```
module Lxz_Tools !通用子程序模块
    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 !数据结构定义
    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
    !........................
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
    type(typ_Plate) :: S_Plate !Shell 里面的板部分
    type(typ_Membrance) :: S_Membrance !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
    !........................

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 MembranceDef !膜单元程序
    use lxz_Tools
```

```
    use TypDef
    implicit none


    contains

    subroutine Membrance_EK(Membrance)
        type(typ_Membrance),intent(in out)::Membrance(:)
        real(rkind)::J1(2,4),J2(4,2),Temp(2,1),InvJ(2,2)
        integer(ikind)::i,j,k
        real(rkind)::r,s
        do k=1,size(Membrance)
            Membrance(k)%EK=0d0
            Membrance(k)%B=0d0
            Membrance(k)%D(1,1)=1D0
            Membrance(k)%D(2,2)=1d0
            Membrance(k)%D(1,2)=Membrance(k)%Mu
            Membrance(k)%D(2,1)=Membrance(k)%Mu
            Membrance(k)%D(3,1)=0d0
            Membrance(k)%D(3,2)=0d0
            Membrance(k)%D(2,3)=0d0
            Membrance(k)%D(1,3)=0d0;
            Membrance(k)%D(3,3)=(1d0-Membrance(k)%Mu)/2d0;

    Membrance(k)%D=(Membrance(k)%E/(1.0d0-Membrance(k)%Mu*Membrance(k)%Mu))*Memb
rance(k)%D;
            do i=1,2
                do j=1,2
                    r=0.577350269189626D0*(-1D0)**i
                    s=0.577350269189626D0*(-1D0)**j
                    J1(1,:)=(/-(1d0-s),(1d0-s),(1d0+s),-(1d0+s)/)
                    J1(2,:)=(/-(1d0-r),-(1d0+r),(1d0+r),(1d0-r)/)

    J2(1,:)=(/Membrance(k)%NCoord(1,1),Membrance(k)%NCoord(2,1)/)

    J2(2,:)=(/Membrance(k)%NCoord(1,2),Membrance(k)%NCoord(2,2)/)

    J2(3,:)=(/Membrance(k)%NCoord(1,3),Membrance(k)%NCoord(2,3)/)

    J2(4,:)=(/Membrance(k)%NCoord(1,4),Membrance(k)%NCoord(2,4)/)
                    Membrance(k)%J=(0.25D0)*(matmul(J1,J2))
                    Temp(1,1)=-0.25D0*(1-s);     Temp(2,1)=-0.25D0*(1-r);
                    InvJ=matinv(Membrance(k)%J)
                    Temp=matmul(InvJ,Temp);
```

```
                    Membrance(k)%B(1,1)=Temp(1,1);
    Membrance(k)%B(2,2)=Temp(2,1);
                    Membrance(k)%B(3,1)=Temp(2,1);
    Membrance(k)%B(3,2)=Temp(1,1);
                    Temp(1,1)=0.25D0*(1-s); Temp(2,1)=-0.25D0*(1+r);
                    Temp=matmul(InvJ,Temp);
                    Membrance(k)%B(1,3)=Temp(1,1);
    Membrance(k)%B(2,4)=Temp(2,1);
                    Membrance(k)%B(3,3)=Temp(2,1);
    Membrance(k)%B(3,4)=Temp(1,1);
                    Temp(1,1)=0.25D0*(1+s); Temp(2,1)=0.25D0*(1+r);
                    Temp=matmul(InvJ,Temp);
                    Membrance(k)%B(1,5)=Temp(1,1);
    Membrance(k)%B(2,6)=Temp(2,1);
                    Membrance(k)%B(3,5)=Temp(2,1);
    Membrance(k)%B(3,6)=Temp(1,1);
                    Temp(1,1)=-0.25D0*(1+s);    Temp(2,1)=0.25D0*(1-r);
                    Temp=matmul(InvJ,Temp);
                    Membrance(k)%B(1,7)=Temp(1,1);
    Membrance(k)%B(2,8)=Temp(2,1);
                    Membrance(k)%B(3,7)=Temp(2,1);
    Membrance(k)%B(3,8)=Temp(1,1);
                    Membrance(k)%EK=Membrance(k)%EK+&
                        (&
                        matmul(matmul(transpose(Membrance(k)%B),&
                        Membrance(k)%D),Membrance(k)%B)&
                        )*matdet(Membrance(k)%J)*Membrance(k)%t
                end do
            end do
        end do
    end subroutine Membrance_EK

end module MembranceDef

module PlateDef !板单元程序
    use lxz_Tools
    use TypDef
    implicit none


    contains

    subroutine Plate_D(Plate)  !形成[D]矩阵
        type(Typ_Plate)  :: Plate(:)
```

```fortran
    integer(ikind)   :: i !循环变量
    real(rkind)     :: D0; !临时变量
    do i=1,size(Plate) !得到本构矩阵[D],
        Plate(i)%D=0.0d0;
        D0 = Plate(i)%E*Plate(i)%t*Plate(i)%t*Plate(i)%t/12.0d0/&
            (1.0d0-Plate(i)%MU*Plate(i)%MU)
        Plate(i)%D(1,1) = D0
        Plate(i)%D(2,2) = D0
        Plate(i)%D(3,3) = D0*(1-Plate(i)%MU)/2
        Plate(i)%D(1,2) = D0*Plate(i)%MU
        Plate(i)%D(2,1) = D0*Plate(i)%MU
        Plate(i)%D(4,4) = Plate(i)%E/2/(1+Plate(i)%MU)*Plate(i)%t/(6.0/5.0)
        Plate(i)%D(5,5) = Plate(i)%E/2/(1+Plate(i)%MU)*Plate(i)%t/(6.0/5.0)
    end do
    return
end subroutine Plate_D

subroutine Plate_N(Plate) !形成形函数矩阵以及形函数矩阵对局部坐标求导矩阵
    type(Typ_Plate) :: Plate(:)
    integer (ikind) :: i,j !循环变量
    do i=1,size(Plate)

        Plate(i)%GaussPoint(1,1)=+0.577350269189626D0
        Plate(i)%GaussPoint(2,1)=+0.577350269189626D0
        Plate(i)%GaussPoint(1,2)=-0.577350269189626D0
        Plate(i)%GaussPoint(2,2)=+0.577350269189626D0
        Plate(i)%GaussPoint(1,3)=-0.577350269189626D0
        Plate(i)%GaussPoint(2,3)=-0.577350269189626D0
        Plate(i)%GaussPoint(1,4)=+0.577350269189626D0
        Plate(i)%GaussPoint(2,4)=-0.577350269189626D0

        do j=1,4
            !设置形函数 N 的值，见江老师书 101 页

    Plate(i)%N(1,j)=0.25d0*(1-Plate(i)%GaussPoint(1,j))*(1-Plate(i)%GaussPoint(2
,j))

    Plate(i)%N(2,j)=0.25d0*(1+Plate(i)%GaussPoint(1,j))*(1-Plate(i)%GaussPoint(2
,j))

    Plate(i)%N(3,j)=0.25d0*(1+Plate(i)%GaussPoint(1,j))*(1+Plate(i)%GaussPoint(2
,j))

    Plate(i)%N(4,j)=0.25d0*(1-Plate(i)%GaussPoint(1,j))*(1+Plate(i)%GaussPoint(2
```

```
,j))
                !设置 dNdcosi 的值，见江老师书 101 页
                Plate(i)%dN(1,1,j)=-0.25d0*(1-Plate(i)%GaussPoint(2,j))
                Plate(i)%dN(2,1,j)= 0.25d0*(1-Plate(i)%GaussPoint(2,j))
                Plate(i)%dN(3,1,j)= 0.25d0*(1+Plate(i)%GaussPoint(2,j))
                Plate(i)%dN(4,1,j)=-0.25d0*(1+Plate(i)%GaussPoint(2,j))
                !设置 dNdata 的值，见江老师书 101 页
                Plate(i)%dN(1,2,j)=-0.25d0*(1-Plate(i)%GaussPoint(1,j))
                Plate(i)%dN(2,2,j)=-0.25d0*(1+Plate(i)%GaussPoint(1,j))
                Plate(i)%dN(3,2,j)= 0.25d0*(1+Plate(i)%GaussPoint(1,j))
                Plate(i)%dN(4,2,j)= 0.25d0*(1-Plate(i)%GaussPoint(1,j))
            end do !for j
        end do ! for i
        return
    end subroutine Plate_N

    subroutine  Plate_jacobi (Plate) !设置 Jacobi 相关变量的值
        type(Typ_Plate) :: Plate(:)
        integer(ikind) :: i,j,k !循环变量
        do i=1,size(plate)
            do j=1,4 !对高斯积分点循环

    !Plate(i)%Jacobi(:,:,j)=matmul(transpose(Plate(i)%dN(:,:,j)),Plate(i)%NCoord
(:,:))

    Plate(i)%Jacobi(1,1,j)=dot_product(Plate(i)%dN(:,1,j),Plate(i)%NCoord(1,:))

    Plate(i)%Jacobi(2,1,j)=dot_product(Plate(i)%dN(:,2,j),Plate(i)%NCoord(1,:))

    Plate(i)%Jacobi(1,2,j)=dot_product(Plate(i)%dN(:,1,j),Plate(i)%NCoord(2,:))

    Plate(i)%Jacobi(2,2,j)=dot_product(Plate(i)%dN(:,2,j),Plate(i)%NCoord(2,:))
                Plate(i)%InvJ(:,:,J)=matinv(Plate(i)%Jacobi(:,:,j));
                do k=1,4

    Plate(i)%d0(k,1,j)=dot_product(Plate(i)%InvJ(1,:,j),Plate(i)%dN(k,:,j));

    Plate(i)%d0(k,2,j)=dot_product(Plate(i)%InvJ(2,:,j),Plate(i)%dN(k,:,j));
                end do !for k
                Plate(i)%SJ(j)=Plate(i)%Jacobi(1,1,j)*Plate(i)%Jacobi(2,2,j)-&
                    Plate(i)%Jacobi(2,1,j)*Plate(i)%Jacobi(1,2,j)
            end do !for j
        end do ! for i
        return
```

```
    end subroutine Plate_jacobi

    subroutine Plate_EK(Plate)
        type(Typ_Plate) :: Plate(:)
        integer(ikind) :: i,j,k !循环变量
        call Plate_D(Plate)
        call Plate_N(Plate)
        call Plate_jacobi (Plate)
        do i=1,size(Plate) !对板单元数循环
            Plate(i)%EK=0.0d0;
            do j=1,4 !对高斯积分点数循环
                Plate(i)%B=0.0d0;
                Plate(i)%S=0.0d0;
                do k=1,4 !对节点循环
                    Plate(i)%B(1,(k-1)*3+1)=-Plate(i)%d0(k,1,j);
                    Plate(i)%B(2,(k-1)*3+2)=-Plate(i)%d0(k,2,j);
                    Plate(i)%B(3,(k-1)*3+1)=-Plate(i)%d0(k,2,j);
                    Plate(i)%B(3,(k-1)*3+2)=-Plate(i)%d0(k,1,j);
                    Plate(i)%B(4,(k-1)*3+1)=-Plate(i)%N(k,j);
                    Plate(i)%B(4,(k-1)*3+3)= Plate(i)%d0(k,1,j);
                    Plate(i)%B(5,(k-1)*3+2)=-Plate(i)%N(k,j);
                    Plate(i)%B(5,(k-1)*3+3)= Plate(i)%d0(k,2,j);
                end do !for k
                Plate(i)%S=matmul(Plate(i)%D,Plate(i)%B);

    Plate(i)%EK=Plate(i)%EK+matmul(transpose(Plate(i)%B),Plate(i)%S)*Plate(i)%SJ(j);
            end do !for j
        end do !for i
        return
    end subroutine Plate_EK


end module

module ShellDef !壳单元程序
    use lxz_Tools
    use TypDef
    use PlateDef
    use MembranceDef
    implicit none


    contains
```

```
    subroutine Shell_TransMatrix(Shell,Node)
        type(typ_Node)    :: Node(:)
        type(Typ_Shell)   :: Shell(:)
        real(rkind)  :: namtax(3),namtay(3),namtaz(3),namta(3,3) !坐标变化矩阵
        real(rkind)  :: X12(3),X13(3)
        real(rkind)  :: A,B,C,S
        integer(ikind)    :: i,j,k

        do i=1,size(Shell)
            X12=Node(Shell(i)%NodeNo(2))%Coord-Node(Shell(i)%NodeNo(1))%Coord
            X13=Node(Shell(i)%NodeNo(3))%Coord-Node(Shell(i)%NodeNo(1))%Coord
            A=X12(2)*X13(3)-X13(2)*X12(2);
            B=X12(3)*X13(1)-X13(3)*X12(1);
            C=X12(1)*X13(2)-X13(1)*X12(2);
            S=sqrt(A**2+B**2+C**2);
            namtaz(1)=A/S; namtaz(2)=B/S;   namtaz(3)=C/S;
            namtax=X12;
            namtax=namtax/(sqrt(X12(1)**2+X12(2)**2+X12(3)**2))
            namtay(1)=namtaz(2)*namtax(3)-namtaz(3)*namtax(2);
            namtay(2)=namtaz(3)*namtax(1)-namtaz(1)*namtax(3);
            namtay(3)=namtaz(1)*namtax(2)-namtaz(2)*namtax(1);
            namta(:,1)=namtax;
            namta(:,2)=namtay;
            namta(:,3)=namtaz;
            Shell(i)%TransMatrix(1:3,1:3)=namta
            Shell(i)%TransMatrix(4:6,4:6)=namta
            Shell(i)%TransMatrix(7:9,7:9)=namta
            Shell(i)%TransMatrix(10:12,10:12)=namta
            Shell(i)%TransMatrix(13:15,13:15)=namta
            Shell(i)%TransMatrix(16:18,16:18)=namta
            Shell(i)%TransMatrix(19:21,19:21)=namta
            Shell(i)%TransMatrix(22:24,22:24)=namta
        end do !for i

        return
    end subroutine Shell_TransMatrix
end module
```