Journal of Electrical Engineering Education
33(4), October 96, pp344-352 (ISSN 0020-7209)

Abstract

The solution of the basic equations relating stiffness, displacement and force in finite element analysis is not trivial, partially due to the large numbers of equations involved. Various techniques have been developed for this purpose and commercial finite element packages like *Ansys* resolve the equations and store the results quite efficiently and have large element libraries. If a particular element formulation is not found in the library of *Ansys*, the users can develop their own elements with the help of *Fortran* routines provided with *Ansys*. It should be emphasised that these routines are not at all user friendly and require considerable work every time a new type of element is needed.

This paper describes the development of a *user friendly* program by interfacing the mathematical package *Mathematica* with *Ansys* to make the user element capability more or less generalised, and more efficient.

Keywords : Ansys, Finite element analysis, Interface, Mathematica, Shape functions, Stiffness matrix, Symbolic computation, User elements

Introduction

While doing a finite element analysis, different type of "finite elements" are needed for different problems according to the size, shape and properties of the

problem structure. User elements are the elements created by the users for their own special needs. These elements are generally dedicated to the particular problem for which they have been created.

Commercial packages available in the market usually contain a very rich library of various types of elements to facilitate most of the general finite element applications. *Ansys* is an example of those packages. It has an extensive element library but also gives the facility to create new user elements, in case the users do not find suitable elements in the library for their specific applications or if they wish to try new element formulations, for example, when there is a special degree of freedom attached to the nodes of the element, or new type of behaviour needs to be attached to the element such as plasticity, creep, swelling, etc.

User elements in Ansys

While creating new user element, information is given to *Ansys* about the special properties desired in that element. It is done by using PREP7, the *Ansys* general database pre-processor, whose primary objective is to organise analysis input data and write the formatted analysis datafile, FILE27.

To the user element programmer, the consequences of using PREP7 to create the analysis file are as follows :

a) In order to create plots of the user element, PREP7 must have access to the information regarding the element shape (e.g. beam, shell, solid..., linear, triangular, quad, etc.)

b) User element data input to PREP7 must be checkable.

Each element type has a unique spreadsheet defined for it within PREP7 which expects different information as the element type dictates, such as number of nodes, degree of freedom set at each node etc. To provide information for the spreadsheet required by the pre-processor for the user element, some *Fortran* routines are available with *Ansys* and the user has to put specific values for some variables in those routines. When those routines are compiled with *Ansys*, it creates the desired element and uses it for the specified problem.

Mathematica - Ansys interface

An interface is provided between finite element package *Ansys* and the mathematical package *Mathematica* to facilitate the creation of user elements with less effort and in a *user friendly* environment, in which more help is given to the user when making inputs.

Whenever a new element type is to be defined, major alterations are required in the *Fortran* user routines. It is a very tedious process since all numerical values have to be changed one by one in each routine.

The task can be made simpler by changing all numerical values into variables and reading those variables from some text files. The text files contain all the numerical values of the variables in the necessary order.

These text files may be created by *Mathematica*, which takes input of all those numerical values from the user. A very useful feature of *Mathematica* has been used namely Notebooks. Notebooks are like electronic books. One can do some calculation work while reading it. It is very useful front end to take input from users giving them various alternatives in the form of help.

A major problem in *Fortran* is the size and complexity of various matrices used in those subroutines. Also, since recursive programming is not allowed in *Fortran*, we have to use various DO loops for even very simple matrix generation program. It increases the size of routines enormously. The problem has been solved by calculating all those matrices in *Mathematica* which is very simple. After calculation, all the results are stored in the text files so that *Fortran* routines can directly read them and can use without calculation. This saves lots of hard and soft memory along with CPU time.

Programming the interface in Mathematica

The *Mathematica* routines of the interface are basically for two purposes :

(a) Getting input from user.

(b) Creating text files for use in *Fortran* routines.

There are six separate routines in the *Mathematica* program which get various inputs in order to calculate various parameters :

(a) *Main[]* :- This routine is the root of program from where each routine is called when needed, and expects the routine DMATRIX[] which is called from Stiff[].

(b) *Userel[]* :- This routine is based on the *Fortran* routine USEREL, and saves various inputs in "userel.txt" as follows :

*) 2-D / 3-D geometry of the element and structure

*) Degrees of freedom sets for each node of element

*) Symmetric / Unsymmetric matrix

*) Set of rotation of degrees of freedom

*) Number of nodes in the element

*) Number of temparatures / heat generation rates during analysis

*) Number of pressures applied to the edge or surface of the element

*) Number of real constants such as thickness and moment of inertia, to be specified in analysis

*) Number of variables to be saved for future use, at the time of result interpretation

*) Number of rows in element matrices

*) Linear / Non-linear element

*) Stress / Thermal analysis

(c) *Userpt[]* :- This routine is based on the *Fortran* routine USERPT, and saves the input of the shape of the element to be plotted in "userpt.txt".

(d) *Usermh[]* :- This routine takes input for the *Fortran* routine USERMH and also interrogates the user about the material properties, which the user wants to specify. The inputs, saved in "usermh.txt" are as follows :

*) Shape of the element to be used for meshing at the time of analysis.

*) Existence of midside nodes in the element.

*) Occurrence of warning messages when meshes are made entirely of triangles or tetrahedrons.

*) Various material properties, specified by the user at the time of analysis.

(e) *Stiff[]* :- This routine has been developed for symbolic calculation of shape functions and stiffness matrices for triangular element of n^{th} order. The order will depend on the number of nodes desired in the element by the user.

The routine takes input for following values :

*) Number of nodes which are inputted at the time of routine call.

*) State of case (i.e. Plane Stress or Plane Strain) is directly input by user.

The routine outputs computed values as text files. The following are the text files containing the output from the routine :

*) The values of local co-ordinates in the form of xy.txt

*) Shape functions in the form of shape.txt

*) Multiplication of the constituent matrix D and the shape functions matrix B in the form of db.txt

*) Stiffness matrix in the form of stiff.txt :

A routine has been written to calcuate stiffness matrix, which outputs the matrix in symbolic form. The logic used in the routine is available from the author on request. There are some limitations of the routine. Since this is designed solely for triangular elements, the number of nodes must suit the triangular elements. Otherwise program will give error message and will quit. It does not take into account quadrilateral elements because of the differences in the structure of shape functions of these elements.

Special care should be taken while using cubic or higher order elements, as the program takes into account 'in-the-element' nodes. This implies that the number of valid nodes for a cubic element will be 10 and not 9, the 10th node being present at the centroid of the triangle.

(f) *DMATRIX[]* :- This routine calculates the constitutive matrix symbolically for the desired status, i.e. Plane Stress or Plane Strain. Separate modules are provided for both cases. Young's modulus, Poisson's ratio and key for the

status are the inputs to the routine and are via subroutine call. The output of constitutive matrix is via 'Return' command.

Transformation of text files into Fortran format

Since the output text files, created by *Mathematica*, are in symbolic expression form, it is difficult to use them in *Fortran*, as *Fortran* needs files in a certain format. Therefore a *Fortran* routine has been devised to convert the text files into desired *Fortran* format. This reads the file, puts a variable name in the beginning of each expression, writes the whole expression using continuation marks and does this process till the end of the file.

Fortran routines

The next step in the development of the *Mathematica* - *Ansys* interface is the use of *Fortran* routines provided by *Ansys,* to create the user element, with all desired properties for user's specific applications.

These routines need some numerical values for various items. These routines are modified to make them generalised. Now these routines receive input from the various text files. Each time, the user can change the element by just replacing the old text files with new ones.

There are many routines provided with *Ansys* for various specialised purposes. The subroutines, which were used in this interface, are :-

(i) *USEREL* :- This subroutine is used to define parameters for the *Ansys* user element. The inputs for this subroutine are via two modes :-

(a) Subroutine call -

ITYP = Type of element. This parameter is used in ET command of *Ansys* which specifies the type of element to be used in the analysis.

IPARM = Array of integers for input.

KYSUB = Array of 9 KEYOPT values.

(b) Taking input from the file "userel.txt" which contains the numerical values of various parameters, for which variables are used in the routine to make the routine flexible.

The routine outputs various parameters as follows :-

IPARM = Array of integers for output.

KEY3D = Key for 2-D or 3-D geometry of the element.

KDOF = Key for DOF selected for each node of the element.

KUNSYM = Key for unsymmetric matrices which may be the case while creating incompatible elements.

KTRANS = Key for type of transformation needed for the degrees of freedom while applying boundary conditions.

(ii) *USERPT* :- This subroutine is for *Ansys* plot shape for display purposes. The inputs for this subroutine are via two modes of data transfer :-

(a) Subroutine call -

INODE = Array of node numbers.

KYSUB = Array of first three KEYOPT values.

(b) Reading "userpt.txt" file which contains the numerical value of the variables used in this routine.

The output of the routine, the value of variable KSHAPE, which is the key representing the type of shape of the element, is via the subroutine call.

(iii) *USERMH* :- This subroutine is to decide shape of the element in the mesh module of the analysis. Inputs for this routine are via subroutine calls and file reading. Input via the subroutine call is a variable KEY1, the only keyopt available to the user element programmer for interfacing with the mesh module. Variable values are being read from "usermh.txt" file.

Outputs are given via the subroutine call :-

LABSH = A character label showing the allowable shapes for the element, which can be used to determine the shape of element in the mesh module.

MID = A logical variable which is .TRUE. if the element has midside nodes. This may be the case, for example, if there are six or more nodes present in a triangular element.

WARN = A logical variable, used for LABSH = 'QDTRI ' or 'BKTET ' only, which is true if meshes made entirely of triangles or tetrahedra are to be discouraged by warning messages.

(iv) *ST100* :- This routine is the stiffness pass for the user element; it calculates the stiffness matrix and passes it to main program for further processing. Data is transferred into ST100 in the following manner :-

(a) Subroutine call -

IELNUM = Element number as being processed.

ITYP = Type of element used by *Ansys* command ET, as explained earlier.

KELIN = Vector of keys if matrices are to be computed.

NR = Final matrix size (number of non-zero rows).

KTIK = Dimensioned matrix size (max. = 60)

(b) COMMON/STCOM where file STCOM contains some variables declarations which are used in this routine.

(c) CALL GETED2 to take data from file FILE03.DAT which contains various properties of the element.

(d) CALL PROPE1 to evaluate the material properties specified by the user.

(e) Reading the text file "stiff.txt" which contains the terms of the stiffness matrix already been calculated by *Mathematica* symbolically.

Data is transferred out of ST100 using the following procedure :-

(a) Subroutine call -

KELOUT = Vector of keys if matrices have been computed.

ZS = Stiffness matrix.

ZASS = Mass matrix.

DAMP = Damping matrix.

GSTIF = Stress stiffness matrix.

ZSC = Force vector.

(b) CALL PUTED2 to put data into file FILE02.DAT which contains the data regarding element matrices to be used in solution phase.

(c) Printed error messages in case of any mishaps.

(v) *SR100* :- This routine is the stress pass for *Ansys* user element; it calculates the stresses and displacements for each element and stores this in the postdata

file for use in post-processing phase. Data is transferred into SR100 with the following procedure :-

(a) Subroutine call - Various parameters such as stiffness matrix, keys for matrix calculation indication, type of element etc. are passed into the routine via subroutine calls.

(b) COMMON / STCOM

(c) CALL GETED2

(d) CALL STRESS1 to calculate stresses and pass the result to the routine SR100.

(e) CALL KSHAPE3 to extrapolate stresses at node points and is used only when the stresses are calculated using Gauss sampling points.

(f) Reading text files.

Data is transferred out of SR100 with the following procedure :-

(a) Subroutine call

(b) CALL PUTED2

(c) Printed output

(d) Postdata file

(vi) *STRESS1* :- This subroutine is for calculating stresses; it is to be called by the SR100 subroutine. The inputs and outputs of the routine are via subroutine

calls. The [D] and [B] matrices multiplication matrix and displacement matrix are input to this routine and stress matrix is the resultant output.

(vii) *KSHAPE3* :- This routine is used only when stresses are calculated by Gauss extrapolation method; it uses stresses on three Gauss points and extrapolates stresses on required nodes with the help of these Gauss points stresses. Inputs to the routine are via two modes :-

(a) Subroutine call -

KXX and KYY = Local co-ordinates of the node at which the stresses are to be calculated. These are calculated by *Mathematica* and are stored in "xy.txt".

GSX3, GSY3 and GTAU3 = Stress matrices at Gauss sampling points.

(b) Reading text file "shape.txt" which contains shape functions.

Outputs from the routine are the stresses at the desired node in the form of matrix and are via subroutine call.

Creation of user element

The text files, created after conversion of *Mathematica* created text files to *Fortran* acceptable format, are then compiled with modified *Fortran* routines to get an object file for *Ansys*. This object file is then linked into the *Ansys* code by using the INSTALL.USER file, which is supplied by *Ansys* supplier for this purpose. After compilation and linking, the file "ansys.e" is obtained which contains all information about the desired user element.

Discussions and conclusions

The objective of this research has been to develop an interface between *Mathematica* and *Ansys* to provide a friendly environment to the user to create new user elements in the *Ansys*. This has been achieved successfully by providing various alternatives, at the time of taking inputs from the user. This is done by using the *Mathematica* front end, Notebooks.

This interface has been verified by using the interface routines to create three-node and six-node elements. The behaviour of these elements has also been analysed by using them to compute stresses and displacements of some arbitrary specified problems.

As a conclusion, it can be said that interfacing between *Mathematica* and *Ansys* has proved very useful to provide a very friendly environment to the user for creating new user elements. It takes only a few minutes to create any new user element using the developed interface, while it was a tedious work of several days in the past whereas formerly the user had to change many numerical values in several complex *Fortran* routines provided by the *Ansys*, in the *Mathematica* approach, the user is not even aware about these *Fortran* routines.

Further areas of research work

The existing interface can be extended in various ways to get more efficient user elements with various benefits. Some of these could be as follows :-

*) The limitations on the size of various arrays restricts the program to calculate various parameters for elements with higher number of nodes. Another major

limitation is to use constant size of arrays instead of adjustable arrays which are required at many places in the routines.

*) Until now, only stiffness matrix calculation is done with the help of *Mathematica*. Further extensions could be done by formulating mass matrix, load vectors, geometric stiffness terms and thermal effects.

*) It is assumed in the existing program that each node must have the same number of degrees of freedom. In practice, this may not be the case. Formulations could be developed to include the cases where each node of an element has unique set of degrees of freedom. Similarly, one more restriction in the current routines is that each degree of freedom is represented by the same shape function. Routines could be modified to take into account this restriction.

Acknowledgement

The author wishes to thank Prof. J. T. Boyle for his constructive comments and to acknowledge the use of *Ansys* and *Mathematica* software packages under academic licences.

References

M.J.Fagan. Finite Element Analysis - Theory and Practice. Longman Sc. & Tech. 1992.

O.C.Zienkiewicz and R.L.Taylor. The Finite Element Method. 4th ed. vol.1. McGraw Hill.

Neville Ian Ash. Maths At The Touch Of A Button. Professional Engineering, 6(6). June93. pp17-18.

M.M.Cecchi and C.Lami. Automatic Generation Of Stiffness Matrices For Finite Element Analysis. Int.J.for Num.Math.in Engg. vol.11. 1977. pp396-400.

A.R.Korncoff and S.J.Fenves. Symbolic Generation of Finite Element Stiffness Matrices. Compu. and Stru., 10. 1979. pp119-124.

ANSYS - Engineering Analysis System - User's Manual. vol. I&II (Swanson Analysis Systems, Inc.).

User's Guide for Microsoft Windows - Mathematica. July 1992 (Wolfram Research).

Roman Maeder. Programming in Mathematica. Addison-Wesley.

Peter P.Silvester. Computer Algebra in Finite Element Software Construction. Int.J.for Num.Meth.in Engg. vol.3 pp169-177 (1992).

Robert D.Cook, David S.Malkus and Michael E.Plesha. Concepts and Applications of Finite Element Analysis. 3rd ed. John Wiley.