Vietnam Journal of Mathematics **32:** *SI* (2004) 77–86

Vietnam Journal
of
MATHEMATICS

FEPG - A Powerful Tool for Generating Finite Element Programs

Liang Guoping[1], Wu Bangxian[2], and Xie Hui[3]

[1] *Institute of Mathematics, Chinese Academy of Sciences, China*
[2] *Institute of Engineering Thermophysics, Chinese Academy of Sciences, China*
[3] *Beijing Fegen Software Co. Ltd. Beijing, China*

Abstract. FEPG is a newly invented software system, which can automatically generate a finite element program based on the given PDE and algorithm expressions by using a special finite element language. It has been used in scientific research, teaching of finite element course and the various industry applications for over ten years. The practice has verified that it is a powerful tool for generating finite element programs. Following a brief introduction of FEPG, its main functions and features are described. The special finite element language is explained. Two typical examples with attached FEPG files, including one nonlinear problem and one coupled fields problem are demonstrated. Finally a conclusion is drawn.

1. Introduction

Finite element method (FEM) has been widely accepted as a useful analytical tool by scientists and engineers in different physical fields since its invention in 1950's. However, its more broad use has been restricted by the complexity of a finite element program and the difficulty of its use. A finite element program with a rather complete function usually has at least thousands lines of codes, which would take a couple of man-year of labor to complete. On the other hand, a lot of commercial codes of FEM appeared on the market can only partly meet the varieties of needs of scientists and engineers. Even a general-purpose commercial program can only solve a particular scope of problems. Clients usually need to make some changes or to add some codes in the original program according to their special problems. But it is almost impossible to realize without the participation of the original code writer. In fact, the more generic the program is, the more difficult to understand and use it is, the more data users should input.

On the other hand, the more specific the program is, the more easily users use it, but the more restricted the problems solved are. That is why FEM cannot be extended to a much more extent as expected. The aim of developing the Finite Element Program Generator (FEPG) is to help scientists and engineers to build their own FEM programs or systems, which is easy to read, maintain and modify. Part 2 of this paper introduces the main functions and features of FEPG system. Part 3 describes the finite element describing language (FEDL), which is used in FEPG system to automatically generate FEM program. Parts 4 and 5 illustrate how to write partial differential equations and algorithms in FEPG system through two examples, including the solution of a non-linear magnetic field problem and the solution of coupled fields problem. Part 6 is conclusion.

2. Functions and Features of FEPG

The essential of FEPG lies in adopting both component programming technique and human intelligence technology to automatically generate finite element source code from the given PDE and algorithm expressions. It is based on the unified mathematical theory for finite element method and its intrinsic rule, in which the program generation is similar to the deduction process of mathematical formulae.

FEPG is an open software platform for Finite Element Analysis (FEA) and Computer Aid Engineering (CAE). Users only need to input all expressions and formulae required for FEM by use of FEDL defined in the system, then FEPG can automatically generate complete source code, including element subroutines, algorithm programs, etc. In this way, a lot of time for code writing can be saved (Compared with normal FORTRAN program for the same problem, the code size can be reduced to about one tenth. Thus, the programming work can be completed within a couple of days or even hours that would otherwise take several months or even years). The correctness and the consistence of the program can be guaranteed as well. Users have a free and convenient choice for embedding any new solvers. By use of FEPG, scientists can concentrate their efforts in more creative work, such as the improvement of the physical model, and the algorithm.

FEPG is suitable for solving finite element problems in various engineering and scientific fields. The application areas range from petrochemical engineering, machinery, energy, transportation, electronic engineering, civil engineering, shipbuilding, biomedical engineering, aeronautical and aerospace engineering, etc. It is especially suitable to scientific research in solving complicated, nonlinear, coupled, multi-physics problems as well as to teaching of FEM course in colleges. It is a major breakthrough to the limitation of current general-purpose finite element softwares, which are only suitable for particular fields and particular problems.

FEPG also set up a formulae library for standard PDE and algorithm, which is helpful for the beginners or user with simple problems. The complete programs can be directly obtained from the formulae library by clicking the menu of AppWizard on the toolbar of FEPG interface. It can also be used as the basis

for modification for solving non-standard problems.

Clients can use network version of FEPG (IFEPG) anywhere and anytime via Internet to get complete source code within only a few seconds. A lot of expenditure in software purchasing, upgrading and maintenance can be saved by using IFEPG.

3. Finite Element Describing Language (FEDL)

FEPG utilizes a kind of script language, which is used by clients to define a finite element problem to be solved, including PDE and algorithm expressions. It is very close to the professional describing language appeared in scientific journals and books for discussing finite element methods and finite element problems. It is concise and clear, but only suitable to FEM, thus can be called as finite element describing language (FEDL). FEDL consists of two parts. The first part is used to describe PDE expressions. In doing so, the system requires the client to use this language to write the PDE in the weak form (WF) in a file called PDE file with its extension name of pde. Another part of FEDL is used to describe the finite element algorithm for solving PDE, including how to linearize a non-linear PDE, how to discretize the time variable for a transient problem, as well as the computation and iteration process.

4. Example 1 - Non-Linear Magnetic Field

The analysis on an axisymmetric electromagnetic field for a DC crane can be described by solving a nonlinear magnetic field problem. The counterpart of the WF equation for the PDE is as follows:

$$(\frac{r}{\mu}\nabla \times A,\ \nabla \times \delta A) = (rJ, \delta A),$$

where A is the magnetic potential in the direction of 'o' in 'roz' cylinder coordinates, δA is the virtual displacement of A, μ is magnetic conductivity. J is the density of the source current in the 'o' direction, $(\cdot, \cdot)$ represents inner product. The axisymmetric vorticity operator '$\nabla\times$' is expressed as:

$$\nabla \times A = \begin{pmatrix} -\frac{\partial A}{\partial z} \\ \\ \frac{\partial A}{\partial r} + \frac{\partial A}{z} \end{pmatrix}.$$

For a $2D$ non-linear magnetic field problem, μ is a function of $|B|$ and is expressed as $\mu(|B|)$, where $B = (B_r, B_z)$ is the magnetic flux density, $B_r = -\dfrac{\partial A}{\partial z}, B_z = \dfrac{\partial A}{\partial r} + \dfrac{A}{r}$. By using Newton iteration method, we get the following iteration scheme

$$\begin{aligned} &(\frac{r}{\mu}\nabla \times w,\ \nabla \times \delta A) - (\frac{r}{\mu^2|B|}\frac{\partial \mu}{\partial |B|}(\nabla \times A.\nabla \times w), (\nabla \times A.\nabla \times \delta A)) \\ &= (rJ, \delta A) - (\frac{r}{\mu^2|B|}\frac{\partial \mu}{\partial |B|}(\nabla \times A.\nabla \times A), (\nabla \times A.\nabla \times \delta A)), \end{aligned}$$

where $w = A + \Delta A$ is the value at the present iteration step. μ B takes the value at the previous iteration step. $\dfrac{\partial \mu}{\partial |B|}$ is obtained by numerical differentiation on the test data curve.

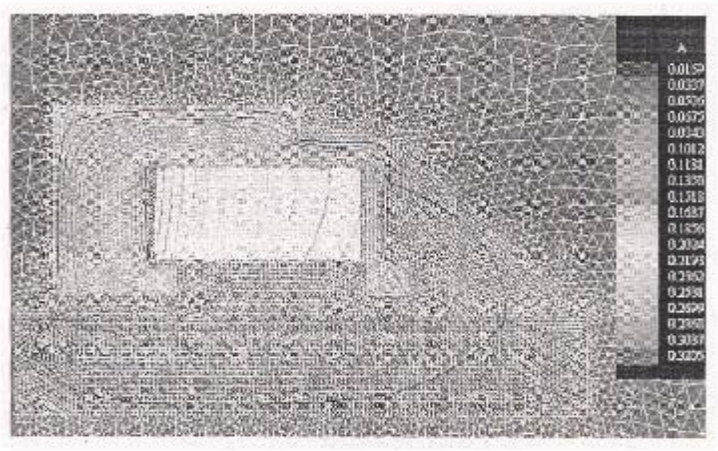

Fig. 1. Contour of magnetic potential A

FEPG can quickly generate the finite element program based on PDE file and GCN file written by use of FEDL. The PDE file for calculating magnetic potential in weak form is as follows (the content behind the symbol '\' is the explanation, which is not included in the file.

In the following, pe represents μ, fu represents J, gmu represents $\dfrac{\partial \mu}{\partial |B|}$.

```
disp u,                  \name of unknown function, here is magnetic potential
coor r,z,                \coordinate variables
coef un,                 \value at the previous iteration step
func funa,funb,func,     \of the auxiliary unknown functions
shap t  3                \shape function, triangle element
gaus m                   \select midpoint of each edge for numerical integral
mass t  r                \mass matrix expression
$c6 common /data/ bmag(10,1000),hmag(10,1000),amu(10,1000)
$c6 common /ndata/ nmag(10),narea
mate pe fu 1.256637d-6 0.0d0              \material names and their data
func
funa=+[u/r]+[u]/r                         \B_z
funb=+[u/z]                               \ - B_r
func=+[u/r]*{un/r}+[u/r]*un/r+[u]/r*{un/r}+[u]/r*un/r
+[u/z]*{un/z}                             \ ∇ × A.∇ × w
stif                                      \stiff matrix section
$c6 fxy=1.0
$c6 if (num.eq.1.and.igaus.eq.1) then
```

```
\The following is to read the test data and get μ, B and gmu
$c6 open(197,file='mate.geo',form='formatted',status='old')
$c6 read(197,*) nmag
$c6 do imag=1,nmag
$c6 read(197,*) bmag(imag),hmag(imag)
$c6 amu(imag)=bmag(imag)/hmag(imag)
$c6 enddo
$c6 close(197)
$c6 endif
$cv fxx={un/r}+un/r
$cv fyy={un/z}
$c6 fxy=dsqrt(fxx**2+fyy**2)                    \get |B|
$c6 if (fxy.lt.1.e-6) fxy=1.e-6
$c6 call fmu(nmag(iarea),iarea,fxy,bmag,amu,xmu,gmu) \calculate μ and gmu
$c6 pe=xmu
dist =
+[funa;funa]/pe*r
+[funb;funb]/pe*r
-[func;func]*gmu/pe/pe/fxy*r                    \the stiff matrix of the WF equation
load =+[u]*fu*r
-[func]*gmu*fxy/pe/pe*r                         \RHS of the WF equation

end

fort
      subroutine fmu(nmate,bmo,bli,ali,xmu,gmu)
      implicit real*8 (a-h,o-z)
      dimension bli(1000),ali(1000)
      do 100 i=1,nmate
      if (bmo.le.bli(i)) goto 200
100   continue
200   continue
        if (i.gt.nmate) then
        xmu=ali(nmate)
        gmu=0.0
        return
      endif
      if (i.eq.1) then
      b1=0.
      a1=ali(1)-(ali(2)-ali(1))/
      *(bli(2)-bli(1))*bli(1)
      b2=bli(1)
      a2=ali(1)
      else
      b1=bli(i-1)
      a1=ali(i-1)
```

```
b2=bli(i)
a2=ali(i)
endif
xmu=((b2-bmo)*a1+(bmo-b1)*a2)/(b2-b1)
gmu=(a2-a1)/(b2-b1)
return
end
```

Using Least Squae Method to get the magnetic flux density. Its PDE and algorithm files are not listed here.

A GCN file defining the algorithm is listed below:

```
defi
a nell &      \field a (magnetic potential) adopt the non-linear elliptic algorithm in the library.
b str a       \field b(magnetic flux density ) adopt the least squae method in the library.
\The following is the computation procedure by command flow
startsin a                          \initialize field a at this line
IF EXIST END DEL END                \following is the nonlinear iteration process
:1
solvsin a                           \solve field a
IF NOT EXIST END GOTO 1
stress b                            \use Least Squa Method to solve field b
```

5. Example 2 - Coupling Between Thermal Field and the Solid Mechanics

In this example, three physical fields are included, that is, temperature field, displacement field and stress field. The stress field will be obtained from the displacement field. Here the equation for temperature field and the displacement field are given.

5.1. Thermal Conduction Equation

$$ec(u) * \frac{\partial u}{\partial t} - \nabla(ek(u)\nabla u) = eq,$$

where u represents temperature, $ec(u)$ is the product of the specific heat and the density of the material, $ek(u)$ is the conductivity, eq is the intensity of the inner heat source. $ec(u)$ and $ek(u)$ change with temperature.

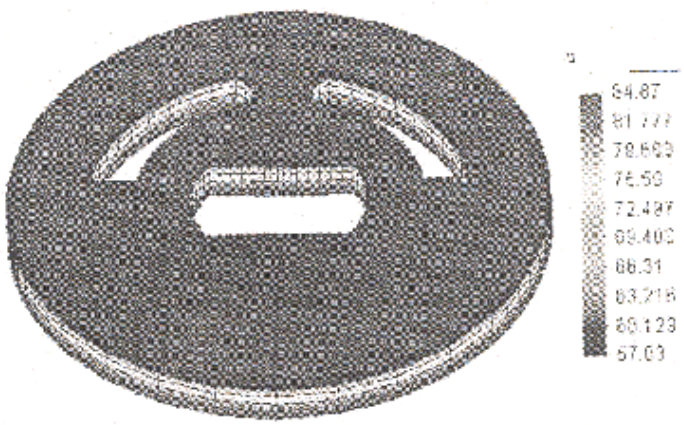

Fig. 2. Contour fill of temperature at certain time step.

5.2. Displacement Field

Equilibrium equations

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + \frac{\partial \sigma_{xz}}{\partial z} + f_x = 0,$$
$$\frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{yz}}{\partial y} + f_x = 0,$$
$$\frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} + f_z = 0.$$

Geometric equations

$$\varepsilon_{xx} = \frac{\partial u}{\partial x} \qquad \varepsilon_{yy} = \frac{\partial v}{\partial y}, \qquad \varepsilon_{zz} = \frac{\partial w}{\partial z},$$
$$\varepsilon_{yz} = \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}, \quad \varepsilon_{xz} = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}, \quad \varepsilon_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}.$$

Constitutional relations

$$\sigma = D\varepsilon - \beta,$$

where

$$D = \frac{E}{(1+\nu)(1-2\nu)} \begin{vmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5-\nu & 0 & \\ 0 & 0 & 0 & 0 & 0.5-\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5-\nu \end{vmatrix},$$

$$\beta = \frac{E\alpha T_n}{(1-2\nu)}(1 \;\; 1 \;\; 1 \;\; 0 \;\; 0 \;\; 0)^T,$$

where E is elastic modulus, ν is the Poisson's ratio, α is the coefficient of thermal expansion. The counterpart of the WF equation is written as:

$$\int_V \delta\varepsilon^T D\varepsilon dV = \int_V \delta u^T f dV + \int_V \delta\varepsilon^T \beta dV.$$

FEPG can quickly generate the finite element program based on PDE file and GCN file written by use of FEDL.
The PDE file for calculating temperature field in the WF equation is as follows (behind '\'is an explanation, not included in the file).

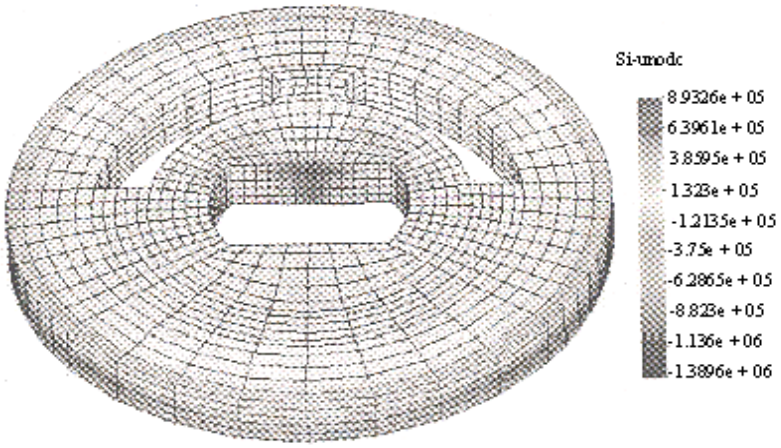

Fig. 3. Contour fill of the first first principle stress at certain time step

```
disp u                                  \name of unknown function, here is temperature
coor x y z                              \coordinate variables
coef un                                 \temperature value at the previous iteration
step
shap c 8                                \shape function, cube element
gaus c                                  \select node point for numerical integral
mass c ec                               \mass matrix expression
mate ek ec eq 1.0d01;0.1d0;0.0d0;       \material names and their data
vect x x y z                            \coordinate vector x

stif                                    \stiff matrix section
$c6 call eku(un,ek)                     \get the value of nonlinear coefficient ek
$c6 call ecu(un,ec)                     \get the value of nonlinear coefficient ec
dist=+[u/x_i;u/x_i]*ek                  \expression for stiff matrix
load=+[u]*eq                            \RHS of the WF equation
end
fort
\The following subroutines calculate nonlinear coefficient ek,ec
      subroutine eku(tn,ek)
```

```
  implicit real*8 (a-h,o-z)
    ek=0.017251*tn+11.8817
return
end
subroutine ecu(tn,ec)
implicit real*8 (a-h,o-z)
  ec=7.92d03*(0.00011684*tn*tn+0.12807*tn+498.5275)
return
end
```

In the following, pe represents E, pv represents v, alpha represents α.
The PDE file for solving displacement field

```
disp u,v,w                          \names of displacement variables
coor x,y,z                          \coordinate variables
func ex ey ez eyz exz exy           \names of the auxiliary unknown functions for strain
shap c 8                            \shape function, cube element
gaus c                              \select node point for numerical integral
coef tn                             \temperature
mate pe pv alfa fx fy fz 1.0e9;0.3;1.0e-5;0;0;0;    \material names and their data
vect em ex ey ez                    \direct strain vector em
vect en eyz exz exy                 \shear strain vector en
vect u u v w                        \displacement vector u
vect fd fx fy fz                    \body force vector fd
vect fe ft*tn ft*tn ft*tn           \thermal strain vector fe
matr de 3 3                         \matrix of constitutional relation de
(1.-pv) pv pv
pv (1.-pv) pv
pv pv (1.-pv)

FUNC
$c6 fact = pe/(1.+pv)/(1.-2.*pv)
$c6 ft=(1.+pv)*fact*alfa
ex=+[u/x]                           \direct strains ex, ey, ez
ey=+[v/y]
ez=+[w/z]
eyz=+[w/y]+[v/z]                    \shear strains eyz, exz, exy
exz=+[w/x]+[u/z]
exy=+[u/y]+[v/x]
stif                                \stiff matrix section
dist =+[em_i;em_j]*de_i_j*fact+[en_i;en_i]*fact*(0.5-pv)
load=+[u_i]*fd_i+[em_i]*fe_i
end
```

Use Least Square Method to solve the stress field. Its PDE and algorithm files are not listed here.

For the solution of coupled fields, a GCN file is need to define the coupling relation and the corresponding algorithm, which is listed below:

```
Field a-displacement field,
Field b-temperature field
Field c-stress field
defi
a ell b &          \adopt elliptic algorithm, coupled with temperature field b
b npar             \adopt nonlinear parabolic algorithm
c str a b          \adopt Least Square Method, coupled with field a and b
\the following is the computation procedure by command flow
startsin a                    \initialize field a
startsin b                    \initialize field b
IF EXIST stop DEL stop        \iteration for time variable
:1
BFT                           \update the time
IF EXIST END DEL END          \nonlinear iteration to solve temperature field
:2
solvsin b                     \solve the temperature field
IF NOT EXIST END GOTO 2
solvsin a                     \solve the displacement field
stress c                      \use Least Square Method to solve the stress field
IF NOT EXIST stop GOTO 1
```

6. Conclusion

From the above two examples, it can be seen that FEPG, aiming at FEM, is very flexible for automatically generating program. FEPG is suitable to various problems of linear and nonlinear, steady and transient, single field and coupled fields. For nonlinear or coupled problems, it can automatically generate finite element program based on the linearized and time discretized scheme and the decoupling algorithm provided by the user (or directly by calling the algorithm from the library). Therefore, FEPG is a powerful tool for finite element computation.

References

1. Liang Guoping, Finite element program generator and finite element language, *Advans in Mechanics Sinica* **20** (1990).
2. Liang Guoping and Fu Zizhi, The component-programming method and its language, *Chinese J. Comput. Struct. Mech. & Appl. Sinica* **2** (1985).