(CODE. 1 to CODE. 7 are shown at the end of this paper.)
The code below is written by MATLAB 2019b

CODE 1 :Stiffness Matrix calculation A

```
clc:clear all:close all:
syms x y z;
N1=(1+x)*(1-y)*(1-z)/8;
N2=(1+x)*(1+y)*(1-z)/8;
N3=(1-x)*(1+y)*(1-z)/8;
N4=(1-x)*(1-y)*(1-z)/8;
N5=(1+x)*(1-y)*(1+z)/8;
N6=(1+x)*(1+y)*(1+z)/8;
N7=(1-x)*(1+y)*(1+z)/8;
N8=(1-x)*(1-y)*(1+z)/8;
Ns=[N1, N2, N3, N4, N5, N6, N7, N8];
Bs_fai=sym(zeros(3,1,8));
Bs_I=sym(zeros(6,3,8));
for i=1:8:
    Bs_fai(:,:,i)=[diff(Ns(i), x); diff(Ns(i), y); diff(Ns(i), z)];
    Bs_I(::,i)=[diff(Ns(i),x), 0, 0:
        0, diff(Ns(1), y), 0;
        0, 0, diff (Ns(i), z);
        0, diff(Ns(i), z), diff(Ns(i), y);
        diff(Ns(i), z), 0, diff(Ns(i), x);
        diff(Ns(i), y), diff(Ns(i), x), 0];
end
```

CODE 2 :Stiffness Matrix calculation B

```
B_fai=[Bs_fai(:, :, 1), Bs_fai(:, :, 2), Bs_fai(:, :, 3), Bs_fai(:, :, 4), Bs_fai(:, :, 5), Bs_fai(:, :, 6), Bs_fai(:, :, 7), Bs_fai(:, ;, 8)],
B_I=[Bs_I(:, :, 1), Bs_I(:, :, 2), Bs_I(:, :, 3), Bs_I(:, :, 4), Bs_I(:, :, 5), Bs_I(:, :, 6), Bs_I(:, :, 7), Bs_I(:, :, 8)];
C=le9*[86.74 6.99 11.91 -17.91 0 0;
    6.99 86.74 11.91 17.91 0 0;
    11.91 11.91 107.2 0 0 0;
    -17.91 17.91 0 57.94 0 0;
    0 0 0 0 57.94 -17.91;
    0 0 0 0 -17.91 39.88];
DM = 1e-12*[39.21 0 0;
    0 39.21 0;
    0 0 41.03];
e=[0.171 -0.171 0 -0.0406 0 0;
    0 0 0 0 0.0406 -0.171;
```

```
14:         0 0 0 0 0 0];
15:   e=e';
16:   k_II=int(int(int(B_I'*C*B_I, x, -1, 1),y,-1,1),z, -1, 1); k_II=double(k_II);
17:   k_faiI=int(int(int(B_I'*e*B_fai, x-1,1), y, -1, 1) z, -1, 1);
18:   k_faiI=double(k_ faiI);
19:   s = xlswrite('k_II_e.xls', k_II);
20:   s1 = xlswrite('k_faiI_e.xls',k_faiI);
21:   s2 = xlswrite('k_faifai_e.xls',k_faifai);
```

CODE 3: Finite Figure Drawing

```
1:    function draw_finite(nelx, nely, nelz, a, F0)
2:    clf;
3:    U=cal_finite(nelx， nely， nelz， a， F 0);
4:    for c=0：1：nelz
5:    for b=0：1：n ely
6:    a=0：1：nelx;
7:    b1(1：(nelx+1) )=b
8:    c1(1：(nelx+1) ) =c
9:    plot3(a，b1，c1， '--ro');
10:   box on
11:   grid on
12:   hold on
13:   axis equal
14:   xlabel('x');  ylabel('y'); zlabel('z');
15:   end
16:   end
17:   for c2=0：1：nelz
18:   for a2=0：1：nelx
19:   b2=0：1：nely；
20:   c3(1：(nely+1) )=c2：
21:   a3(1：(nely+1) ) =a2；
22:   plot 3(a 3， b 2， c 3， '--ro') ；
23:   box on
24:   grid on
25:   hold on
26:   end
27:   end
28:   for all= 0：1：nelx
29:   for  b11=0：1：nely
30:   c 11=0：1：nelz：
31:   a 12(1：(nelz+1) )
32:   b 12(1：(nelz+1) ) =b 11；
33:   plot 3(a 12， b 12， c 11， '--ro')
34:   box on
```

```
grid on
hold on
end
end
Ulen=size(U， 1) ；
a=Ulen/3：
b=1：a；
Ux=U(3*b(；)-2)*1e9；
Uy=U(3*b(：)-1)*1e9；
Uz=U(3*b(：))*1e9；
xx=1：nel x；
yy=1：n ely；
zz=1：nelz；
aa(1：(nel x+1) )= 0：nel x；
aa=repmat(aa， (nely+1) ， 1) ；
aal=zeros((nely+1) ， (nelx+1) ， (nelz+1) ) ；
for i=1：1：(nelz+1)
aal(:, :, i)=aa;
end
bb=zeros((nely+1) ， (nelx+1) ) ；
for i=1：1：(nely+1)
bb(i， 1：(nelx+1) ) =i-1；
end
bbl=zeros((nely+1) ， (nelx+1) ， (nelz+1) ) ；
for i=1：1：(nelz+1)
bb1(：，：，i) = bb；
end
ccl=zeros((nely+1) ， (nelx+1) ， (nelz+1) ) ；
for i=1：1：(nelz+1)
cc=repmat(i-1， (nely+1) ， (nelx+1) ) ；
ccl(：， ：， i)=cc
end
Uxx=aal+reshape(Ux， (nely+1) ， (nelx+1) ， (nelz+1) )
U yy=bbl+reshape(Uy， (nely+1) ， (nelx+1) ， (nelz+1) )
U zz=cc1+reshape(Uz， (nely+1) ， (nelx+1) ， (nelz+1) )
for i=1：1：(nelz+1)
x1=Uxx(：， ：， i) ；
yl= U yy(：， ：， i) ；
z1=U zz(：， ：， i) ；
plot3(x 1， y 1， z 1， '-ko') ；
box on
grid on
hold on
axis equal
```

```
79:    xlabel('x') ： ylabel('y') ； z label('z') ：
80:    end
81:    for i=1：1：(nelz+1)
82:    x1=Uxx(：， ：， i) ';
83:    y1=Uyy(：， ：， i) ';
84:    z1=Uzz(：， ：， i) ';
85:    plot 3(x 1， yl， z 1， '-ko') ；
86:    box on
87:    grid on
88:    hold on
89:    axis equal
90:    end
91:    for i=1：1：(nely+1)
92:    for j=1：1：(nel x+1)
93:    x1=U xx(i， j， ：) ；
94:    xl =reshape(x1， (nelz+1) ， 1) ；
95:    yl= Uyy(i， j， ：) ；
96:    yl=reshape(yl， (nelz+1) ， 1) ；
97:    zl =Uzz(i， j， ：) ；
98:    zl=reshape(z 1， (nelz+1) ， 1) ；
99:    plot 3(x 1， yl， zl， '-ko') ；
100:   end
101:   end
102:   end
103:   function[u] =cal_finite(nelx， nely， nelz， a， F 0)
104:   %DOI 10.1007/s 00158-014-1107-x
105:   k_II_e=xlsread('k_II_e.xls') ；
106:   k_Ifai_e=xlsread('k_Ifai_e.x1s) ；
107:   k_faifai_e=xlsread('k_faifai_e.xls') ；
108:   k_II=sparse(3*(nelx+1) *(nely+1) *(nelz+1) ， 3*(nelx+1) *(nely+1) *(nelz+1) )
109:   k_Ifai=sparse(3*(nelx+1) *(nely+1) *(nelz+1) ， (nelx+1) *(nely+1) *(nelz+1) ) ；
110:   k_faifai=sparse((nelx+1) *(nely+1) *(nelz+1) ， (nelx+l) *(nely+1) *(nelz+1) ) ：
111:   %USER-DEFINED LOAD DOFs
112:   [il， j1， k1] =meshgrid(nelx， nely， nelz) ；
113:   loadnid=kl*(nelx+1) *(nely+1) +il*(nely+1) +jl+1； %Node IDs
114:   loaddof=3*loadnid(：) -a；
115:   %USER-DEFINED SUPPORT FIXED DOFs
116:   [iif， jf， kf] =meshgrid(0， 0：nely， 0：nelz)          %Coordinates
117:   fixednid=kf*(nelx+l) *(nely+l) +iif*(nely+l) +1+jf：% Node IDs
118:   fixeddof=[3*fixednid(：) ； 3*fixednid(：) -1； 3*fixednid(：) -2] ； %DOFs
119:   %USER-DEFINED ELECTRONIC BOUNDARY DOFs
120:   [ifai， jfai， kfai] =meshgrid(O：nelx， nely， O：nelz) ；
121:   boundnid=kfai*(nelx+1) *(nely+1) +ifai*(nely+1) +1+jfai； %Node IDs
122:   bounddof=boundnid(：) ：%DOFs
```

```
%PREPARE FINITE ELEMENT ANALYSIS
ndof_u=3*(nelx+1) *(nely+1) *(nelz+1) ；
ndof_fai=(nelx+1) *(nely+1) *(nelz+1) ：
F=sparse(1oaddof， 1， F0， ndof_u， 1) ；
U=zeros(ndof_u， 1) ：
Q=zeros(ndof_fai， 1) ；
fai=zeros(ndof_fai， 1) ：
freedofs_u=setdiff(1：ndof_u， fixeddof) ；
freedofs_fai=setdiff(1：ndof_fai， bounddof) ；
For elx=1：nelx
for ely=l：nely
for elz=1：nelz
n1=(nely+1) *(nelx+1) *(elz-1) +(n ely+1)*(elx-1)+ely; n2=(nely+1) *(nelx+1) *(elz-1) +(nely+1)*elx+ely；
n3=(nely+1) *(nelx+1) *elz+(nely+1) *(elx-1) +ely；
n4=(nely+1) *(nelx+1) *elz+(nely+1) *elx+ely；
edof_u=[3*n1-2； 3*n1-1； 3*n1； 3*n1+1； 3*n1+2； 3*n1+3；
3*n2-2：3*n2-1：3*n2：3*n2+1：3*n2+2：3*n2+3：
3*n3-2；3*n3-1；3*n3；3*n3+1；3*n3+2；3*n3+3；
3*n4-2：3*n4-1：3*n4：3*n4+1：3*n4+2：3*n4+3]：
edof_fai=[n1； n1+1； n2； n2+1； n3； n3+1； n4； n4+1] ；
k_II(edof_u， edof_u) =k_II(edof_u， edof_u) +k_II_e；
k_Ifai(edof_u， edof_fai) =k_If a i(edof_u， edof_fai) +k_Ifai_e；
k_faifai(edof_fai， edof_fai) =k_faifai(edof_fai， edof_fai) ...
+k_faifai_e：
end
end
end
usize=size(freedofs_u， 2) ； faisize=size(freedofs_fai， 2) ；
a=zeros((usize+faisize) ， 1) ；
a(1：usize， 1) =le-5：
a((usize+1) ：(usize+faisize) ， 1) =1e5；
P=diag(a， 0) ：
bb=(P*[k_II(freedofs_u， freedofs_u) k_If a i(freedofs_u， freedofs_fai) ； ， ， ，
(k_Ifai(freedofs_u， freedofs_fai) ) '...
-k_faiai(freedofs_fai， freedofs_fai) ] *P) ；
bbl=P*[F(freedofs_u， ：) ； Q(freedofs_fai， ：) ] ；
ufai=bb \ (P*[F(freedofs_u， ：) ：Q(freedofs_fai， ：) ] ) ：
usize=size(freedofs_u， 2) ：faisize=size(freedofs_fai， 2)
ufai=P*ufai：
U(freedofs_u， ：) =ufai(l：usize， ：) ；
fai(freedofs_fai， ：) =ufai((usize+1) ：(usize+faisize) ， ：) ：
end
```

CODE 4: Encoding A

```
for elx = 1:nelx
for ely = 1:nely
for elz = 1:nelz
nl =(nely+1)*(nelx+1)*(elz-1)+(nely+1)*(elx-1)+ely;
n2=(nely+1)*(nelx+1)*(elz-1)+(nely+1)*elx+ely;
n3=(nely+1)*(nelx+1)*elz+(nely+1)*(elx-1)+ely;
n4=(nely+1)*(nelx+1)*elz+(nely+1)*elx+ely;
```

CODE 5: Encoding B

```
edof_u =[3*n1-2;3*n1-1:3*nl;3*n1+1:3*n1+2:3*n1+3;
3*n2-2;3*n2-1;3*n2;3*n2+1;3*n2+2;3*n2+3：
3*n3-2;3*n3-1:3*n3;3*n3+1;3*n3+2;3*n3+3;
3*n4-2;3*n4-1:3*n4;3*n4+1:3*n4+2:3*n4+3]；
edof_fai=  [nl;nl+1：n2;n2+1;n3;n3+1;n4;n4+1]：
```

CODE 6: Assemble Stiffness Matrix

```
k_II(edof_u,edof_u) = k_II(edof_u,edof_u)  +...
        x(elx,ely,elz)^penal_II*k_II_e；
k_Ifai(edof_u,edof_fai) = k_Ifai(edof_u,edof_fai)  +...
        x(elx,ely,elz)^ penal_Ifai*k_Ifai_e；
k_faifai(edof_fai,edof_fai)= k_faifai(edof_fai,edof_fai) +...
        x(elx,ely,elz)^ penal_faifai*k_faifai_e;
```