```
module Lxz_Tools
    implicit none
    integer (kind(1)),parameter ::ikind=(kind(1))
    integer (kind(1)),parameter ::rkind=(kind(0.D0))
    real (rkind),     parameter :: Zero=0.D0,One=1.D0,Two=2.D0,Three=3.D0, &
   &    Four=4.D0,Five=5.D0,Six=6.D0,Seven=7.D0,Eight=8.D0,Nine=9.D0, &
   &    Ten=10.D0

    contains
    function matinv(A) result (B)
        real(rkind) ,intent (in)::A(:,:)
        !real(rkind) , allocatable::B(:,:)
        real(rkind) , pointer::B(:,:)
        integer(ikind):: N,I,J,K
        real(rkind)::D,T
        real(rkind), allocatable::IS(:),JS(:)
        N=size(A,dim=2)
        allocate(B(N,N))
        allocate(IS(N));allocate(JS(N))
        B=A
        do  K=1,N
            D=0.0D0
            do I=K,N
                do J=K,N
                    if(abs(B(I,J))>D) then
                        D=abs(B(I,J))
                        IS(K)=I
                        JS(K)=J
                    end if
                end do
            end do
            do J=1,N
                T=B(K,J)
                B(K,J)=B(IS(K),J)
                B(IS(K),J)=T
            end do
            do I=1,N
                T=B(I,K)
                B(I,K)=B(I,JS(K))
                B(I,JS(K))=T
            end do
            B(K,K)=1/B(K,K)
            do J=1,N
                if(J.NE.K) then
```

```
                B(K,J)=B(K,J)*B(K,K)
            end if
        end do
        do I=1,N
            if(I.NE.K) then
                do J=1,N
                    if(J.NE.K) then
                        B(I,J)=B(I,J)-B(I,K)*B(K,J)
                    end if
                end do
            end if
        end do
        do I=1,N
            if(I.NE.K) then
                B(I,K)=-B(I,K)*B(K,K)
            end if
        end do
    end do
    do K=N,1,-1
        do J=1,N
            T=B(K,J)
            B(K,J)=B(JS(K),J)
            B(JS(K),J)=T
        end do
        do I=1,N
            T=B(I,K)
            B(I,K)=B(I,IS(K))
            B(I,IS(K))=T
        end do
    end do
    return
end function matinv

subroutine IntSwap(a,b)
    integer(ikind),intent(in out)::a,b
    integer(ikind)::t
    t=a;a=b;b=t
end subroutine IntSwap

subroutine RealSwap(a,b)
    real(rkind),intent(in out)::a,b
    real(rkind)::t
    t=a;a=b;b=t
end subroutine RealSwap
```

```
subroutine matprint(A,n)
    real(rkind),intent(in)::A(:,:)
    integer(ikind)::n
    integer(ikind)::n1,n2
    integer(ikind)::i,j
    character(10)::C
    n1=size(A,dim=1)
    n2=size(A,dim=2)
    C='('//trim(itoc(n2))//'E'//trim(itoc(n))//&
    '.'//trim(itoc(n-7))//')'
    do I=1,n1
        write(*,C)(A(I,J),J=1,n2)
    end do
end subroutine matprint

function matdet(B) result(det)
    real(rkind),intent(in)::B(:,:)
    real(rkind)::det
    integer(ikind)::n,i,j,k,is,js
    real(rkind),pointer::A(:,:)
    real(rkind)::f,d,q
    n=size(B,dim=1)
    allocate (A(n,n))
    A=B
    f=1.0D0;        det=1.0D0
    do k=1,n-1
        q=0.0D0
        do i=k,n
            do j=k,n
                if(abs(a(i,j)).gt.q) then
                    q=abs(a(i,j))
                    is=i
                    js=j
                end if
            end do
        end do
        if(q+1.0D0.eq.1.0D0) then
            det=0.0d0
            return
        end if
        if(is.ne.k) then
            f=-f
            do j=k,n
```

```
                d=a(k,j)
                a(k,j)=a(is,j)
                a(is,j)=d
            end do
        end if
        if(js.ne.k) then
            f=-f
            do i=k,n
                d=a(i,js)
                a(i,js)=a(i,k)
                a(i,k)=d
            end do
        end if
        det=det*a(k,k)
        do i=k+1,n
            d=a(i,k)/a(k,k)
            do j=k+1,n
                a(i,j)=a(i,j)-d*a(k,j)
            end do
        end do
    end do
    det=f*det*a(n,n)
    deallocate (a)
    return
end function matdet

function itoc(i1) result (c)
    integer(ikind),intent(in)::i1
    character(len=2)::c
    real(rkind)::x
    integer(ikind) ::n,b,i,j
    i=i1
    x=i
    c(1:2)=' '
    x=log10(x)
    n=int(x)+2
    do j=n-2,0,-1
        b=mod(i,10**j)
        b=(i-b)/(10**j)
        i=i-b*(10**j)
        c(n-j-1:n-j-1)=achar(iachar('0')+b)
    end do
end function itoc
```

```
    subroutine Gauss(GStif,GLoad,GDisp)
        real (rkind),intent (in) :: GStif(:,:),GLoad(:)
          real (rkind),intent (out) :: GDisp(:)
          integer (ikind) :: i,j,k
          integer (ikind) :: N
          real (rkind) :: P,I1,X,Y
          real (rkind),allocatable :: A(:,:)
          N=size(GDisp,dim=1)
          allocate (A(N,N+1))
          A(1:N,1:N)=GStif(1:N,1:N)
          A(1:N,N+1)=GLoad(1:N)
          DO  j=1,N
            P=0.0D0
          DO k=j,N
          IF(ABS(A(k,j)).LE.P) cycle
          P=ABS(A(k,j))
          I1=k
       end do
       IF(P.GE.1E-15)GO TO 230
       WRITE(22,'(A)') 'NO UNIQUE SOLUTION'
       RETURN
230    IF(I1.EQ.j)GO TO 280
       DO 270 K=J,N+1
          X=A(J,K)
          A(J,K)=A(I1,K)
270       A(I1,K)=X
280    Y=1.D0/A(J,J)
       DO 310 K=J,N+1
310       A(J,K)=Y*A(J,K)
       DO 380 I=1,N
          IF(I.EQ.J)GO TO 380
          Y=-A(I,J)
          DO 370 K=J,N+1
370          A(I,K)=A(I,K)+Y*A(J,K)
380    CONTINUE
390 end do

        GDisp=A(1:N,N+1)
        end subroutine Gauss

end module Lxz_Tools


module TypDef
```

```
    use Lxz_Tools
    implicit none

    integer(ikind) :: NNode, NSolid, NShell !节点数量，实体单元数量，壳单元数量
    integer(ikind) :: NMaterial, NRealConstant !材料数量，实参数数量
    integer(ikind) :: NGlbDOF !整体自由度总数

    type Typ_Node !定义节点类型
        real(rkind)    :: coord(3)  !节点坐标
        integer(ikind) :: EleTyp    !从属单元类型 1—soild 单元，2—shell 单元
        integer(ikind)  :: GDOF(6)    !整体自由度编码 如果不从属与 shell，则
GDOF(4:6)=0
        real(rkind)    :: disp(6)   !节点位移
    end type typ_Node

    type Typ_Material !定义材料
        real(rkind) :: E !弹性模量
        real(rkind) :: mu !泊松比
    end type Typ_Material

    type Typ_RealConstant !定义实参数
        real(rkind) :: Thickness !板单元厚度
    end type Typ_RealConstant

!==============================================================================
    type Typ_Plate !定义板单元                                                 !
        real(rkind) :: NCoord(2,4) !节点的局部坐标                             !
        integer(ikind) :: NodeNo(4) !节点编号
        real(rkind) :: t !板厚度                                               !
        real(rkind) :: E,MU !弹性模量                                          !
        real(rkind) :: D(5,5) ![D]矩阵                                         !
        real(rkind) :: B(5,12) ![B]矩阵                                        !
        real(rkind) :: EK(12,12) ![EK]单元刚度矩阵                             !
        real(rkind) :: S(5,12) ![S]单元应力矩阵                                !
        real(rkind) :: GaussPoint(2,4)  !高斯积分点坐标                        !
        real(rkind) :: N(4,4) !形函数矩阵，四个高斯积分点                      !
        real(rkind) :: dN(4,2,4) !形函数矩阵局部坐标系下求导，四个高斯积分点 !
        real(rkind) :: dO(4,2,4) !形函数矩阵整体坐标系下求导，四个高斯积分点 !
        real(rkind) :: Jacobi(2,2,4) !Jacobi 矩阵，四个高斯积分点              !
        real(rkind) :: InvJ(2,2,4) !Jacobi 矩阵的逆矩阵                        !
        real(rkind) :: SJ(4) !|J|，Jacobi 矩阵行列式的值，四个高斯积分点       !
                                                                               !
        !.......................                                                !
    end type Typ_Plate                                                         !
```

```
!================================================================================

    type Typ_Membrance !定义膜单元
        real(rkind) :: NCoord(2,4) !节点的局部坐标                                  !
        integer(ikind) :: NodeNo(4) !节点编号
        real(rkind)::EK(8,8),B(3,8),D(3,3),J(2,2)
        real(rkind)::E,MU,t
        !........................
    end type Typ_Membrance

    type Typ_Solid !定义实体单元
        integer(ikind) :: NodeNo(8) !节点编号
        integer(ikind) :: MatNo !材料号
        real(rkind)    :: E,MU
        real(rkind) :: EK(24,24)
        !........................
    end type Typ_Solid

    type Typ_Shell !定义壳单元
        integer(ikind) :: NodeNo(4) !节点坐标
        integer(ikind) :: MatNo !材料号
        integer(ikind) :: RealNo !实参数号
        real(rkind) :: E,MU,t
        type(typ_Plate) :: S_Plate(1) !Shell 里面的板部分
        type(typ_Membrance) :: S_Membrance(1) !shell 里面膜部分
        real(rkind) :: TransMatrix(24,24)  !坐标转换矩阵
        real(rkind) :: EK(24,24)  !刚度矩阵
        !........................
        real(rkind) :: NCoord(2,4) !节点的局部坐标
    end type Typ_Shell



    type Typ_Load
        integer(ikind) :: NodeNo
        integer(ikind) :: DOF
        real(rkind) :: Value
    end type Typ_Load

    type Typ_Support
        integer(ikind) :: NodeNo
        integer(ikind) :: DOF
    end type Typ_Support
```

```
    contains

    subroutine TypDef_DOFCount(Node, Solid, Shell) !单元自由度编码子程序
        type(Typ_Node)  :: Node(:)
        type(Typ_Solid) :: Solid(:)
        type(Typ_Shell) :: Shell(:)
        integer(ikind) :: i,j,k !循环变量
        integer(ikind) :: TempDOF !总体自由度的工作变量

        Node(:)%EleTyp=1 !假设所有节点都是只从属于实体单元
        do i=1, NNode
            do j=1, NShell
                do k=1,4
                    if(Shell(j)%NodeNo(k)==i) then !如果壳单元 j 的第 k 个节点和 i
节点相同
                        Node(i)%EleTyp=2; ! 那么节点 i 从属于壳单元
                    end if
                end do  !   for k
            end do  !for j
        end do  ! for i

        !以下开始计算各个单元的自由度数量和总体自由度数量
        TempDOF=0 !清空变量
        do i=1, NNode
            if(Node(i)%EleTyp==1) then !如果节点只从属与实体单元
                Node(i)%GDOF(1)=TempDOF+1;  Node(i)%GDOF(2)=TempDOF+2;
Node(i)%GDOF(3)=TempDOF+3;
                Node(i)%GDOF(4:6)=0;
                TempDOF=TempDOF+3; !总体自由度增加了 3 个
            end if
            if(Node(i)%EleTyp==2) then !如果节点从属与壳单元
                Node(i)%GDOF(1)=TempDOF+1;  Node(i)%GDOF(2)=TempDOF+2;
Node(i)%GDOF(3)=TempDOF+3;
                Node(i)%GDOF(4)=TempDOF+4;  Node(i)%GDOF(5)=TempDOF+5;
Node(i)%GDOF(6)=TempDOF+6;
                TempDOF=TempDOF+6; !总体自由度增加了 6 个
            end if
        end do !for i
        NGlbDOF=TempDOF


        return
    end subroutine TypDef_DOFCount
```

```
end module TypDef

module SolidDef
    use Lxz_Tools
    use TypDef

    contains
!****************************************************
!      得到形函数
      SUBROUTINE Solid_SHAP3(U,V,W,XQ,XJAC,XVJ,DETJ,SHP)
!
!****************************************************
! -------------------------------------------------------------
! COMPUTE SHAPE FUNCTION AND DERIVATIVES FOR 3D 8-NODE ELEMENT
! --------------------------------------------------------------
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION SHP(3,8),XQ(3,8),XJAC(3,3),XVJ(3,3),&
               UI(8),VI(8),WI(8),IT2(3),IT3(3)
      DATA UI/-0.5D0,0.5D0,0.5D0,-0.5D0,-0.5D0,0.5D0,0.5D0,-0.5D0/
      DATA VI/-0.5D0,-0.5D0,0.5D0,0.5D0,-0.5D0,-0.5D0,0.5D0,0.5D0/
      DATA WI/-0.5D0,-0.5D0,-0.5D0,-0.5D0,0.5D0,0.5D0,0.5D0,0.5D0/
      DATA IT2/2,3,1/, IT3/3,1,2/
!     变量说明
!       U,V,W 为高斯积分点坐标
!       SHP(1:3,:) 先存放 Ni 对 U, V, W 取偏导，后存放对 X，Y，Z 的偏导
!       SHP(4,:) 为 Ni
!       XQ(:,8) 为节点坐标
!       XJAC 为雅可比矩阵，XVJ 为雅可比逆矩阵
!       求 Ni 及 Ni 对 U,V,W 的偏导

      DO 10 I=1,8
!     SHP(4,I)=(0.5D0+UI(I)*U)*(0.5D0+VI(I)*V)*(0.5D0+WI(I)*W)
      SHP(1,I)=UI(I)*(0.5D0+VI(I)*V)*(0.5D0+WI(I)*W)
      SHP(2,I)=VI(I)*(0.5D0+UI(I)*U)*(0.5D0+WI(I)*W)
      SHP(3,I)=WI(I)*(0.5D0+UI(I)*U)*(0.5D0+VI(I)*V)
10    CONTINUE
!       求雅可比矩阵
      DO 20 I=1,3
      DO 20 J=1,3
      XJAC(I,J)=0.0D0
      DO 20 K=1,8
20    XJAC(I,J)=XJAC(I,J)+XQ(I,K)*SHP(J,K)
!       求雅可比矩阵行列式
      WJ1=XJAC(1,1)*XJAC(2,2)*XJAC(3,3)+XJAC(3,1)*XJAC(1,2)*&
```

```
      XJAC(2,3)+XJAC(1,3)*XJAC(2,1)*XJAC(3,2)
    WJ2=XJAC(1,3)*XJAC(3,1)*XJAC(2,2)+XJAC(1,2)*XJAC(2,1)*&
      XJAC(3,3)+XJAC(2,3)*XJAC(3,2)*XJAC(1,1)
    DETJ=WJ1-WJ2
!          得到雅可比逆矩阵
    DO 25 I=1,3
    DO 25 J=1,3
    M2=IT2(I)
    M3=IT3(I)
    N2=IT2(J)
    N3=IT3(J)
25  XVJ(I,J)=(XJAC(M2,N2)*XJAC(M3,N3)-XJAC(M2,N3)&
      *XJAC(M3,N2))/DETJ
!          W1，W2，W3 为临时变量，存放 Ni 对 X，Y，Z 的偏导
    DO 30 I=1,8
    W1=0.0D0
    W2=0.0D0
    W3=0.0D0
    DO 35 K=1,3
    W1=W1+XVJ(1,K)*SHP(K,I)
    W2=W2+XVJ(2,K)*SHP(K,I)
35  W3=W3+XVJ(3,K)*SHP(K,I)
!          把 Ni 对 X,Y,Z,的偏导存入 SHP
    SHP(1,I)=W1
    SHP(2,I)=W2
    SHP(3,I)=W3
30  CONTINUE
    RETURN
    END subroutine
!---------------------------------------------------
!

!   得到矩阵 B(6,24)
   Subroutine Solid_GETB(B,SHP)
      implicit real*8(A-H,O-Z)
      dimension B(6,24),SHP(3,8)
      integer i,j
      do 10 i=1,8
          j=(i-1)*3+1
          b(1,j)=SHP(1,I)
          B(1,J+1)=0.0d0
          B(1,J+2)=0.0D0

          B(2,J)=0.0d0
```

```
          B(2,J+1)=SHP(2,I)
          B(2,J+2)=0.0D0

          B(3,J)=0.0d0
          B(3,J+1)=0.0d0
          B(3,J+2)=SHP(3,I)

          B(4,J)=SHP(2,I)
          B(4,J+1)=SHP(1,I)
          B(4,J+2)=0.0D0

          B(5,J)=0.0d0
          B(5,J+1)=SHP(3,I)
          B(5,J+2)=SHP(2,I)

          B(6,J)=SHP(3,I)
          B(6,J+1)=0.0d0
          B(6,J+2)=SHP(1,I)
10    continue
      return
      end subroutine
!**********************************************************
!**********************************************************
      subroutine Solid_MutBAB(M,N,A,B,C)
!**********************************************************
!**********************************************************
!**********************************************************
      implicit real*8 (A-H,O-Z)
      Dimension A(N,N),C(M,M),B(N,M),AB(N,M)
!
      do 12 I=1,N
      do 12 J=1,M
      W1=0.0d0
      DO 14 K=1,N
14    W1=W1+A(I,K)*B(K,J)
12    AB(I,J)=W1
      DO 16 I=1,M
      Do 16 J=1,M
      W2=0.0D0
      Do 18 K=1,N
18    W2=W2+B(K,I)*AB(K,J)
16    C(I,J)=W2
      return
      end subroutine
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

    subroutine Solid_GetDB(DB,D,B)
        implicit none
        real*8 DB(6,24),D(8,6,6),B(6,24)
        integer i,j,k,l
        do 5 i=1,6
        do 5 j=1,24
5       DB(I,J)=0.0d0
        Do 10 I=1,6
            do 20 J=1,24
                L=(J-1)/3+1
                    Do 30 K=1,6
                        DB(I,J)=DB(I,J)+D(L,I,K)*B(K,J)
30              continue
20          continue
10      continue
        return
    end subroutine

!*********************************************************************

    subroutine Solid_GetBDB(BDB,B,D)
        implicit none
        real*8  BDB(24,24),B(6,24),D(6,6)
        BDB=Matmul(matmul(transpose(B),D),B)
        return
    end subroutine

!*********************************************************************
    subroutine Solid_GetD(D,E,EMU)
        implicit real*8(A-H,O-Z)
        Dimension D(6,6)
        integer I,J
        D=0.0d0
        Temp=E*(1-EMU)/((1+EMU)*(1-2*EMU))
        D(1,1)=1
        D(2,2)=1
        D(3,3)=1
        D(1,2)=EMU/(1-EMU)
        D(2,1)=D(1,2)
        D(1,3)=EMU/(1-EMU)
        D(3,1)=D(1,3)
```

```
      D(2,3)=EMU/(1-EMU)
      D(3,2)=D(2,3)
      D(4,4)=(1-2*EMU)/(2*(1-EMU))
      D(5,5)=D(4,4)
      D(6,6)=D(4,4)
      D=temp*D
      return
   end subroutine

!*****************************************************************

   subroutine Solid_GetEK(EK,XQ,E0,EMU0)
      implicit none
      real*8 XQ(3,8),XJAC(3,3),XVJ(3,3),SHP(3,8)
      real*8 B(6,24),DB(6,24),BDB(24,24),D(6,6),DISP(24)
      real*8 E0,EMU0
      real*8 EK(24,24),PLASTICD(8,6,6)
      real*8 U,V,W,H1,H2,H3,DETJ
      real*8 I,J,K,II,JJ,KK
      integer RETVAL

      DO 5 II=1,24
      DO 5 JJ=1,24
5     EK(II,JJ)=0.0d0
      DO 10 I=1,3
      IF(I.EQ.1) U=0.77459669241483d0
      if(I.eq.1) H1=0.555555555555555d0
      if(I.eq.2) U=0.0d0
      if(I.eq.2) H1=0.888888888888889d0
      if(I.eq.3) U=-0.77459669241483d0
      if(i.eq.3) H1=0.555555555555555d0

      do 20 J=1,3
      IF(J.EQ.1) V=0.77459669241483d0
      if(J.eq.1) H2=0.555555555555555d0
      if(J.eq.2) V=0.0d0
      if(J.eq.2) H2=0.888888888888889d0
      if(J.eq.3) V=-0.77459669241483d0
      if(J.eq.3) H2=0.555555555555555d0

      do 30 K=1,3
      IF(K.EQ.1) W=0.77459669241483d0
      if(K.eq.1) H3=0.555555555555555d0
      if(K.eq.2) W=0.0d0
```

```
        if(K.eq.2) H3=0.888888888888889d0
        if(K.eq.3) W=-0.774596669241483d0
        if(K.eq.3) H3=0.555555555555555d0

        call Solid_SHAP3(U,V,W,XQ,XJAC,XVJ,DETJ,SHP)
        call Solid_GetB(B,SHP)
        call Solid_GetD(D,E0,EMU0)
        call Solid_GetBDB(BDB,B,D)

        do 100 II=1,24
        do 100 JJ=1,24
100     EK(II,JJ)=EK(II,JJ)+H1*H2*H3*BDB(II,JJ)*DETJ
30      continue
20      continue
10      continue
        return
    end subroutine

    subroutine Solid_EK(Solid,Node)
        type(typ_Solid) :: Solid(:)
        type(Typ_Node)  :: Node(:)
        integer(ikind)  :: i,j,k
        real(rkind)     :: XQ(3,8)
        do i=1,size(Solid)
            do j=1,8
                XQ(:,j)=Node(Solid(i)%NodeNo(J))%Coord
            end do !for j
            call Solid_GetEK(Solid(i)%EK,XQ,Solid(i)%E,Solid(i)%MU)
        end do !for i
        return
    end subroutine
end module

program Main
    use lxz_Tools
    use TypDef
    use SolidDef
    use IMSL
    implicit none

    type(typ_Solid),pointer :: Solid(:)
    type(Typ_Node), pointer :: Node(:)

    real(rkind),pointer :: GK(:,:), GF(:), GD(:)
```

```
real(rkind) :: temp
integer(ikind) :: NElem, NSupport, NLoad;
integer(ikind) :: i,j,k,l,m

open(55, file='datain.txt')
read(55,*)
read(55,*) NNode, NElem, NSupport, NLoad

allocate(Node(NNode))
allocate(Solid(NElem))
allocate(GK(3*NNode,3*NNode))
allocate(GF(3*NNode))
allocate(GD(3*NNode))

read(55,*)
do i=1,size(Node)
    read(55,*) j, Node(i)%Coord(1:3)
end do
read(55,*)
do i=1,size(Solid)
    read(55,*) j, solid(i)%NodeNo
!   do k=1,4
!       Plate(i)%NCoord(:,k)=Node(Plate(i)%NodeNo(k))%Coord(1:2)
!   end do
end do
solid(:)%E=210.0D9; solid(:)%MU=0.3D0;
call Solid_EK(Solid,Node)
GK=0.0d0; GF=0.0d0; GD=0.0d0
do i=1,size(Solid)
    do j=1,8
        do k=1,8
        do l=1,3
        do m=1,3
            GK((Solid(i)%NodeNo(j)-1)*3+l,(Solid(i)%NodeNo(k)-1)*3+m)=&
                GK((Solid(i)%NodeNo(j)-1)*3+l,(Solid(i)%NodeNo(k)-1)*3+m)+&
                Solid(i)%Ek((j-1)*3+l,(k-1)*3+m)

        end do ! for m
        end do ! for l
        end do ! for k
    end do ! for j
end do ! for i

GF(58)=1000;
```

```
    temp=maxval(GK)
    GK(1,1)=GK(1,1)+1.0D5*temp;
    GK(2,2)=GK(2,2)+1.0D5*temp;
    GK(3,3)=GK(3,3)+1.0D5*temp;
    GK(4,4)=GK(4,4)+1.0D5*temp;
    GK(5,5)=GK(5,5)+1.0D5*temp;
    GK(6,6)=GK(6,6)+1.0D5*temp;
    GK(16,16)=GK(16,16)+1.0D5*temp;
    GK(17,17)=GK(17,17)+1.0D5*temp;
    GK(18,18)=GK(18,18)+1.0D5*temp;
    call DLSARG (size(GF), GK, size(GF), GF, 1, GD)
!   open (77,file='dataout1.txt')
!       write(77,*) i,shell(1)%NCoord
!   close(77)

    open (77,file='dataout.txt')
        do i=1,NNode
            !write(77,*) i
            write(77,'(I3,3E12.4)') i,GD((i-1)*3+1),GD((i-1)*3+2),GD((i-1)*3+3)
        end do
    close(77)
    stop
    stop
end program
```